





**ABSTRACT**



---

## INDICE

---

|       |                              |    |
|-------|------------------------------|----|
| 1     | Stato dell'Arte              | 9  |
| 2     | Architettura di Sistema      | 11 |
| 2.1   | Descrizione generale         | 11 |
| 2.2   | Constituent Systems          | 12 |
| 2.3   | Specifiche delle Interfacce  | 14 |
| 2.3.1 | Relied Upon Interfaces       | 14 |
| 2.3.2 | Altre Interfacce             | 16 |
| 2.4   | Interazioni                  | 16 |
| 2.4.1 | Acquisizione dei dati        | 17 |
| 2.4.2 | Trasmissione della posizione | 17 |
| 3     | Specifiche Software          | 19 |
| 3.1   | Architettura Software        | 19 |
| 3.2   | Protocolli di Comunicazione  | 20 |
| 3.2.1 | Trasmissione in entrata      | 20 |
| 3.2.2 | Trasmissione in uscita       | 22 |
| 3.3   | Scenario di Esempio          | 23 |
| 4     | Ambiente di Analisi          | 27 |
| 4.1   | Architettura Hardware        | 27 |
| 5     | Esperimenti e Risultati      | 29 |
| 6     | Conclusioni                  | 31 |



---

## ELENCO DELLE TABELLE

---

|           |  |    |
|-----------|--|----|
| Tabella 1 | Specifiche delle RUPI del sistema      | 15 |
| Tabella 2 | Protocollo di comunicazione in entrata | 21 |
| Tabella 3 | Significato del campo SENSOR_TYPE      | 21 |
| Tabella 4 | Protocollo di comunicazione in uscita  | 22 |
| Tabella 5 | Formato del pacchetto di <i>ack</i>    | 23 |
| Tabella 6 | Condizioni iniziali                    | 23 |



---

## ELENCO DELLE FIGURE

---

|          |                                   |    |
|----------|-----------------------------------|----|
| Figura 1 | Schema SFA                        | 11 |
| Figura 2 | <i>Inertial Measurment Unit</i>   | 13 |
| Figura 3 | Ricevitore GPS ublox EVK-M8T      | 13 |
| Figura 4 | Nvidia TX-Jetson                  | 14 |
| Figura 5 | Modem TP-LINK M7350 LTE-4G        | 15 |
| Figura 6 | RUMI                              | 16 |
| Figura 7 | Sequenza di acquisizione dati     | 17 |
| Figura 8 | Architettura software bordo treno | 20 |
| Figura 9 | Modifiche architettura hardware   | 28 |



# 1

---

## STATO DELL'ARTE

---



# 2

---

## ARCHITETTURA DI SISTEMA

---

In questo capitolo viene descritta l'architettura *hardware* del CPS in esame, in particolare, ne vengono evidenziati i *Constituent System* (CS) e le loro interfacce di comunicazione. Vengono infine descritte le interazioni alle particolari interfacce del sistema.

### 2.1 DESCRIZIONE GENERALE

Lo scopo del sistema è quello di implementare un meccanismo di posizionamento basato su SFA.<sup>[1]</sup>

Il software che esegue tale algoritmo è schematizzabile come una *black-box* (figura 1), la quale prende in ingresso un certo insieme di misure, e fornisce in uscita una stima affidabile della posizione del treno lungo la traccia.

