



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Magistrale in Informatica

Resilient and Secure Cyber-Physical System

ANALISI DI UN SOTTOSISTEMA DI POSIZIONAMENTO FERROTRAMVIARIO

ANALYSIS OF A TRAMWAY POSITIONING SUBSYSTEM

ALEX FOGLIA

ANDREA BONDAVALLI

Anno Accademico 2018-2019

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	Introduzione	9
1.1	Dependability	10
1.2	Osservazione dei sistemi	13
1.3	Fault Injection	15
1.4	Scopo della Tesi	17
2	Stato dell'arte	19
2.1	Sistemi Ferroviari e Ferrotramviari	19
2.1.1	La safety nei sistemi ferroviari	20
2.2	Il problema del posizionamento	21
2.2.1	Odierne Tecniche di Posizionamento	21
2.2.2	Verso ETCS-3	23
2.3	Cyber Physical Systems of Systems	25
3	Il Sistema Analizzato	29
3.1	Descrizione generale	29
3.2	Constituent Systems	30
3.2.1	Sensor Set	30
3.2.2	Piattaforma di elaborazione dati	32
3.2.3	OBCU	32
3.3	Specifiche delle Interfacce	33
3.3.1	RUI	33
3.4	Interazioni	34
3.4.1	Acquisizione dei dati	35
3.4.2	Trasmissione della posizione	36
3.4.3	Interazioni con eventuali operatori umani	36
3.5	Architettura Software	37
3.5.1	Sensor Fusion Library	37
3.5.2	Listener	38
3.5.3	Interface Modules	38
4	Ambiente di Analisi	41
4.1	Principi di base	41
4.2	Software Impiegati	41
4.2.1	SensorFusionLib	41
4.2.2	SynthDataGen	42
4.2.3	Rail Track Tool	42
5	Parte Sperimentale	49

5.1	Misure di interesse	49
5.2	Esperimenti	50
5.2.1	Scenario 1	50
5.2.2	Scenario 2	52
5.2.3	Scenario 3	55
6	Conclusioni	59

ELENCO DELLE TABELLE

Tabella 1	Livelli SIL	21
Tabella 2	Specifiche Tecniche NVidia TX-Jetson	32
Tabella 3	Specifiche delle RUPI del sistema	33
Tabella 4	Specifiche delle RUMI del sistema	35
Tabella 5	Moduli di <i>interface-modules</i>	39
Tabella 6	Scenario 1, Esperimento 1.1	50
Tabella 7	Esperimento 1.1: Risultati	51
Tabella 8	Scenario 1, Esperimento 1.2	51
Tabella 9	Esperimento 1.2: Risultati	51
Tabella 10	Scenario 2, Esperimento 2.1	52
Tabella 11	Esperimento 2.1: Risultati	52
Tabella 12	Esperimento 2.1: Confronto con esperimento 1.2	52
Tabella 13	Scenario 2, Esperimento 2.2	53
Tabella 14	Esperimento 2.2: Risultati	53
Tabella 15	Esperimento 2.2: Confronto con esperimento 1.2	53
Tabella 16	Scenario 2, Esperimento 2.3	54
Tabella 17	Esperimento 2.3: Risultati	54
Tabella 18	Esperimento 2.3: Confronto con esperimento 1.2	55
Tabella 19	Scenario 3, Esperimento 3.1	55
Tabella 20	Esperimento 3.1: Risultati	55
Tabella 21	Esperimento 3.1: Confronto con esperimento 1.2	56
Tabella 22	Scenario 3, Esperimento 3.2	56
Tabella 23	Esperimento 3.2: Risultati	56
Tabella 24	Esperimento 3.2: Confronto con esperimento 1.2	57

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	Il ciclo di vita dei sistemi: modello a V	13
Figura 5	Monitoring black-box	14
Figura 6	Monitoring white-box	14
Figura 7	Elementi di un processo <i>fault injection</i>	15
Figura 8	Schema di un tipico scenario ferrotramviario	20
Figura 9	Livelli ETCS	23
Figura 10	Schema dell'ambiente di <i>fault injection</i>	24
Figura 11	Interfacce di un CPS	26
Figura 12	RUMI e RUPI	27
Figura 13	Schema SFA	29
Figura 14	<i>Inertial Measurement Unit</i>	31
Figura 15	Ricevitore GPS ublox EVK-M8T	31
Figura 16	Nvidia TX-Jetson	32
Figura 17	Modem TP-LINK M7350 LTE-4G	34
Figura 18	Diagramma a blocchi del sottosistema di posizionamento	34
Figura 19	Sequenza di acquisizione dati	35
Figura 20	Sequenza di trasmissione della posizione	36
Figura 21	Diagramma a blocchi di <i>listener</i>	38
Figura 22	Diagramma a blocchi del sistema software	39
Figura 23	Schema di SynthDataGen	42
Figura 24	Schema riassuntivo RTT	43
Figura 25	Interfaccia RTT	44
Figura 26	Pannello di configurazione generale <i>SynthDataGen</i>	45
Figura 27	Pannello di configurazione IMU simulato	46
Figura 28	Posizione del treno mostrata sulla mappa	46
Figura 29	Grafico valori di accelerazione assi X e Y	47
Figura 30	Grafico errore sulla stima della posizione	47
Figura 31	Traccia di analisi	49

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*.

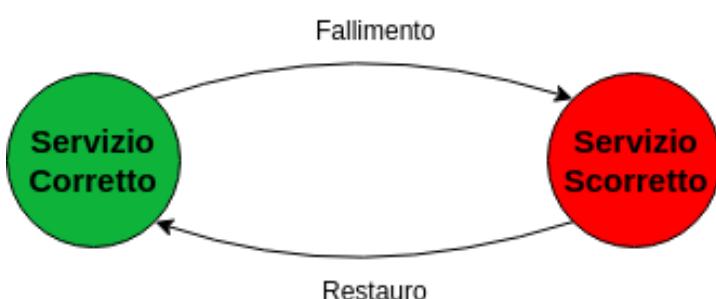


Figura 1: Fallimento e restauro

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

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 $[0, t]$

- *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.

