UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

Realizzazione di un soft-sensor basato sulla tecnologia LoRa per localizzazione in campo aperto

Relatore: Prof. Giada Giorgi

Laureando: Samuele Corrò

ANNO ACCADEMICO 2022–2023

Sommario

1. Introduzione

Il tracciamento è un elemento essenziale in diversi settori applicativi: dalla logistica ai trasporti, alle attività all'aperto. Monitorare in tempo reale la posizione di persone, animali o oggetti assume rilevanza in molti ambiti, in quanto consente di stabilire sia la loro posizione geografica, sia altre informazioni come percorso seguito, velocità, ecc., rendendo più efficienti le operazioni di soccorso o rintracciamento in caso di emergenza.

Per effettuare il tracking esistono diverse tecnologie; la più utilizzata è basata sul GNSS (Global Navigation Satellite System). Si tratta di un sistema di navigazione satellitare globale, basato su costellazioni di satelliti, tra cui il più utilizzato è l'americano GPS (Global Position System) e il più recente Galileo, gestito dall'Unione Europea che comprende 30 satelliti.

I sistemi satellitari, tuttavia, presentano limitazioni di funzionamento in ambienti interni o in aree boschive, dove il segnale satellitare è scarsamente o per nulla rilevato. Per questi luoghi si ricorre a tecnologie radio, ad esempio, 3G, 4G, 5G, Wi-Fi e Bluetooth. Nel caso di alcuni sport all'aperto, come l'escursionismo, la mountain bike e lo sci di fondo, gli ambienti in cui vengono praticati sono spesso boschivi o montani, caratterizzati da una infrastruttura cellulare molto limitata e spessp assente. In questi casi non è possibile utilizzare i tradizionali strumenti per il tracciamento, che si basano sull'utilizzo di una qualche infrastruttura già esistente.

In questi contesti una soluzione è fornita dalla tecnologia LoRaWAN, che permette di ricoprire un'area di decine di chilometri con un costo molto limitato e una durata energetica più longeva dei sistemi satellitari. Sfruttando la tecnologia LoRa si possono adottare diverse tecniche per la localizzazione basate ad esempio sull'angolo di arrivo (AoA), sul tempo di arrivo (ToA) o sulla potenza del segnale ricevuto (RSSI). In questa tesi verrà considerata l'ultima tecnica.

L'obiettivo principale di questo lavoro di tesi consiste nello sviluppare un soft-sensor, in grado di stimare la posizione dell'utente che si sposta in un percorso chiuso, su un terreno collinare, sfruttando il parametro RSSI ricevuto attraverso un sistema trasmettitore-ricevitore Lora e la mappatura effettuata per mezzo di un sistema di localizzazione satellitare.

2. Setup di misura

Il sistema di misura utilizzato per le misure di RSSI si basa sulla scheda STM32L476-NUCLEO di STMicroelectronics [1], che offre prestazioni energetiche ridotte e un'ampia personalizzazione. La scheda di valutazione include un Micro Controller Unit (MCU) STM32L476RG. Il dispositivo STM32L476RG è basato sulla tecnologia RISC ARM -M4 a 32 bit che opera ad una frequenza massima di 80 MHz.

Fig. 1: Scheda di sviluppo STM32L476-NUCLEO

La scheda implementa un'interfaccia di comunicazione wireless LoRa, basata sul ricetrasmettitore Hope RF-M96 [2]. Fornisce comunicazioni a spettro esteso a ampio raggio con elevata immunità alle interferenze riducendo al minimo il consumo di corrente. Presenta una sensibilità fino a -148 dBm utilizzando un'antenna omnidirezionale a ʎ/8 con guadagno 2 dB. Il valore di RSSI rilevabile di questa scheda è compreso tra -127 dBm e 0 dBm

Fig. 2 : Hope RF-M96

Per raccogliere i dati GPS si viene utilizzato un modulo GPS GY-NEO6MV2 [3] che raggiunge una sensibilità di circa -161 dBm e altitudine 1,5 km. Può tracciare 22 satelliti su 50 canali consumando 45mA di corrente. Il tutto alimentato con un power bank.

Fig. 3: GPS GY-NEO6MV2

Per il ricevitore LoRa, è stato utilizzato un gateway LG308 Dragino con antenna da 2dBi e una sensibilità fino a -140dBm. LG308 utilizza bande di frequenza LoRaWAN standard preconfigurate da utilizzare per diversi paesi. Il gateway è dotato di internet per inviare i dati al server.