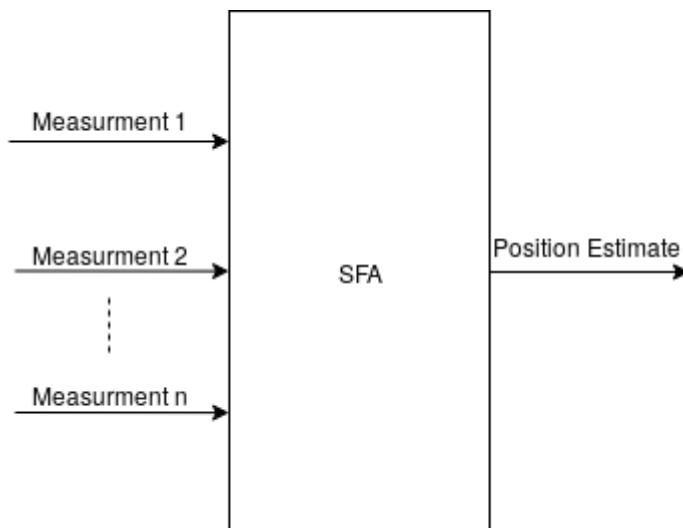


Figura 1: Schema SFA

Questo algoritmo verrà eseguito su di un hardware installato a bordo treno, ed ha lo scopo di monitorare costantemente il moto dello stesso. Le grandezze fisiche che dovranno essere misurate e fornite a SFA sono:

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

SFA utilizzerà queste informazioni in combinazione con un'apposita digitalizzazione della traccia tramviaria su cui si trova il treno monitorato. Queste informazioni si suppongono note a priori ed accedibili tramite un *database* caricato in memoria centrale.

## 2.2 CONSTITUENT SYSTEMS

Il sistema studiato si compone dei seguenti CS:

- *Sensor Set*, ossia un insieme di sensori atto a campionare le misure di interesse per il sistema. Il *Sensor Set* è composto dai seguenti moduli:
  - *Inertial Measurement Unit* (IMU):  
Unità incaricata di trasmettere al sistema i vettori **accelerazione** (**a**) e **velocità angolare** (**v<sub>ang</sub>**). Le misure di IMU sono prese rispetto alla Terra e sono espresse in unità stabilite dallo standard internazionale (SI):

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

Esso è il sensore principale. Date le caratteristiche intrinseche del particolare SFA utilizzato, ossia un *Filtro di Kalman*, il sistema potrebbe funzionare anche senza i rimanenti sensori. Si osserverebbe tuttavia un calo delle performance in termini di errore commesso sulla stima della posizione del treno. [2]

- **Odometro**:  
Unità incaricata di fornire al sistema i campionamenti dei valori di velocità lineare del treno, espressi in  $\frac{\text{m}}{\text{s}}$ .

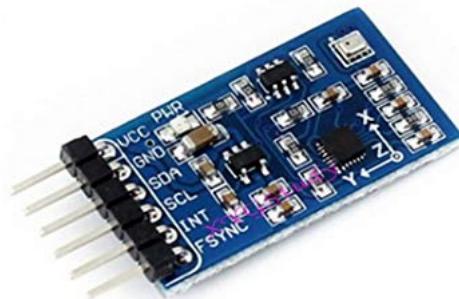


Figura 2: *Inertial Measurement Unit*

- GPS:

Unità che fornisce al sistema 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 3: Ricevitore GPS ublox EVK-M8T

- Piattaforma di elaborazione dati. Consiste di una scheda Nvidia TX-Jetson su cui viene eseguito SFA.



Figura 4: Nvidia TX-Jetson

- *On Board Control Unit* (OBCU). 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. [3].

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 dal resto del sistema di posizionamento (tabella 1).

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

| 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 1: Specifica delle RUPI del sistema

- 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 5: Modem TP-LINK M7350 LTE-4G