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. Quando esiste questa possibilità, il sistema viene definito *safety-critical*.[2]

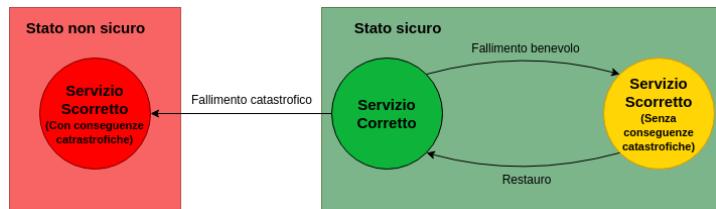


Figura 2: La *safety* estende il concetto di *reliability*

Esistono numerosi contesti in cui i sistemi impiegati vengono definiti *safety-critical*, uno di questi è il trasporto ferroviario.

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*.

La validazione 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: [3]

- 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 modelli analitici *state-based* basati su processi casuali, come ad esempio le Catene di Markov. Si rilassano 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

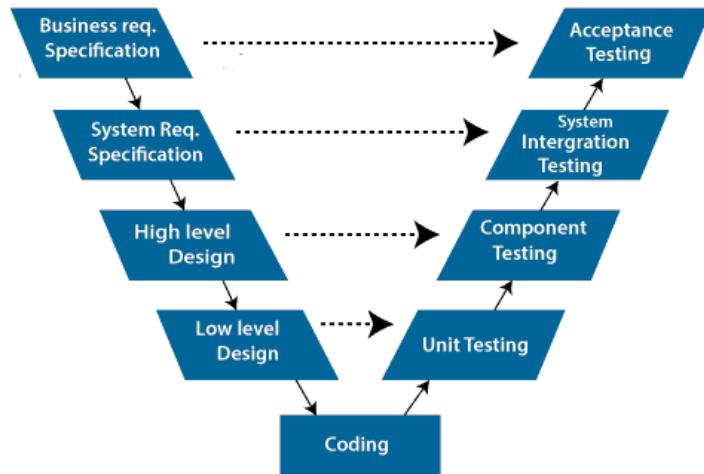


Figura 4: Il ciclo di vita dei sistemi: modello a V

direttamente il comportamento per effettuare misure sperimentali. L'attività di osservazione e misurazione prende il nome di *monitoring*;

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

1.2 OSSERVAZIONE DEI SISTEMI

L'osservazione, o *monitoring*, è una tecnica per valutare la *dependability* di un sistema o un suo prototipo osservandone l'esecuzione nell'ambiente finale.

In questa sezione viene esposto il *monitoring* dei sistemi informatici come descritto nei lavori [4], [5] e [6].

L'obiettivo è quello di monitorare costantemente l'esecuzione del sistema, verificando che il comportamento osservato sia conforme alle aspettative. Un'attività di *monitoring* è seguita da un'attività di verifica. Questa può essere fatta durante l'esecuzione del sistema oppure *offline*, esaminando successivamente i risultati.

Il *monitoring* è stato riconosciuto come una valida opzione per valutare gli attributi di *dependability* dei sistemi informatici.

I problemi tecnici da affrontare e risolvere quando si crea un sistema di *monitoring* riguardano:

- Individuazione e classificazione degli eventi di interesse, al fine di raccogliere misurazioni utili per valutare correttamente la

dependability del sistema monitorato;

- Trasmissione delle informazioni al luogo dove queste saranno elaborate;
- Filtraggio e classificazione degli eventi rispetto alle misure di interesse.

Si definisce *target system* il sistema al quale vengono applicate le attività di *monitoring*. Il componente hardware o software interno al sistema verso il quale si riferisce il *monitoring*, viene detto *target component* o *target application*.

Esistono due differenti configurazioni di un processo di monitoring: *black-box* o *white-box*. Nella configurazione a *black-box* il sistema viene sottoposto a un certo carico di input al fine di osservare l'output prodotto. L'input fornito al sistema prende il nome di *workload*, ed è il carico di informazioni che il sistema dovrà processare durante il *monitoring*.

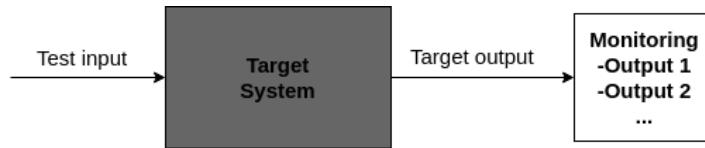


Figura 5: Monitoring black-box

Il monitoring *white-box* prevede l'utilizzo aggiuntivo di strumenti necessari a osservare anche gli output intermedi che il sistema produce. Questi strumenti sono chiamati sonde, o *probes*.

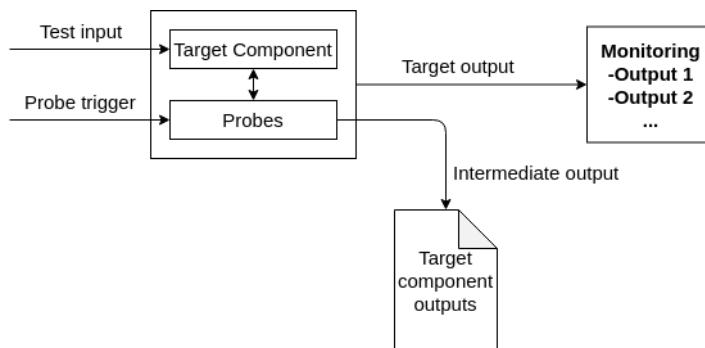


Figura 6: Monitoring white-box

I *probes* sono collocati solitamente all'interno del sistema e il loro scopo è quello di fornire informazioni più dettagliate sul comportamento del

target component.

Se si vogliono monitorare segnali hardware interni al sistema, i *probes* utilizzati saranno effettivamente hardware, mentre si utilizzano *probe* software se si vuole ottenere informazioni sull'esecuzione interna di un programma. Questa tecnica è chiamata instrumentazione del codice, e il codice aggiunto prende il nome di *software probe*.

Quando si progetta il *monitoring* di un sistema, si devono rispettare due regole principali:

- I *probes* devono essere in grado di raccogliere il giusto numero di informazioni circa il comportamento del sistema;
- Il comportamento del *target system* non deve essere alterato in maniera significativa a causa dell'inserimento dei *probes*. Quando questo avviene, si dice che il sistema di monitoraggio è *poco intrusivo*.

1.3 FAULT INJECTION

La *fault injection* è un approccio all'analisi della *dependability* che estende le tecniche di *monitoring* esposte in 1.2.

È definita come la volontaria introduzione di guasti all'interno di un sistema, al fine di osservarne il comportamento in presenza di guasti. [7] [8] [9]

Gli strumenti utilizzati per effettuare un'attività di *fault injection* sono mostrati in figura 7.