Fig. 4: LG308 Dragino

I test preliminari sono stati eseguiti in un terreno di campagna, come si può notare da Fig. 5, il gateway Lora è stato posto nel punto più alto per limitare l'influenza di ostacoli.

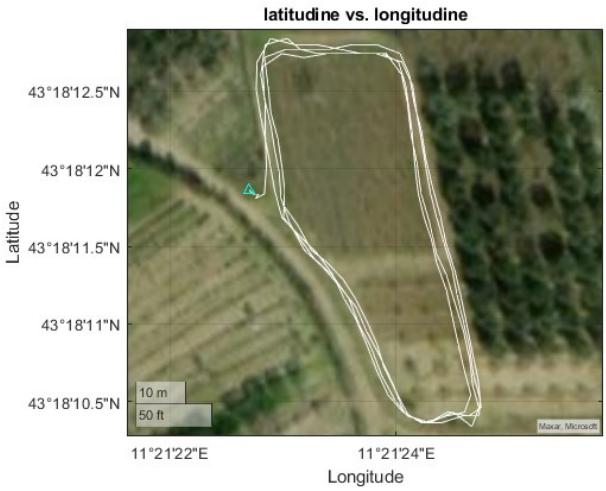


Fig. 5: Percorso GPS satellitare

3. LoRa

LoRa (Long Range) è una tecnica di modulazione di tipo spreading spectrum basata sul sistema CSS (Chirp Spread Spectrum). Essa consiste in una variazione lineare in frequenza denominata chirp (Compressed High Intensity Radar Pulse) ossia un segnale sinusoidale la cui frequenza aumenta o diminuisce nel tempo. Nel 2015 viene fondata la LoRa Alliance, un'associazione che garantisce il supporto del protocollo LoRaWAN, rendendo questa tecnologia adatta ad un gran numero di applicazioni, dall'industria alla IoT (Internet of Things).

Questa tecnologia wireless utilizza frequenze nella banda libera, senza licenza, ISM (Industrial, Scientific, and Medical), precisamente opera nelle bande di frequenza di 433 MHz e 868 MHz in Europa e nella banda a 915 MHz nel Nord America, consentendo trasmissioni a lungo raggio (fino a 20 km). Per normativa in queste bande i dispositivi di trasmissione devono limitare la loro potenza massima a 14 dBm (27 dBm nella banda a 868 MHz). Secondo la normativa europea il Duty Cycle non deve superare l'1%.

La tecnologia LoRa è altamente efficiente dal punto di vista energetico: infatti i dispositivi che la adottano possono funzionare a batteria per lunghi periodi di tempo, grazie al ridotto consumo richiesto per la trasmissione.

Le reti LoRaWAN posso avere topologia a stella, ad albero o mesh a seconda delle esigenze e richiede dispositivi hardware economici e di facile configurazione.

I dispositivi LoRa si dividono in tre classi, denominate A, B, C. Le prime due sono alimentate a batteria mentre la classe C a rete. I dispositivi di classe A sono i più usati, essi hanno solo due finestre di ricezione molto brevi dopo aver trasmesso un pacchetto, successivamente si mette a riposo per risparmiare energia. I dispositivi di classe B aprono finestre di ricezioni a intervalli programmati. Infine, i dispositivi di classe C, che generalmente non dipendono da batterie sono in grado di mantenere costantemente il canale radio in modalità ricezione (a meno che non debbano trasmettere essi stessi), consentendo una trasmissione istantanea di dati. La scelta delle diverse modalità operative influenza il consumo di energia e di conseguenza la durata della batteria del dispositivo.

3.1 RSSI

Nel contesto della tecnologia LoRa, l'RSSI (Received Signal Strenght Indicator) è un parametro importante utilizzato per misurare la potenza del segnale ricevuto. È un valore numerico espresso in decibel (dBm), più è alto il suo valore e più il segnale è forte, al contrario un valore più basso indica un segnale più debole. Oltre alla potenza di uscita del trasmettitore il parametro RSSI è influenzato dalla attenuazione nel percorso, dal guadagno dell'antenna e dalla perdita del cavo/connettore.