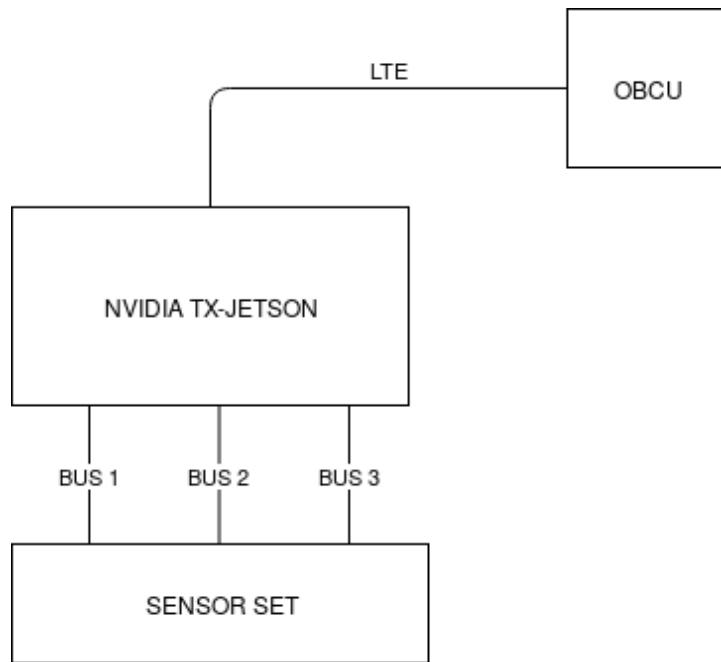


Figura 6: RUMI

### 2.3.2 Altre Interfacce

Oltre alle RUI, descritte in 2.3.1, esistono altre interfacce che hanno lo scopo di rendere il sistema osservabile e manutenibile, e sono le seguenti: [4]

- *Time Synchronization Interfaces* (TSI). Le TSI permettono al CPS di effettuare una sincronizzazione col tempo fisico al fine di stabilire una *global timebase* [5].
- *Utility Interfaces* (UI). Interfacce dei CS che ne consentono la configurazione, il controllo, e l'osservazione non intrusiva del suo comportamento [6].

Come verrà approfondito nel successivo capitolo, sia le TSI che le UI sono nella fattispecie interfacce *software*.

## 2.4 INTERAZIONI

In questa sezione vengono descritte le interazioni osservabili alle interfacce del sistema.

### 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 di tipo 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 7 viene riportata una sequenza esempio di campionamento e

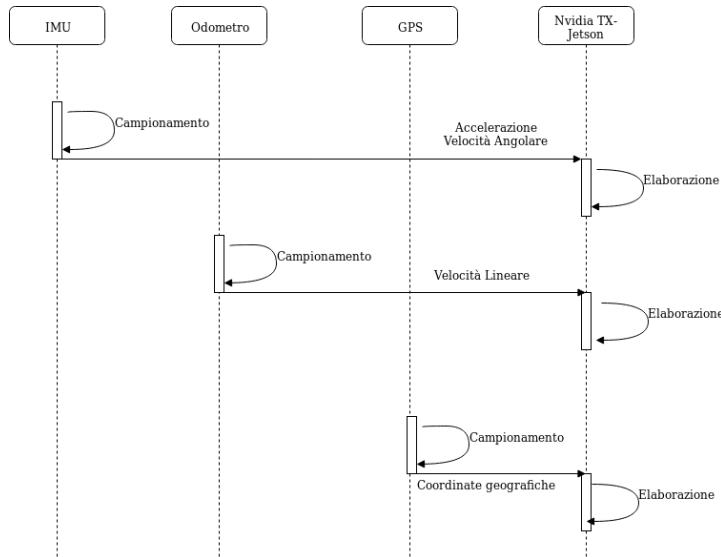


Figura 7: Sequenza di acquisizione dati

invio dei dati. I protocolli di comunicazione tra *Sensor Set* e la piattaforma di elaborazione dati sono definiti a livello *software*, e saranno descritti nel prossimo capitolo. Questa tipologia di interazione è detta *time-triggered*, in quanto è determinata unicamente dallo scorrere del tempo. [7] [8]

### 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 un'aggiornamento di SFA viene completato, avviene 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* [8] in quanto le tempistiche di interazione non sono note a priori, ma dipendono dal tempo impiegato da SFA a compiere un'iterazione per aggiornare la stima prodotta.