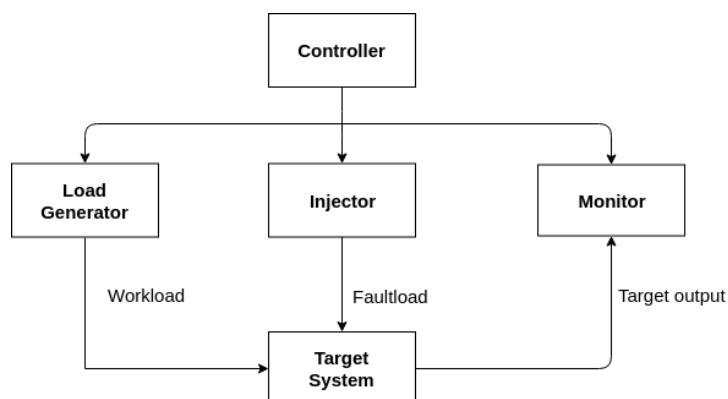


Figura 7: Elementi di un processo *fault injection*

Il *target system* è, come nel *monitoring*, il sistema che si vuole osservare. Il sistema viene alimentato con un *workload* generato dal *load generator*.

L'input fornito al sistema è costituito da dati che vengono normalmente processati dal sistema a *runtime*.

L'*injector* introduce guasti all'interno del sistema durante l'esperimento, i quali alterano la struttura o lo stato del sistema. Il carico di guasti iniettati prende il nome di *faultload*.

Il *monitor* colleziona i dati prodotti dal sistema per compiere misure di *dependability*, in maniera non diversa da quanto avviene nel classico *monitoring*.

Il *controller* coordina le operazioni degli altri componenti e itera gli esperimenti.

Da un punto di *dependability means*, la *fault injection* è una tecnica di *fault removal* e di *fault forecasting* in quanto viene testata l'efficacia dei meccanismi di *fault tolerance* del sistema e stimata la *dependability* osservando il comportamento dello stesso in presenza di guasti.

Le caratteristiche dei guasti iniettati, in termini di *tipo* e *frequenza*, vengono chiamate *fault model*.

Il *fault model* di un sistema è ottenuto considerando:

- I requisiti di sistema;
- L'ambiente in cui esso opera;
- Le tecnologie con cui il sistema è realizzato.

Per essere affidabile, un'attività di *fault injection* deve rispettare alcune proprietà fondamentali:

- Rappresentatività: sia il *workload* che il *faultload* devono corrispondere ai profili di utilizzo del sistema e alle modalità di fallimento durante la fase operazionale;
- Non intrusività: come nel caso del *monitoring*, l'intrusività data dall'iniezione di guasti e dall'osservazione del sistema deve essere minimizzata;
- Fattibilità: le tecniche di iniezione dei guasti devono richiedere un minimo sforzo implementativo, al fine di limitare i costi;
- Ripetibilità: Una campagna di *fault injection* deve produrre risultati comparabili se l'esperimento è ripetuto diverse volte sotto le stesse condizioni, per aumentare l'affidabilità dei risultati ottenuti.

Le attività di *monitoring* e *fault injection* fanno spesso affidamento sull'utilizzo di *tool* specifici. [10]

1.4 SCOPO DELLA TESI

Questa Tesi descrive l'attività di *monitoring* e *fault injection* effettuata sul prototipo di un sistema informatico impiegato nell'ambito del posizionamento ferrotramviario. Per quanto esposto in 1.1, il dominio ferrotramviario è un contesto *safety-critical*, come tale, lo sviluppo di sistemi informatici da impiegare nell'ambito ferrotramviario è regolamentato da specifici standard e requisiti di *dependability*.

Si introducono i sistemi ferrotramviari come derivazione dai classici sistemi ferroviari, si descrive il problema del posizionamento e le tecniche attualmente utilizzate per risolverlo, inquadrando il contesto operativo in cui si colloca il sistema analizzato.

2

STATO DELL'ARTE

2.1 SISTEMI FERROVIARI E FERROTAMVIARI

È possibile schematizzare un sistema ferroviario, o ferrotramviario, come un insieme di vetture vincolate a muoversi lungo una traccia fissata. Questa schematizzazione è approssimativamente 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, anche dette *rotabili*, 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 8.



Figura 8: Schema di un tipico scenario ferrotramviario

2.1.1 *La safety nei sistemi ferroviari*

La gestione della *safety* nei sistemi ferroviari è regolamentata dallo standard EN 50126. [11]

Questo standard regolamenta la gestione della *safety* nel dominio ferroviario, con particolare enfasi verso le attività di valutazione RAMS (*Reliability, Availability, Maintainability, Safety*).

Esso prevede che vengano redatti i seguenti documenti.

SAFETY PLAN

Un insieme documentato di attività programmate, risorse ed eventi necessari a implementare la struttura organizzativa, le responsabilità, le procedure, le attività, le funzioni e le risorse che insieme assicurano la *safety* del sistema.

In sintesi, il *safety plan* individua *chi fa cosa, e quando*, al fine di realizzare un sistema *safe*.

SAFETY REQUIREMENTS

Definisce i requisiti di *safety* che il sistema deve soddisfare prima di poter essere impiegato sul campo.

SAFETY CASE

La dimostrazione documentata che un prodotto sia conforme ai suoi requisiti di *safety*. Il *Safety Case* deve essere redatto come una dimostrazione

logica della *safety* del sistema.

Per quanto riguarda le procedure e i requisiti tecnici per lo sviluppo di applicazioni software utilizzate nel controllo ferroviario, lo standard di riferimento è EN 50128. [12]

Lo standard EN 50128 impone che a ciascun software integrato all'interno di un sistema ferroviario debba essere assegnato un *Safety Integrity Level* (SIL). [13]

Il SIL viene assegnato sulla base del tempo medio al fallimento catastrofico, ossia il MTTF relativo a fallimenti catastrofici.

Livello SIL	Tempo medio al fallimento catastrofico
SIL-1	$[10^5 - 10^6]$ ore
SIL-2	$[10^6 - 10^7]$ ore
SIL-3	$[10^7 - 10^8]$ ore
SIL-4	$\geq 10^8$ ore

Tabella 1: Livelli SIL

Mentre lo standard EN 50126 è generico e applicabile a qualunque sistema ferroviario o parti di esso, poiché relativo alla *gestione* della *safety* piuttosto che al suo effettivo raggiungimento, lo standard EN 50128 è inerentemente tecnico e specifico per i software critici. Esso identifica le fasi principali nel ciclo di vita del software e per ciascuna di queste prevede un insieme di tecniche da utilizzare per soddisfare il SIL assegnato al software.