$$P_{RX} = \frac{P_{TX} G_{TX} G_{RX}}{L_{FS}}$$

dove P_{RX} è la potenza ricevuta (RSSI) (in mW), P_{TX} è la potenza trasmessa in uscita (in mW), G_{TX} e G_{RX} sono i guadagni delle antenne del trasmettitore e ricevitore e L_{FS} è l'attenuazione nello spazio libero che, secondo uno dei modelli più noti in letteratura (modello di Friis), cresce quadraticamente con la distanza tra ricevitore e trasmettitore:

$$L_{FS} = \left(\frac{4\pi df}{c}\right)^2$$

in cui *d* è la distanza in linea d'aria in metri, *f* indica la frequenza della portante in Hertz e *c* la velocità della luce in metri al secondo.
Analizziamo l'andamento del RSSI al raddoppio della distanza:

$$(P_{RX})_{dB} = (P_{TX})_{dB} + (G_{TX})_{dB} + (G_{RX})_{dB} - (L_{FS})_{dB}$$

$$(L_{FS})_{dB} = 20\log_{10}(\tfrac{4\pi df}{c})$$

$$\begin{aligned}(L_{FS})'_{dB} &= 20\log_{10}(\frac{4\pi(2d)f}{c}) = \\ &= 20\log_{10}(\tfrac{4\pi df}{c}) + 20\log_{10}(2) \\ &= 20\log_{10}(\tfrac{4\pi df}{c}) + 6dB\end{aligned}$$

$$(P_{RX})'_{dB} - (P_{RX})_{dB} = -6dB$$

Si conclude che in linea teorica al raddoppio della distanza tra trasmettitore e ricevitore, il valore di RSSI diminuisce di 6 dB in spazio aperto.

Attraverso l'utilizzo del setup di misura, saranno registrati i dati relativi al valore di RSSI lungo il percorso. Ogni punto di misurazione sarà associato alle coordinate GPS corrispondenti e all'orario in cui è stata effettuata l'acquisizione del dato. Sfruttando le coordinate GPS, è stato possibile calcolare la distanza in linea d'aria tra ciascun punto di acquisizione e il gateway.

Rappresentando graficamente i dati di RSSI e la distanza, ed eseguendo un'interpolazione con un'equazione di secondo grado è possibile verificare l'andamento del RSSI.

Dalla Fig. 6 si può verificare l'andamento dell'RSSI, in questo caso diminuisce di 10dB raddoppiando la distanza, ciò è dovuto agli ostacoli fisici presenti tra trasmettitore e ricevitore, il parametro che tiene conto di questa perdita è il path loss.

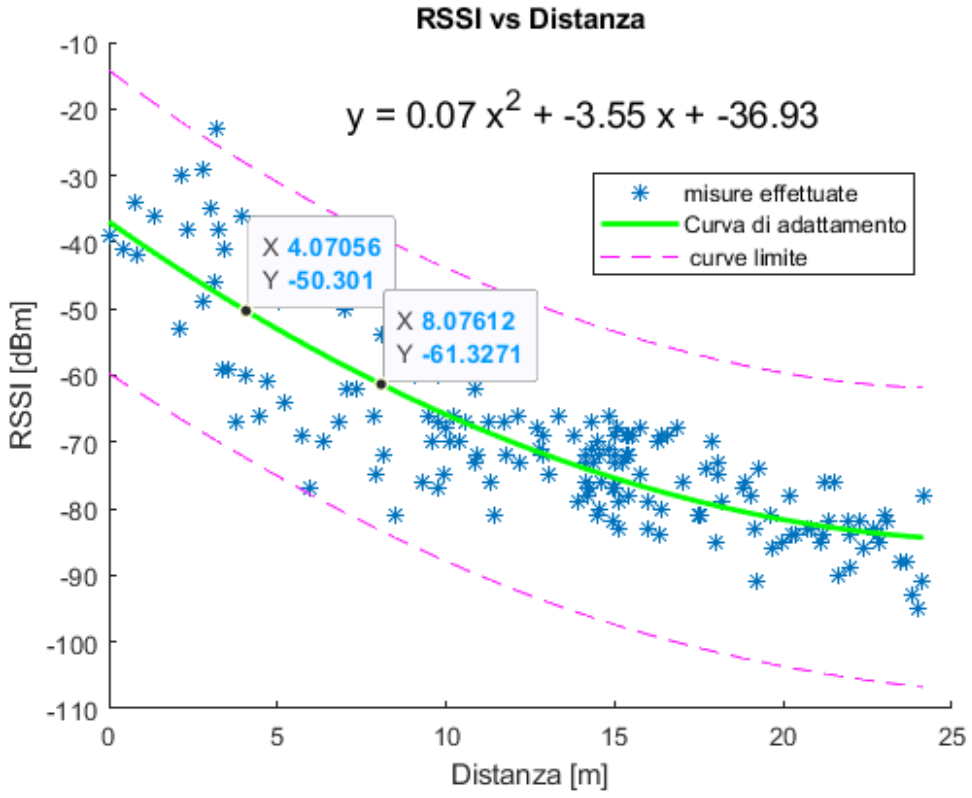


Fig. 6: In verde la curva di adattamento delle misure totali effettuate (in blu), in viola le curve di tolleranza entro le quali sono compresi tutti i dati. Si può notare che per un valore generico di RSSI la distanza varia di decine di metri.