LTE è a tutti gli effetti una regolare interfaccia di rete. A livello di trasporto, il messaggio trasmesso è contenuto nel *payload* di un datagramma UDP; in accordo al modello di rete ISO-OSI. [9] La specifica del messaggio a livello applicazione sarà descritta nel prossimo capitolo.

# 3

---

## SPECIFICHE SOFTWARE

---

In questo capitolo vengono descritte le specifiche software del sistema di posizionamento SFA. Si evidenziano le caratteristiche architetturali dei software che concorrono al raggiungimento dello scopo del sistema, e se ne descrivono i protocolli di comunicazione.

### 3.1 ARCHITETTURA SOFTWARE

Sulla piattaforma di elaborazione dati è installato il sistema operativo Ubuntu 16.04 LTS, basato su kernel Linux.

Su tale piattaforma viene eseguito il modulo SFA, opportunamente encapsulato in un eseguibile, denominato *listener*.

Un set di moduli software, denominati *interface-modules*, sono in esecuzione sulla scheda. Lo scopo di questo insieme di programmi è quello di funzionare come interfaccia interna verso i sensori del CPS. La comunicazione fra *interface-modules* e *listener* avviene attraverso un protocollo applicazione stabilito arbitrariamente, sia esso INPUT\_PROTOCOL, mentre a livello di trasporto si utilizza UDP.

I valori ricevuti da *listener* vengono salvati in apposite *strutture dati* rappresentanti misure della stessa sorgente:

- I vettori accelerazione e velocità angolare rilevati da IMU vengono convertiti nella struttura dati IMU\_POD;
- La velocità rilevata dal Radar/Odometro viene convertita nella struttura dati ODO\_POD;
- La posizione rilevata dal GPS viene infine convertita nella struttura dati GPS\_POD.

Il software *listener* è in grado di inviare le misurazioni ricevute da *interface-modules* a SFA, quali variabili di tipo IMU\_POD, ODO\_POD,

GPS\_POD ed altresí di ricevere la stima della posizione del treno, essendo questo l'output dell'algoritmo, quale variabile di tipo SFA\_OUTPUT\_POD. Ogniqualvolta listener riceva un' uscita da SFA, si fa carico della comunicazione tra scheda e OBCU, utilizzando un protocollo arbitrario a livello applicazione, sia esso OUTPUT\_PROTOCOL.

Uno schema dell'architettura software è quello mostrato in figura 8.

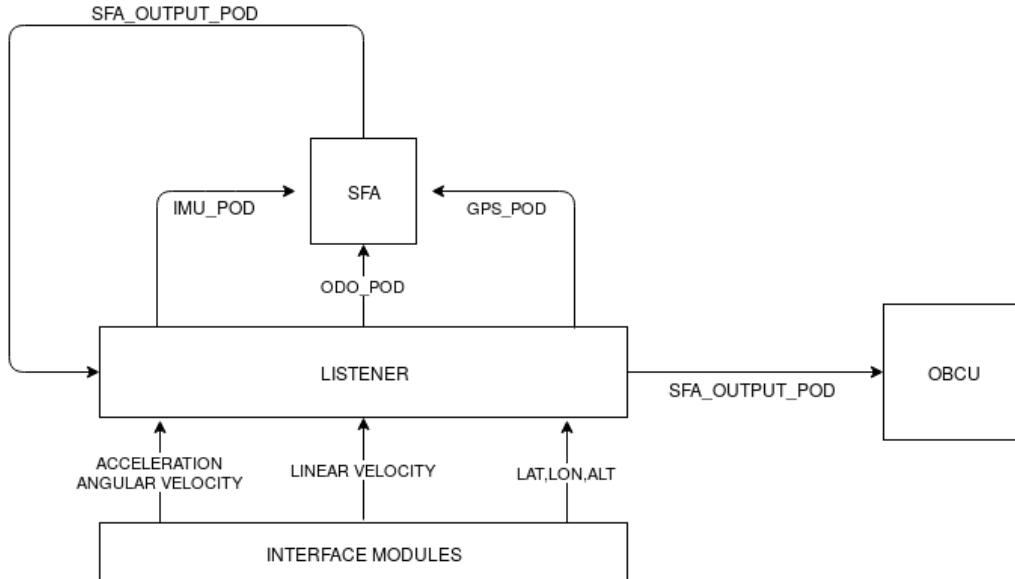


Figura 8: Architettura software bordo treno

### 3.2 PROTOCOLLI DI COMUNICAZIONE

Nella precedente sezione sono stati brevemente introdotti i protocolli di comunicazione implementati per gestire la comunicazione UDP:

- In entrata, tra interface-modules e listener (INPUT\_PROTOCOL);
- In uscita, tra listener e OBCU (OUTPUT\_PROTOCOL).

#### 3.2.1 Trasmissione in entrata

Per trasmettere i dati da interface-modules a listener, e dunque dai sensori al modulo software che implementa SFA, è stato realizzato un protocollo di comunicazione denominato INPUT\_PROTOCOL.

Tale protocollo fa affidamento a livello trasporto su UDP per massimizzare

la velocità di trasmissione.

Il protocollo definisce il formato del *payload* del pacchetto UDP che contiene le informazioni di IMU, Radar/Odometro, o GPS, ed è descritto in tabella 2.

A discrezione del valore del campo SENSOR\_TYPE si distingue il tipo di

| Campo       | Descrizione                | Indici di bit | Tipo     |
|-------------|----------------------------|---------------|----------|
| SENSOR_TYPE | ID Sensore Sorgente        | 0-7           | uint8_t  |
| Seq.N0      | Numero di sequenza         | 8-23          | uint16_t |
| N_INT       | Numero di interi trasmessi | 24-31         | uint8_t  |
| N_DOUBLE    | Numero di double trasmessi | 31-38         | uint8_t  |

Tabella 2: Protocollo di comunicazione in entrata

informazione trasportata dal pacchetto, come descritto in tabella 3.

I pacchetti GROUND TRUTH sono pacchetti di inizializzazione dell'algo-

| Valore di SENSOR_TYPE | Sorgente del pacchetto |
|-----------------------|------------------------|
| 1                     | IMU                    |
| 2                     | ODOMETRO               |
| 3                     | GPS                    |
| 8                     | GROUND TRUTH           |
| 9                     | STROBE                 |
| 10                    | STOP                   |

Tabella 3: Significato del campo SENSOR\_TYPE

ritmo: alla ricezione del pacchetto GROUND TRUTH l'algoritmo si avvia leggendo i valori trasmessi in coda al pacchetto, in accordo al valore dei campi N\_INT e N\_DOUBLE. Tali valori forniscono informazioni come progressiva chilometrica e velocità iniziali del treno.

I pacchetti STROBE sono inviati ogni secondo e forniscono un solo valore double, ossia un *timestamp* che l'algoritmo utilizza per sincronizzarsi. Per quanto esposto nel Capitolo 2, l'interfaccia UDP attraverso cui comunicano interface-modules e listener funge da TSI per il sistema.

Il pacchetto STOP non contiene alcuna informazione utile: indica soltanto all'algoritmo di terminare l'esecuzione.

Alla ricezione di un pacchetto, listener legge il valore del campo SENSOR\_TYPE, e costruisce, in accordo alla relazione sorgente-struttura

dati, la variabile da inviare a SFA.

Il corretto ordinamento dei pacchetti trasmessi a SFA è garantito attraverso l'esplicito utilizzo di un buffer, codificato all'interno di `listener`, in cui i pacchetti vengono temporaneamente salvati prima di essere inviati a SFA, ed eventualmente ordinati sulla base del valore del campo `Seq.N0`. Si osservi che UDP non garantisce che l'ordine dei messaggi ricevuti sia lo stesso con i quali essi sono stati inviati. I messaggi potrebbero essere soggetti a ritardi casuali in base allo stato del sistema operativo. Utilizzando TCP si ovvierebbe a questa problematica, ma l'overhead insito nel protocollo stesso causerebbe un notevole degrado delle performance di SFA.