2.2 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. [14]

2.2.1 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.[15] 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) [16], 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.

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 sistema 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. [17]

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*;^[18]
- ETCS - 3: Posizionamento autonomo, nessun utilizzo di apparati a terra.

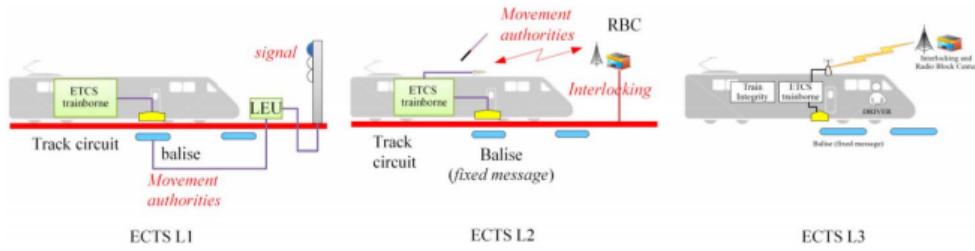


Figura 9: Livelli ETCS

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.[19]

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*.

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.

2.2.2 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. [20]

Il sistema analizzato in questa Tesi è un sottosistema di posizionamento conforme alla filosofia ETCS-3. Nell'ottica di migrazione verso sistemi ETCS-3, è stato progettato un sistema di posizionamento ferrotramviario autonomo, basato principalmente sull'utilizzo di un *sensore inerziale*, un odometro e un GPS installati a bordo treno, le cui misurazioni vengono processate da un software al fine di stimare la posizione del treno. [21] Le misure fornite al software vengono integrate attraverso l'utilizzo di

un algoritmo noto come *Sensor Fusion Algorithm* (SFA). [23]

Attualmente il software ha superato le fasi di analisi e definizione dei requisiti, *system design*, e si trova nella fase di implementazione. Si è quindi reso disponibile per l'analisi di *dependability* un prototipo del software. Per quanto esposto in 1.1, questo può essere osservato durante l'esecuzione nel suo ambiente operativo.

L'analisi condotta sul sistema è del tipo *fault injection*. Poichè esso opererà in un contesto *safety critical*, il focus è quello di valutare l'efficacia dei meccanismi di *fault tolerance* implementati nel sistema.

Il *target system* è l'intero sottosistema di posizionamento installato a bordo, quindi include gli strumenti di misura e l'*On Board Control Unit* (OBCU). OBCU è un computer di bordo che dovrà elaborare le informazioni sulla posizione del treno al fine di interagire, quando necessario, con il sistema di *interlocking* della traccia.

Il *target component* in esame è il modulo software che processa le misure campionate. Durante il processo di analisi, si farà affidamento a un *tool* realizzato appositamente per valutare le performance del software che implementa SFA. Il *tool* utilizzato include:

- Un *load generator* che alimenta SFA con misure che verosimilmente potrebbero essere campionate dal sistema nel suo ambiente reale;
- Un *injector* in grado di iniettare i guasti che potrebbero verificarsi quando il sistema sarà impiegato sul campo;
- Un *monitor* che colleziona le uscite e i risultati intermedi del modulo SFA.

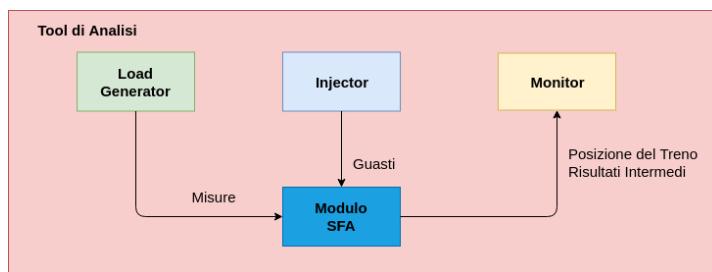


Figura 10: Schema dell'ambiente di *fault injection*

Il *tool* si comporta come *controller* in quanto è in grado di controllare il *load generator*, l'*injector* e il *monitor*. Esso itera gli esperimenti un numero arbitrario di volte, stabilito dall'osservatore, e fornisce un report riassuntivo della campagna sperimentale. Il modulo SFA è incluso nel *tool* come libreria.

2.3 CYBER PHYSICAL SYSTEMS OF SYSTEMS

Il sottosistema di posizionamento studiato rientra nella categoria dei *Cyber Physical Systems of Systems* (CPSoS). [22]

Un *Cyber Physical System* (CPS) è un sistema composto da due parti:

- *Cyber*: Elementi di computazione, comunicazione e controllo che operano in un tempo discreto;
- *Physical*: Sottosistema naturale o costruito dall'uomo che è governato dalle leggi della fisica e opera in un tempo continuo.

Un CPS è un sistema che consiste in un computer (il sistema *cyber*) e un oggetto controllato (il sistema *physical*), che include la possibilità di interazione con gli esseri umani.

In talune applicazioni i sistemi sono integrati in un più ampio contesto al fine di fornire un *servizio* che un sistema *standalone* non sarebbe in grado di fornire. Si parla in questo caso di *Systems of Systems* (SoS).

I SoS sono caratterizzati da una gerarchia multi-livello: una struttura ricorsiva in cui un sistema viene considerato *il tutto* al più alto livello di interesse (*macro-level*), e composto da un insieme di sottosistemi al livello sottostante (*micro-level*). Ciascun sottosistema può essere considerato a sua volta come un sistema, e la ricorsione termina quando l'ulteriore suddivisione di un sottosistema perde di interesse.

Si definisce *componente* un sottosistema al più basso livello di interesse.

Un SoS composto da CPS prende il nome di CPSoS.

Gli elementi architetturali che compongono un CPSoS sono:

- *Itom*: costituiscono l'unità minima di interazione. Sono elementi informativi composti da dati e metadati;
- *Constituent Systems* (CS): Entità di tipo CPS che processano e scambiano gli *itom*;
- Ambiente: tutte le entità con cui può interagire un CS.

I CS interagiscono fra di loro, o con l'ambiente, scambiandosi *itom* attraverso le loro *interfacce*.