3.2 Path Loss

Il Path Loss descrive l'attenuazione che subisce il segnale radio che si propaga da un'antenna trasmittente a un'antenna ricevente attraverso uno spazio aperto. Questo parametro è influenzato da diversi fattori, ad esempio la diffrazione, la riflessione, o fattori ambientali come ostacoli presenti sul percorso. Conseguentemente, il valore di path loss non è costante ma varia in ogni punto del percorso. Il valore di path loss può essere ricavato mediante la formula di Friis non ideale.

$$P_{rx} = P_{tx} G_{tx} G_{rx} \left(\frac{c}{4\pi f}\right)^2 \left(\frac{1}{d}\right)^n$$

$$(P_{rx})_{db} = A - 10\, n \log_{10}(d)$$

n indica l'esponente di path loss, A [dB] è la potenza ricevuta alla distanza di un metro.
Nel seguito si riporta il codice per il calcolo del path loss dai dati raccolti, per ottenere un calcolo accurato verranno tolti i dati inferiori al metro di distanza dal gateway. La funzione polyfit restituisce i coefficienti del polinomio inserito, restituendo così il valore di pathloss e il valore di RSSI a un metro di distanza.

```
%creo una tabella rimuovendo i dati inferiori al metro di distanza, quindi con
indice maggiore di 6.
 x= table();
 x.rssi=P_sorted.rssi(6:end);
 x.S0=P_sorted.S0(6:end);

 %calcolo i coefficenti della formula di Friis che interpola i valori
sperimentali di RSSI misurati durante l'esperimento.
 p1=polyfit(-10*log10(x.S0),x.rssi,1); %funzionione interpolatrice ricavata
dalla formula di Friis
 N=p1(1)  %coefficiente di di path loss
```

```
N = 4.4174
```

```
 A=p1(2)  %potenza a un metro di distanza
```

```
A = -23.3336
```