### 3.2.2 Trasmissione in uscita

La trasmissione dei dati in uscita da SFA avviene, in accordo al protocollo `OUTPUT_PROTOCOL` tra `listener` e `OBCU`.

Come specificato, la comunicazione è posta in essere, a livello fisico, attraverso il protocollo `LTE`, mentre a livello trasporto si è scelto di continuare a usare `UDP` in luogo di `TCP`, col fine di massimizzare le *performance* del sistema. Il conseguente rischio di ricevere alcuni messaggi in maniera errata, o non riceverli del tutto, è tanto più elevato quanto l'ambiente entro cui si propaga fisicamente la radiazione elettromagnetica è più disturbato da sorgenti esterne e corpi fisici interposti.

Questa problematica è risolta a livello software, attraverso l'esplicito utilizzo di un meccanismo di `acknowledgment` simile a quello utilizzato da `TCP`: ciascun pacchetto in uscita da SFA viene indicizzato con un *sequence number* e, in ricezione, viene inviato ogni secondo un *ack* replicante l'ultimo numero di sequenza correttamente ricevuto. Anche in questo caso, il protocollo definisce il formato del *payload* del pacchetto `UDP` inviato da `listener`, ed è riportato in tabella 4.

| Campo                   | Descrizione              | Indici di bit | Tipo                  |
|-------------------------|--------------------------|---------------|-----------------------|
| <code>Seq.N0</code>     | Numero di sequenza       | 0-15          | <code>uint16_t</code> |
| <code>Epoch</code>      | Timestamp                | 16-79         | <code>double</code>   |
| <code>FU_ARC_LEN</code> | Progressiva chilometrica | 80-143        | <code>double</code>   |

Tabella 4: Protocollo di comunicazione in uscita

In ricezione, OBCU dovrà inviare un pacchetto *ack* al mittente, ed il suo formato è descritto in tabella 5. Si osservi che, ai fini del posizionamento,

| Campo | Descrizione   | Indici di bit | Tipo     |
|-------|---------------|---------------|----------|
| ACK   | Ultimo Seq.NO | 0-15          | uint16_t |

Tabella 5: Formato del pacchetto di *ack*

l'ordinamento dei pacchetti ricevuti non è fondamentale. A differenza di quanto esposto nel caso della trasmissione dei dati in entrata a SFA, è sufficiente che OBCU faccia riferimento al messaggio con il *timestamp* più recente.

### 3.3 SCENARIO DI ESEMPIO

Si suppongano le condizioni iniziali riportate in tabella 6.

| Vettore Velocità                | Progressiva | IMU Sample Rate | ODO Sample Rate |
|---------------------------------|-------------|-----------------|-----------------|
| (0.0, 0.0, 0.0)ms <sup>-1</sup> | 0 km        | 100 Hz          | 20 Hz           |

Tabella 6: Condizioni iniziali

1. t = 0:

- interface-modules invia a `listener` il seguente pacchetto GROUND TRUTH:

| SENSOR_TYPE | Seq. NO | N_INT | N_DOUBLE |
|-------------|---------|-------|----------|
| 0x08        | 0x00    | 0     | 4        |

E vi accoda i seguenti tre valori double: 0.0, 0.0, 0.0, ossia il vettore velocità iniziale, ed il valore double 0.0, che rappresenta la progressiva chilometrica iniziale.