Si distinguono le seguenti interfacce:

- *Local I/O* (L-interface): interfacce che abilitano l'interazione tra un CS e il suo ambiente. Non sono rilevanti ai fini del servizio che si intende fornire;

- *Configuration* (C-interface): interfacce utilizzate per scopi di configurazione o *update* hardware e software;
- *Diagnostic* (D-Interface): Interfacce di diagnostica utili per il *monitoring* di un CS;
- *Time-Sync* (TSI): interfacce di sincronizzazione esterna che vengono utilizzate per rendere un CS *time-aware* e stabilire una *global timebase*;
- *Relied Upon Interfaces* (RUI): Un CPSoS fa affidamento alle RUI per fornire il servizio atteso.

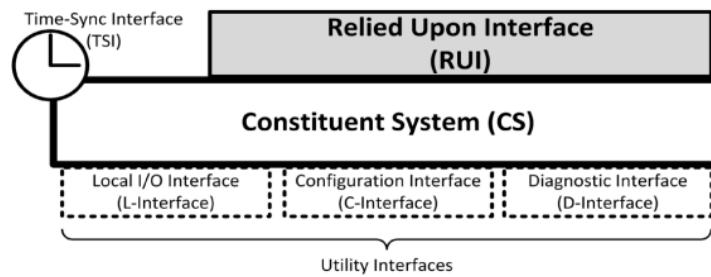


Figura 11: Interfacce di un CPS

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. [32]

In base al canale di comunicazione e agli *item* scambiati attraverso le RUI, si distinguono:

- *Relied Upon Message Interfaces* (RUMI): L'interazione consiste di uno scambio di messaggi a livello *cyber*, come ad esempio un pacchetto TCP;
 - *Relied Upon Physical Interfaces* (RUPI): Le componenti *physical* di un CS interagiscono attraverso un canale *stigmergico*.
- Un CS altera una grandezza fisica nell'ambiente, in maniera tale per cui un altro CS possa trarre contenuto informativo attraverso il rilevamento della modifica.

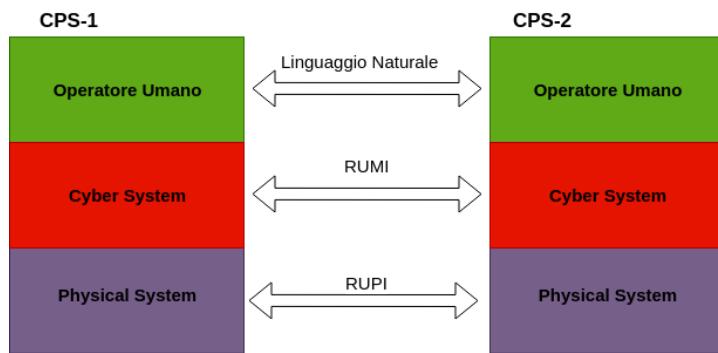


Figura 12: RUMI e RUPI

Un CPS crea canali stigmergici attraverso l'uso di *sensori* e *attuatori*. I sensori sono in grado di rilevare parte delle grandezze fisiche che caratterizzano l'ambiente operativo di un CS, mentre gli attuatori sono in grado di alterare tali grandezze.

3

IL SISTEMA ANALIZZATO

3.1 DESCRIZIONE GENERALE

Il sistema analizzato rientra nella categoria dei CPSoS. 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.[24]

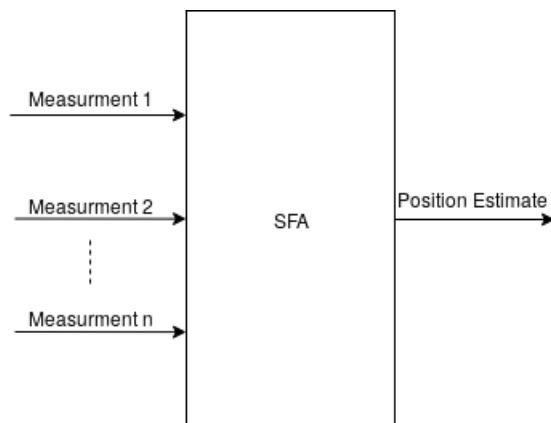


Figura 13: Schema SFA

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.[25][26][27]

Queste informazioni si suppongono note a priori ed accedibili tramite un *database* caricato in memoria centrale. [28]

3.2 CONSTITUENT SYSTEMS

Il sottosistema di posizionamento si compone dei moduli, o CS, descritti nel seguito di questa sezione.

3.2.1 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}_{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. [29] [30]

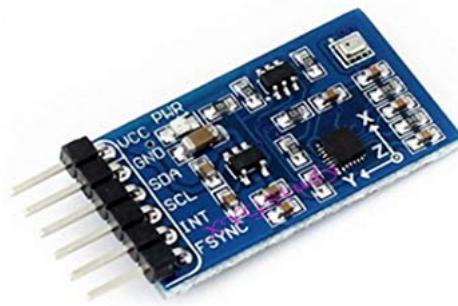


Figura 14: *Inertial Measurement Unit*

- **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 15: Ricevitore GPS ublox EVK-M8T

3.2.2 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 16: 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. [31]

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

Tabella 2: Specifiche Tecniche NVidia TX-Jetson

3.2.3 OBCU

L'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.

3.3 SPECIFICA DELLE INTERFACCE

3.3.1 RUI

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

Il sistema di posizionamento interagisce con il treno attraverso le RUPI del *Sensor Set*, ossia gli strumenti di misura che esso integra. Questi sensori campionano, 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 3.

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

Tabella 3: 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 UDP contenenti le informazioni sulla progressiva chilometrica stimata da SFA, ed eventualmente messaggi di *acknowledgment* di OBCU verso la scheda.



Figura 17: Modem TP-LINK M7350 LTE-4G

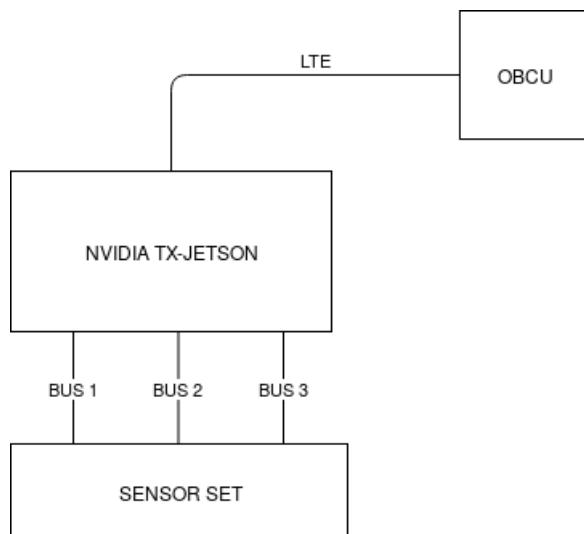


Figura 18: Diagramma a blocchi del sottosistema di posizionamento

3.4 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.