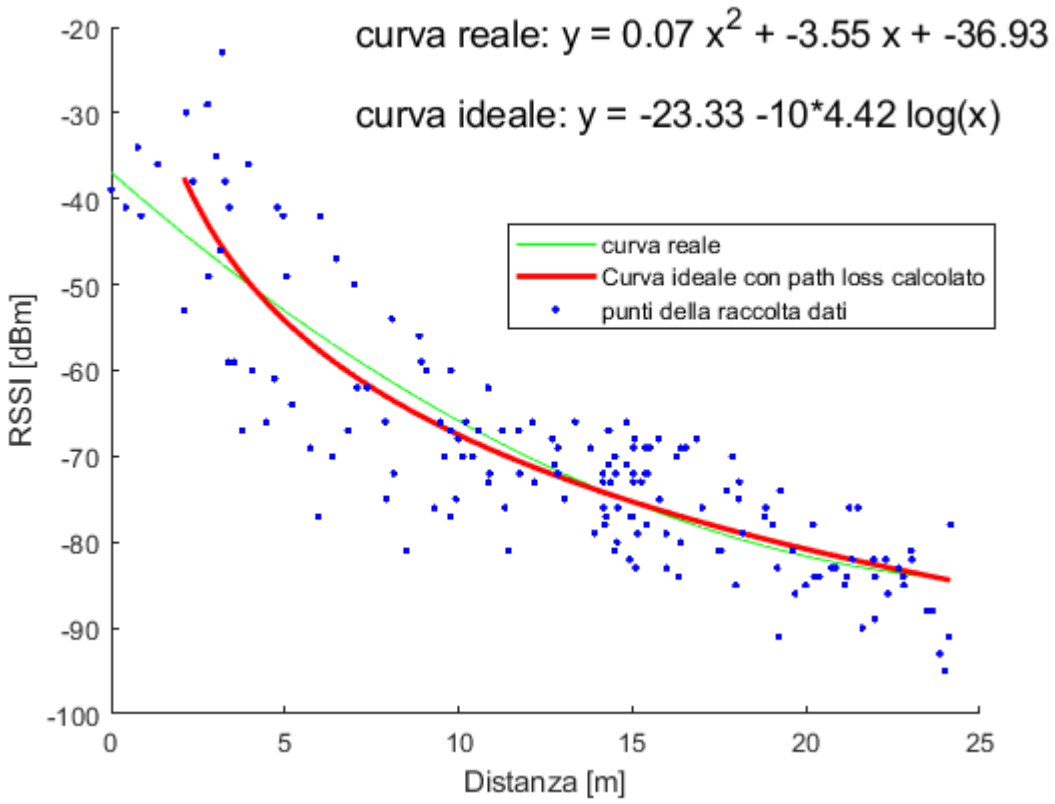


Fig. 7: Retta interpolante (verde) dei dati raccolti confrontata con la curva ideale (rosso) del path-loss calcolato con la formula di Friis. In blu i punti della raccolta dati.

4. Obiettivi

L'obiettivo principale di questo lavoro di tesi consiste nello sviluppare un soft-sensor, in grado di stimare la posizione dell'utente che si sposta in un percorso chiuso, su un terreno collinare, sfruttando il parametro RSSI ricevuto attraverso un sistema trasmettitore-ricevitore Lora e la mappatura effettuata per mezzo dei dati raccolti in una prima fase usando un sistema satellitare. Per fare ciò bisogna svolgere due diverse operazioni:

4.1 Mappatura

Un'azione preliminare consiste nel mappare l'area da monitorare acquisendo i valori di RSSI lungo il percorso assieme ai valori di latitudine e longitudine per ciascun punto di misura. Questi dati verranno utilizzati sia per la costruzione del sensore software, sia per il confronto finale con i valori stimati. Per una maggiore accuratezza nella realizzazione del sensore software, è opportuno effettuare una ripetizione della mappatura così da eliminare il rumore causato dagli eventuali ostacoli fisici che si frappongono tra il trasmettitore e il ricevitore e da realizzarla con condizioni atmosferiche differenti. Questi, infatti, potrebbero influenzare il valore di RSSI rendendo più difficile e meno accurata la stima della posizione. Poiché i dati sperimentali a disposizione per questa tesi corrispondevano a quattro ripetizioni dello stesso percorso, si è scelto di usare i primi tre tracciati per la realizzazione del sensore software, e l'ultimo percorso per la validazione del sensore realizzato.

4.2 Analisi

Si procederà poi con il confronto dei valori RSSI ricevuti dal sistema Lora con quelli di mappatura, ricavati per il sensore software, ottenendo così una stima della posizione dell'utente. Nel corso di questa tesi si sono sviluppate diverse tecniche per lo stima di questa grandezza, verificando quella più efficace. L'accuratezza non è un elemento essenziale per lo sviluppo del sensore dal momento che bastano poche decine di metri perché la ricerca risulti efficace.
Per valutare l'accuratezza del sensore si misurerà la distanza in metri tra il punto stimato e il riferimento lungo il percorso e quella in linea d'aria tra i due. Gli algoritmi di test verranno iterati per ogni punto del percorso 4, usato come segnale di input.

5. Mappatura

I dati rappresentati in Fig. 8 mostrano le quattro realizzazioni dello stesso percorso, per ciascuno dei quali sono stati ricavati i valori delle coordinate GPS e i corrispondenti valori di RSSI. Come già evidenziato, lo stesso tracciato è stato percorso per quattro volte, tre di queste serviranno per la mappatura, mentre la quarta per il test.