- `listener` riceve il pacchetto e inizializza SFA con:
  - Velocità iniziale: (0, 0, 0)
  - Progressiva chilometrica iniziale: 0.0

2.  $t = t_0$ :

- IMU campiona il seguente vettore accelerazione:

$$\mathbf{a} = (0.0001, -0.0001, -9.8100)$$

Assieme al seguente vettore velocità angolare:

$$\mathbf{v}_{\text{ang}} = (0.0003, -0.0001, 0.0002)$$

Il pacchetto viene inviato a `interface-modules` attraverso i bus dati, e conseguentemente:

- `interface-modules` invia a `listener`, attraverso la socket UDP, il seguente pacchetto IMU:

| <code>SENSOR_TYPE</code> | <code>Seq. NO</code> | <code>N_INT</code> | <code>N_DOUBLE</code> |
|--------------------------|----------------------|--------------------|-----------------------|
| 0x01                     | 0x01                 | 0                  | 6                     |

Accodandovi nell'ordine il vettore accelerazione, e il vettore velocità angolare.

- `listener` riceve il pacchetto, crea e inoltra a SFA la seguente variabile `IMU_POD`:

- `Seq.NO` = 1
- `Epoch` =  $t_0$
- `ACC_X` = 0.0001
- `ACC_Y` = -0.0001
- `ACC_Z` = -9.8100
- `GYRO_X` = 0.0003
- `GYRO_Y` = -0.0001
- `GYRO_Z` = 0.0002

- SFA elabora i dati ricevuti e inizia una computazione parallela per fornire a `listener` una variabile `SFA_OUTPUT_POD` della forma:

- `Seq.NO` = 0
- `FU_ARC_LEN` =  $P_{KM}$

$$3. t_0 < t < t_0 + \frac{1}{ODO\_SAMPLE\_RATE} = t_0 + \frac{1}{20}$$

Fintantoché l'odometro non campiona il suo primo valore di velocità, si ripetono le operazioni viste al passo precedente per ogni campionamento di IMU.

$$4. t = t_0 + \frac{1}{20}$$

- Odometro campiona, e invia a `interface-modules`, il seguente valore di velocità:

$$v = (1.0010)$$

`interface-modules` invia a `listener` il seguente pacchetto ODOMETRO:

| SENSOR_TYPE | Seq. NO | N_INT | N_DOUBLE |
|-------------|---------|-------|----------|
| 0x02        | Seq_N0  | 0     | 2        |

Accordandovi nell'ordine il valore di velocità rilevato, e il valore dello scarto quadratico medio della sorgente, noto a priori, in quanto caratteristica tecnica intrinseca dello strumento di misura, il radar; sia esso `SIGMA_RADAR`.

- `listener` riceve il pacchetto, crea e invia a SFA la seguente variabile `ODO_POD`:
  - `Seq.NO = Seq_NO`
  - `Epoch = t_0 + \frac{1}{20}`
  - `vel = 1.0010`
  - `sigma = SIGMA_RADAR`
- SFA elabora i dati ricevuti e utilizza la rilevazione di velocità in maniera utile a migliorare la stima della posizione.

$$5. t = n t_0 \quad n \in \mathbb{N}^+$$

Ogni secondo, il modulo `STROBE` di `interface-modules`, invia a `listener` un pacchetto della forma:

| SENSOR_TYPE | Seq. NO | N_INT | N_DOUBLE |
|-------------|---------|-------|----------|
| 0x09        | Seq_N0  | 0     | 1        |

Accordandovi un *timestamp* che `listener` inoltra a SFA per scopi di sincronizzazione.

Quanto elencato viene ripetuto per ciascun campionamento successivo di IMU e odometro.

Nonappena la prima uscita di SFA si rende disponibile a `listener` questo si comporta come segue:

- `listener` riceve la variabile `SFA_OUTPUT_POD`, da SFA;
- Supponendo di trovarsi al tempo  $T$ , `listener` costruisce ed invia ad OBCU il seguente pacchetto:

| Seq. NO | Epoch | FU_ARC_LEN |
|---------|-------|------------|
| 0x00    | $T$   | $P_{KM_T}$ |

- OBCU riceve il pacchetto e invia a `listener` l'*ack* 0x00.

# 4

---

## AMBIENTE DI ANALISI

---

Al fine di effettuare adeguate campagne di analisi, è stato progettato un ambiente che permetta l’osservazione non intrusiva del comportamento del sistema descritto.