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 4: Specifica delle RUMI del sistema

3.4.1 Acquisizione dei dati

L'acquisizione dei dati si divide in due differenti interazioni: la prima, con la vettura, 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 19 viene riportato un *sequence diagram* rappresentante una possibile sequenza di campionamento e invio dei dati. Questa tipologia di

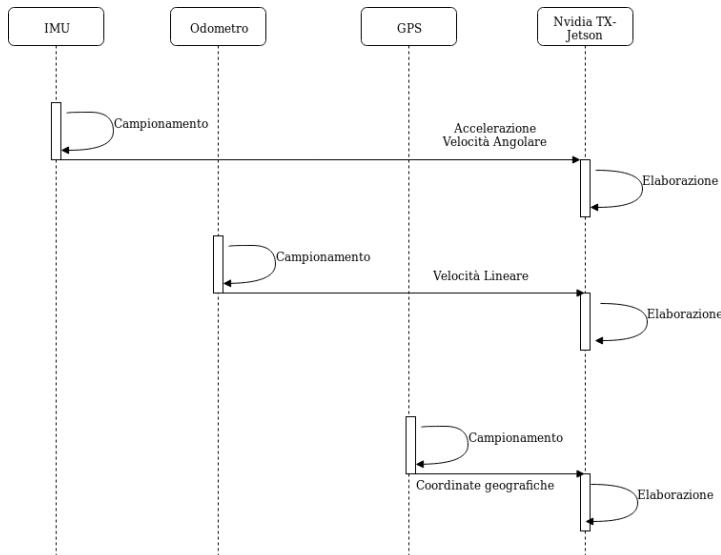


Figura 19: Sequenza di acquisizione dati

interazione è detta *time-triggered*, in quanto è determinata unicamente dallo scorrere del tempo. [34]

3.4.2 Trasmissione della posizione

Ogniqualvolta viene 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* [35] 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.

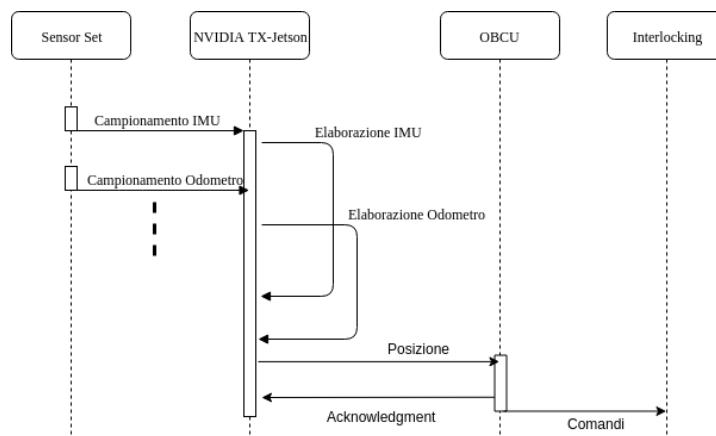


Figura 20: Sequenza di trasmissione della posizione

3.4.3 Interazioni con eventuali operatori umani

La filosofia del sistema 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 *demon* 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*.

3.5 ARCHITETTURA SOFTWARE

In questa sezione viene completata la panoramica sul sistema descrivendo i moduli software in esecuzione sulla piattaforma NVidia TX-Jetson. Su tale piattaforma è installato il sistema operativo Ubuntu 16.04 LTS, basato su Kernel Linux.

3.5.1 *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.

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; [33]

- `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.

3.5.2 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.

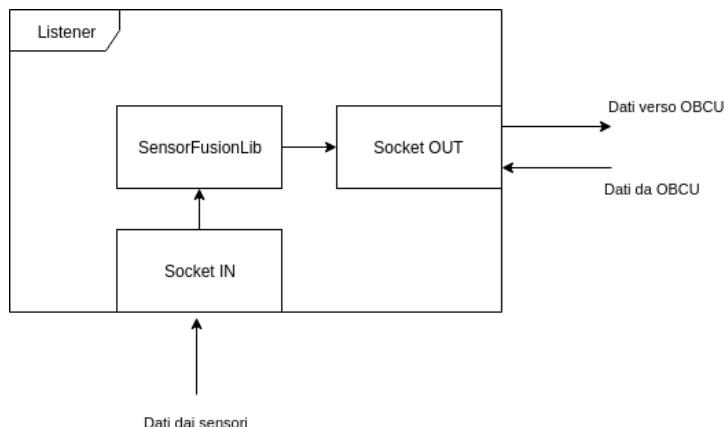


Figura 21: Diagramma a blocchi di *listener*

3.5.3 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*.

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

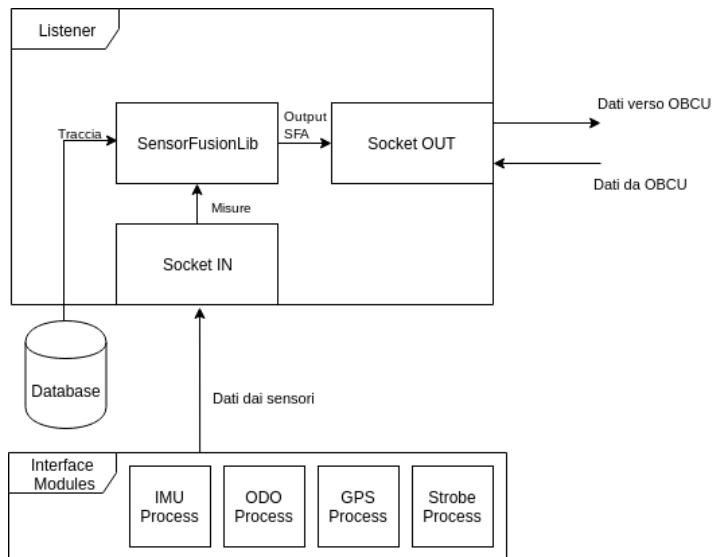
Tabella 5: Moduli di *interface-modules*

Figura 22: Diagramma a blocchi del sistema software

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. [10]

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* [39] 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* [38], 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*.