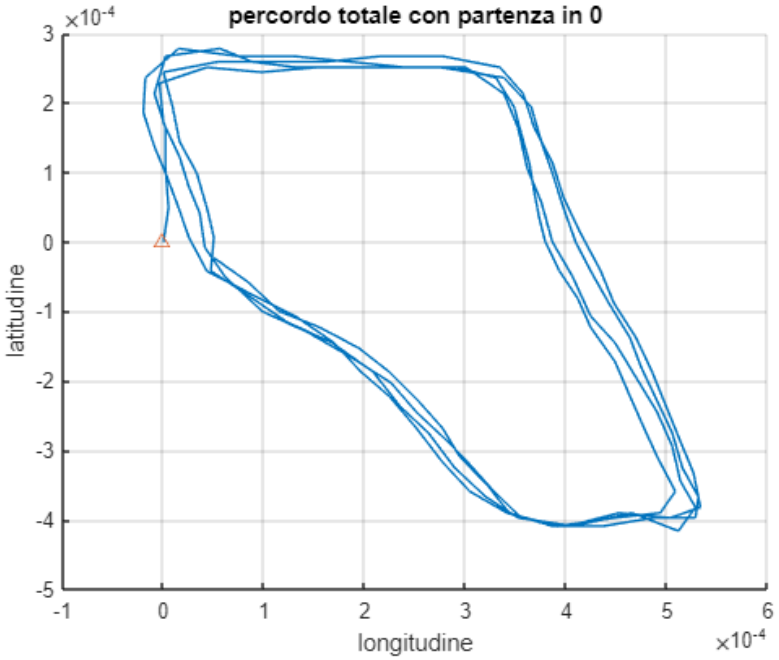


Fig. 8: Percorso totale composta dai 4 percorsi con longitudine e latitudine traslaste in zero. Punto iniziale rappresentato dal triangolo rosso.

Il primo problema che è stato necessario risolvere consisteva in una corretta segmentazione dei quattro percorsi. Per dividere i percorsi si è fatto riferimento all'istante in cui l'utente si trovava alla minima distanza rispetto al punto iniziale, indicato con un triangolo rosso in Fig. 8 e corrispondente alle coordinate (0,0) – si è scelto di normalizzare le coordinate di latitudine e longitudine. Il risultato della segmentazione è illustrato in Fig. 9. Questa figura rappresenta in blu la distanza in linea d'aria in rapporto con il tempo trascorso dalla prima misura ricevuta. Il tempo di percorrenza, in secondi, viene calcolato in base alla ricezione del primo dato, mentre la distanza in linea d'aria, tra soggetto e origine, viene calcolata utilizzando la formula dell'emisenoverso [4]. Questa formula calcola la distanza di due punti su una sfera di raggio pari a quello terrestre.

$$d = 2 * R * \arcsin\left(\sqrt{\sin^2\left(\frac{\lambda_B - \lambda_a}{2}\right) + \cos(\lambda_a)\cos(\lambda_B)\sin^2\left(\frac{\mu_B - \mu_a}{2}\right)}\right)$$

d è la distanza in metri tra le due coordinate GPS, R corrisponde al raggio terrestre ossia 6372.8 Km, $\lambda_a \; e \; \lambda_b$ le latitudini in radianti, $\mu_a \; e \; \mu_b$ le longitudini dei punti in radianti.

I punti di minimo rappresentati con gli asterischi rossi rappresentano i punti più vicini al gateway, di conseguenza, segnano il completamento del giro. In base a questi punti di minimo si divideranno i quattro percorsi.

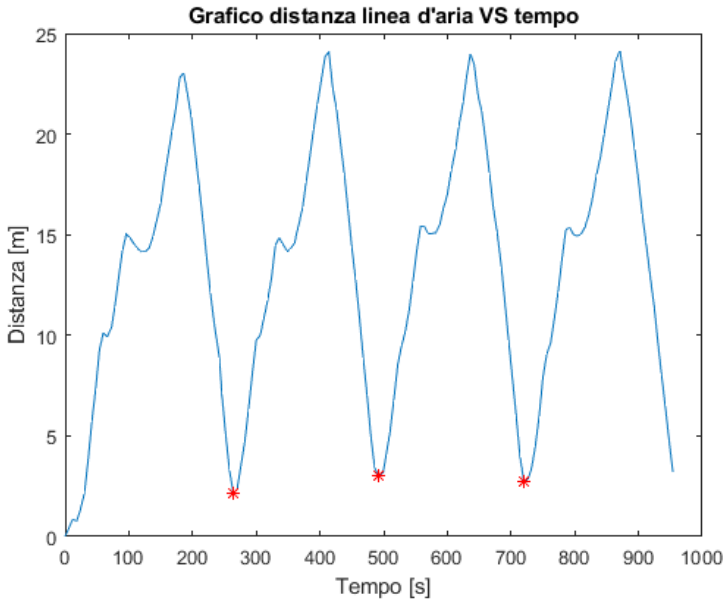


Fig. 9: Distanza in linea d'aria vs tempo di percorrenza, con i punti di separazione dei percorsi alla minima distanza dall'origine.

Dal momento che i valori dell'RSSI sono altamente sensibili alle condizioni ambientali, possono variare rapidamente durante la fase di rilevamento, per questo motivo è stato calcolato un percorso mediato, facendo uso di una media mobile a cinque campioni per ogni percorso. Successivamente si è calcolata la media dei valori ottenuti per i tre percorsi, interpolati con un passo di 1,5 metri e riferiti alla stessa scala delle distanze.