In questo capitolo si descrivono le principali modifiche apportate al sistema a tale scopo di analisi.

### 4.1 ARCHITETTURA HARDWARE

Partendo dall’architettura nominale descritta nel capitolo 2, si osservano le seguenti modifiche:

- *Sensor Set* viene rimpiazzato da un PC in grado di interfacciarsi con la piattaforma di elaborazione dati attraverso una LAN;
- OBCU viene rimpiazzato dallo stesso PC.

Lo schema dell’architettura descritta è riportato in figura 9.

A livello di RUMI, i bus dati e il collegamento LTE sono stati rimpiazzati da una connessione LAN, ma permangono inalterate le interazioni ivi osservabili.

Le modifiche descritte permettono di osservare variazioni nel comportamento di SFA al variare di parametri in ingresso come numero di sensori abilitati e rumore di misura. L’accesso LAN alla piattaforma di elaborazione dati, permette altresì di simulare guasti nel sistema di comunicazione, come ad esempio guasti hardware nei bus dati, oppure perdite di segnale LTE.

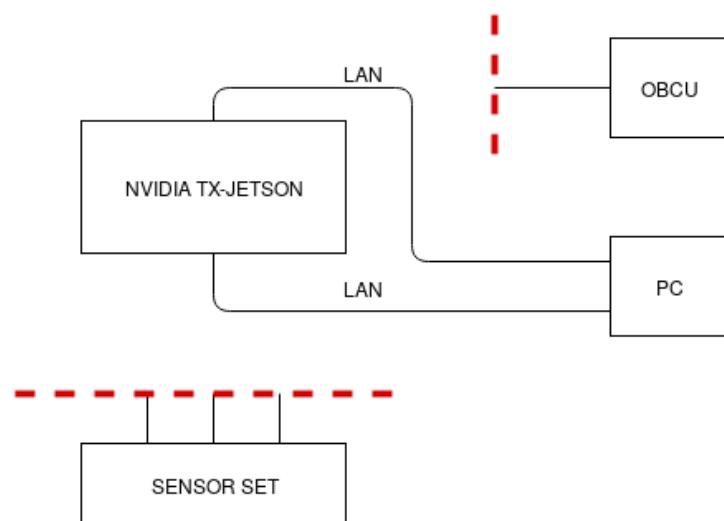


Figura 9: Modifiche architettura hardware

# 5

---

## ESPERIMENTI E RISULTATI

---



# 6

---

## CONCLUSIONI

---



---

## BIBLIOGRAFIA

---

- [1] A. Mirabadi et al, *Application of sensor fusion to railway systems*, IEEE (1996). (Cited on page 11.)
- [2] X. Liu, A. Goldsmith, *Kalman Filtering with Partial Observation Losses*, Department of Electrical Engineering, Stanford University, Stanford, USA (Cited on page 12.)
- [3] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications 2nd edn*, Springer, New York (2011) (Cited on page 15.)
- [4] A. Ceccarelli et al, *Basic Concepts on Systems of Systems, Cyber-Physical Systems of Systems*, Springer (2017) (Cited on page 16.)
- [5] H. Kopetz, W. Ochsenreiter, *Clock synchronization in distributed real-time systems*, IEEE (1987) (Cited on page 16.)
- [6] K. Wolter et al, *Resilience Assessment and Evaluation of Computing Systems*, Springer (2012) (Cited on page 16.)
- [7] M.J. Pont, *Patterns for Time-Triggered Embedded Systems*, Addison-Wesley (2001) (Cited on page 17.)
- [8] H. Kopetz, *Event-Triggered versus Time-Triggered Real-Time Systems*, Springer (1991) (Cited on pages 17 and 18.)
- [9] J.F Kurose, K.W Ross, *Computer Networking: A Top-Down Approach*, Pearson (2013) (Cited on page 18.)