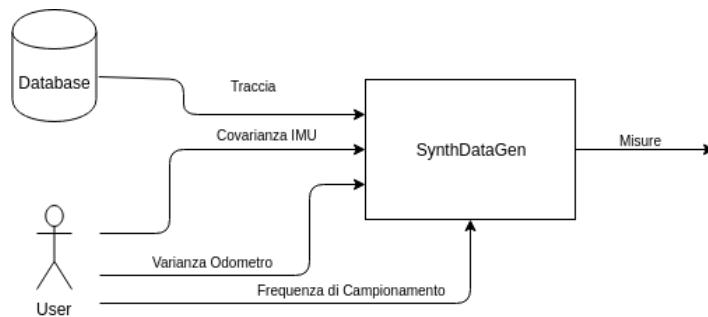


Figura 23: Schema di *SynthDataGen*

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.

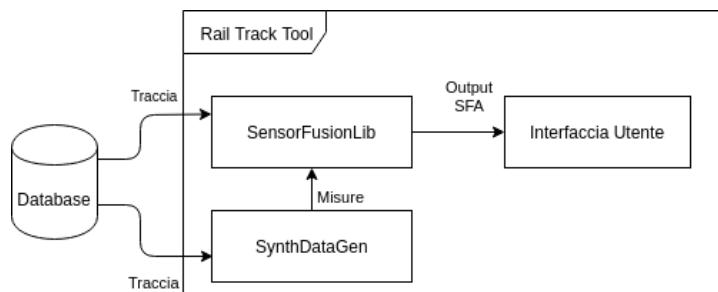


Figura 24: Schema riassuntivo RTT

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;

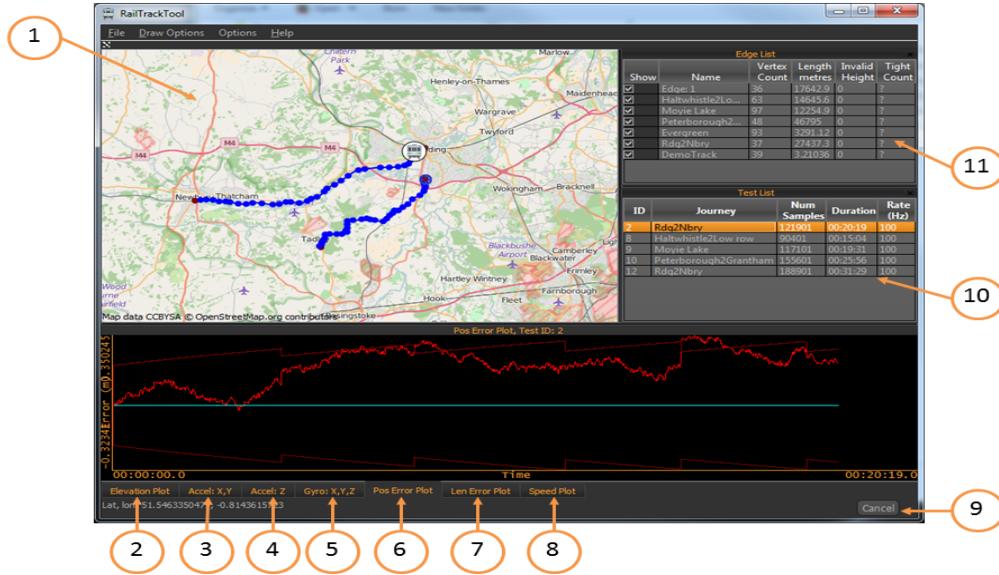


Figura 25: Interfaccia RTT

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 26).

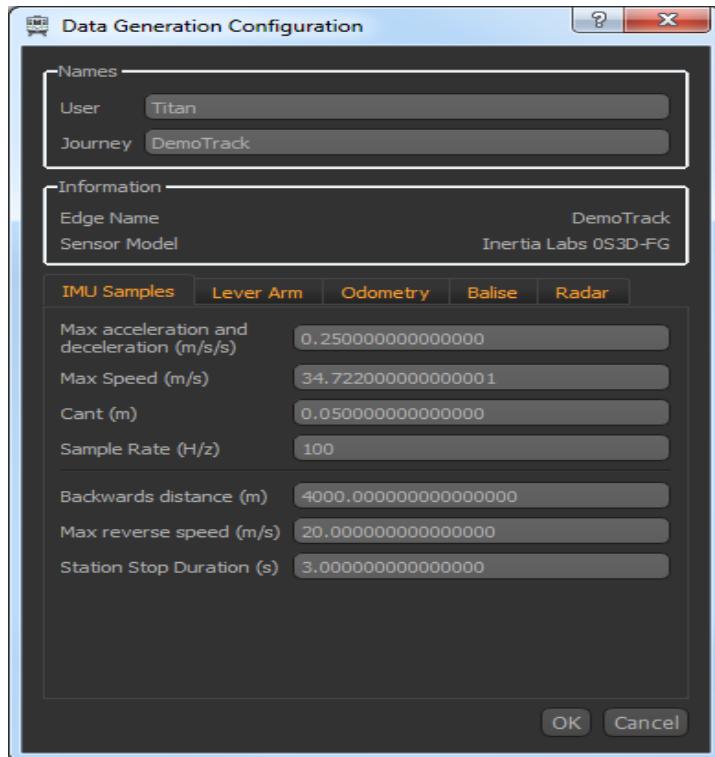


Figura 26: Pannello di configurazione generale *SynthDataGen*

Un'altra interfaccia permette infine di definire le caratteristiche dei sensori simulati, come il rumore del processo di misura (figura 27).

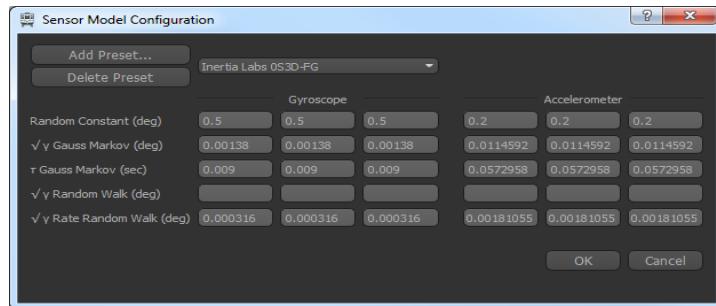


Figura 27: Pannello di configurazione IMU simulato

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.