Come si può notare in Fig. 10, per tratti in linea d'aria equidistanti dal gateway, il valore di RSSI rimane circa costante, questa risulterà essere una delle principali limitazioni nella realizzazione del sensore software.

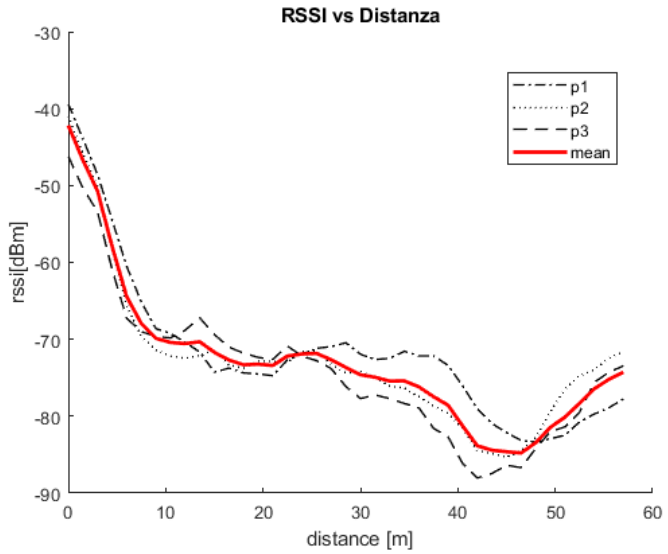


Fig. 10: Valori di RSSI in funzione della distanza percorsa per i 3 percorsi usati per la mappatura (in nero). In rosso è riportato il risultato finale della mappatura che verrà utilizzato per la costruzione del sensore software.

Infine, le coordinate geografiche dei tre percorsi di mappatura vengono unite e mediate per ottenere un percorso GPS più preciso. Il risultato di questa operazione è riportato in Fig. 11, dove è rappresentato il risultato della media dei percorsi GPS (in rosso) e tutti i dati GPS dei tre percorsi di mappatura (in blu).

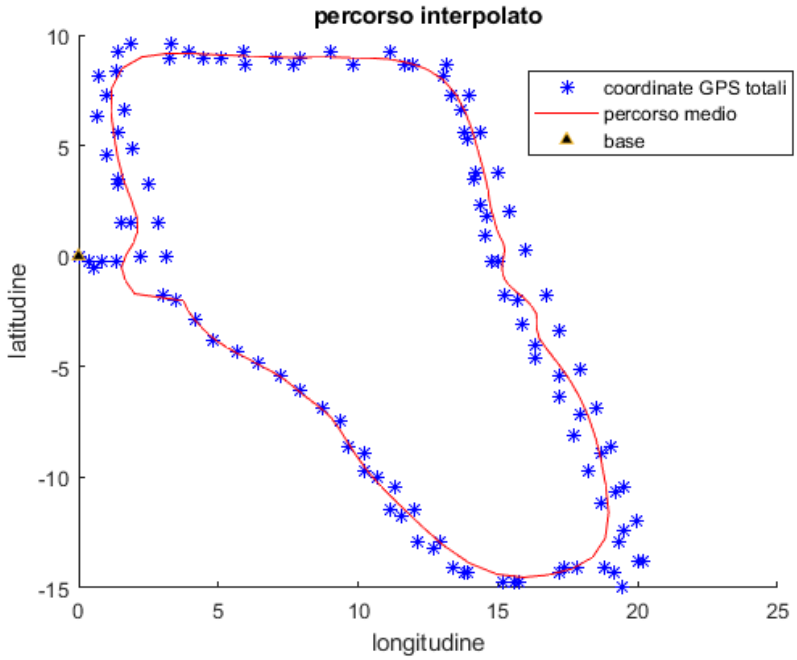


Fig. 11: In rosso è riportato il percorso ricavato mediando in modo opportuno le coordinate GPS dei tre percorsi di mappatura. In blu sono riportate le coordinate GPS di tutti i punti raccolti.

6. Sensore software – Prima realizzazione senza memoria

6.1 Obiettivo

Dato in ingresso un valore misurato di RSSI, si desidera realizzare un sensore software capace di stimare il punto corrispondente del percorso di riferimento. Successivamente, per la validazione del sensore realizzato, si calcolerà la differenza di distanza tra il punto stimato e la posizione reale del trasmettitore (calcolata tramite coordinate GPS) sia come distanza in linea d'aria, che come distanza effettivamente percorsa dal soggetto lungo il sentiero.