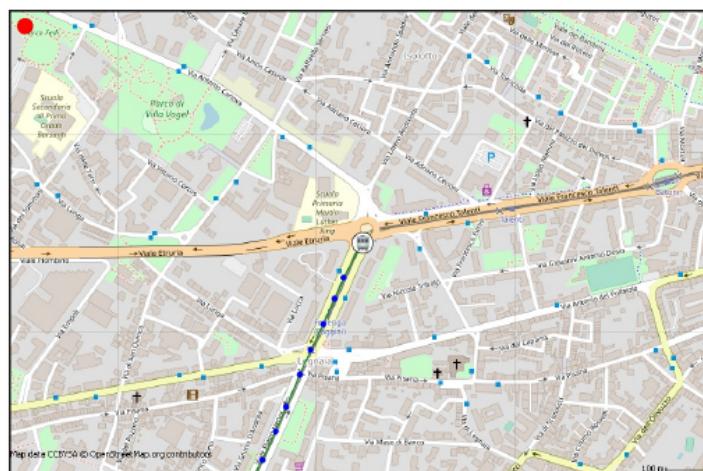


Figura 28: Posizione del treno mostrata sulla mappa

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.

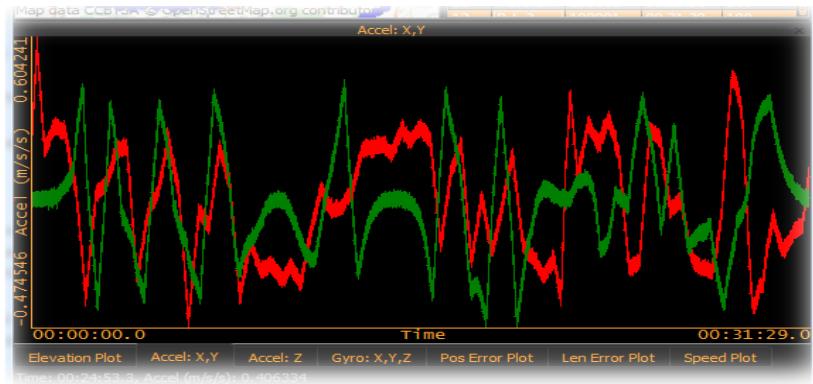


Figura 29: Grafico valori di accelerazione assi X e Y



Figura 30: 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 31: 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 6: 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 7: Esperimento 1.1: Risultati

Esperimento 1.2

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

Tabella 8: 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 9: 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 10: 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 11: 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 12: 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 13: 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 14: 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 15: 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 16: 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 17: 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 18: 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 19: 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 20: 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 21: 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 22: 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 23: 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 24: 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] J.C. Knight, *Safety Critical Systems: Challenges and Directions, Proceedings of the 24th International Conference on Software Engineering* (2002) (Cited on page 11.)
- [3] 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.)
- [4] G.J. Nutt, *Tutorial: Computer system monitors*, Computer (1975) (Cited on page 13.)
- [5] B. Plattner, *Real-time execution monitoring*. *IEEE Transactions on Software Engineering* (1984) (Cited on page 13.)
- [6] B. Plattner, J. Nievergelt, *Special feature: Monitoring program execution: A survey*, Computer (1981) (Cited on page 13.)
- [7] M. Vieira, *Fault Injection and Robustness Testing - Enabling Techniques for Assessing Computer Systems*, University of Coimbra, Portugal (Cited on page 15.)
- [8] M. Vieira, *Assessing the robustness and security of Web Services*, AMBER, University of Coimbra, Portugal (Cited on page 15.)
- [9] H. Madeira, *Design for experiments for resilience assessment and benchmarking*, University of Coimbra, Portugal (Cited on page 15.)
- [10] K. Wolter et al, *Resilience Assessment and Evaluation of Computing Systems*, Springer (2012) (Cited on pages 16 and 41.)
- [11] CENELEC European Committee for Electrotechnical Standardization, *EN 50126:2000 - Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)* (2000) (Cited on page 20.)

- [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 21.)
- [13] International Electrotechnical Commission, *61508-1: Functional safety of electrical/electronic/programmable electronic safety-related systems, edition 2.0* (2010) (Cited on page 21.)
- [14] J. Otegui, *A Survey of Train Positioning Solutions*, *IEEE Sensors Journal*, Vol. 17, No. 20 (2017) (Cited on page 21.)
- [15] 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 22.)
- [16] European Commission, *Delivering an effective and interoperable European Rail Traffic Management System (ERTMS) - the way ahead* (2017) (Cited on page 22.)
- [17] 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 22.)
- [18] P. Josserand, F.H Willard, *Rights of Trains* (5th ed.), Simmons-Boardman Publishing Corporation, New York (1957) (Cited on page 22.)
- [19] 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 23.)
- [20] 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) (Cited on page 23.)
- [21] 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 23.)
- [22] A. Mirabadi et al, *Application of sensor fusion to railway systems*, *IEEE* (1996) (Cited on page 24.)
- [23] A. Ceccarelli et al, *Basic Concepts on Systems of Systems, Cyber-Physical Systems of Systems*, Springer (2017) (Cited on page 25.)

- [24] 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 29.)
- [25] 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 30.)
- [26] 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 30.)
- [27] 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 30.)
- [28] L. Junyan et al, *Application Research of Embedded Database SQLite*, IEEE (2009) (Cited on page 30.)
- [29] X. Liu, A. Goldsmith, *Kalman Filtering with Partial Observation Losses*, Department of Electrical Engineering, Stanford University, Stanfond, USA (Cited on page 31.)
- [30] 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 31.)
- [31] 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 32.)
- [32] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications 2nd edn*, Springer, New York (2011) (Cited on page 26.)
- [33] H. Kopetz, W. Ochsenreiter, *Clock synchronization in distributed real-time systems*, IEEE (1987) (Cited on page 37.)
- [34] M.J. Pont, *Patterns for Time-Triggered Embedded Systems*, Addison-Wesley (2001) (Cited on page 35.)
- [35] H. Kopetz, *Event-Triggered versus Time-Triggered Real-Time Systems*, Springer (1991) (Cited on page 36.)

- [36] ISO/IEC 7498-1:1994, *Information technology, Open Systems Interconnection - Basic Reference Model: The Basic Model* (1994)
- [37] D.E. Kalman, *Estimation, Control, and the Discrete Kalman Filter, Applied Mathematical Science, Springer* (1987)
- [38] S. Dilhaire, D. Maillet, *Dealing with the measurement noise of a sensor* (2015) (Cited on page 42.)
- [39] 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 41.)