6.2 Svolgimento

Al fine di condurre una verifica sul sensore, saranno adoperati i dati relativi al percorso 4. Di conseguenza, risulta impraticabile inserire un valore RSSI generico come input per il test, poiché tale valore potrebbe corrispondere a posizioni diverse lungo il percorso o addirittura non essere disponibile. Al fine di superare questa problematica, si adopera una variabile che identifica il numero di cella dalla quale è prelevato il dato di RSSI sull'array di dati del percorso 4. Si procede con la ricerca del valore a minima distanza tra il valore di RSSI di input con quello della media (l'indice sarà contenuto in idx0). In seguito, si ricava il valore della distanza tra il gateway e il punto stimato seguendo il percorso di mappatura e successivamente la distanza in linea d'aria tra questi due punti. Per verificare l'accuratezza l'output del sensore verrà confrontato con i valori reali.

```
 h=36; %indice dell cella contenente il valore di RSSI da testare scelto
casualmente
 valore_rssi=P4.rssi(h)       %valore RSSI misurato in dB
```

```
valore_rssi = -67 [dBm]
```

```
 % La funzione caratteristica f(rssi, distanza) realizzata dal sensore
software viene per semplicità rappresentata in
 % forma vettoriale. Ogni elemento dell'array fornisce il valore di rssi,
mentre l'indice corrispondente identifica la distanza lungo il percorso.
 %idx0 è l'indice della cella contente il valore di RSSI di test più vicino a
quello medio
 [~,idx0]=min(abs(sensore-P4.rssi(h)));
 %idx1 è l'indice della cella con lo spazio percorso più simile a quello reale
 %questo calcolo è necessario per calcolare la distanza in linea d'aria e
 %trovare le coordinate del punto ricavato
 [~,idx1]=min(abs(PALL.S-S_interp(idx0)));
```

```
dist_di_riferimento=P4.S(h)          %distanza di rifierimento
```

```
dist_di_riferimento = 60.5564 [m]
```

```
dist_stimata=S_interp(idx0)   %distanza stimata dal sensore
```

```
dist_stimata = 7.5000 [m]
```

```
%differenza di posizione lungo il percorso
metri_su_percorso=abs(dist_di_riferimento-dist_stimata);

if(metri_su_percorso>(max(PALL.S)/2))
    metri_su_percorso=abs(max(PALL.S)-metri_su_percorso);
end
metri_su_percorso
```

```
metri_su_percorso = 14.8337 [m]
```

```
%metri in linea d'aria
%per ricavarla si utilizza funzione 'distanza_aerea'
metri_differenza_aerei=distanza_aerea(P4.latitudine(h),P4.longitudine(h),PALL
.latitudine(idx1),PALL.longitudine(idx1))
```

```
metri_differenza_aerei = 13.0209 [m]
```

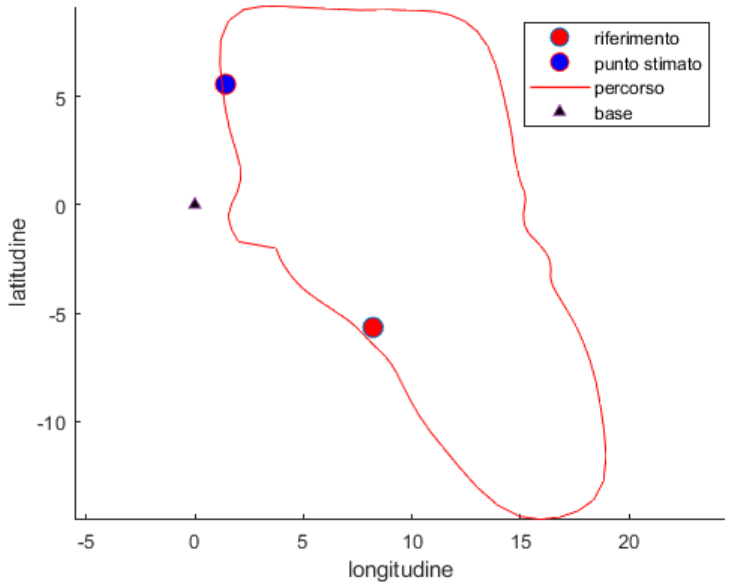


Fig. 12: In questa figura di può verificare il funzionamento del sensore confrontando la posizione reale del soggetto (cerchio rosso) e la posizione stimata dal sensore (cerchio blu). In rosso è riportato il percorso medio di riferimento ricavato durante la mappatura.

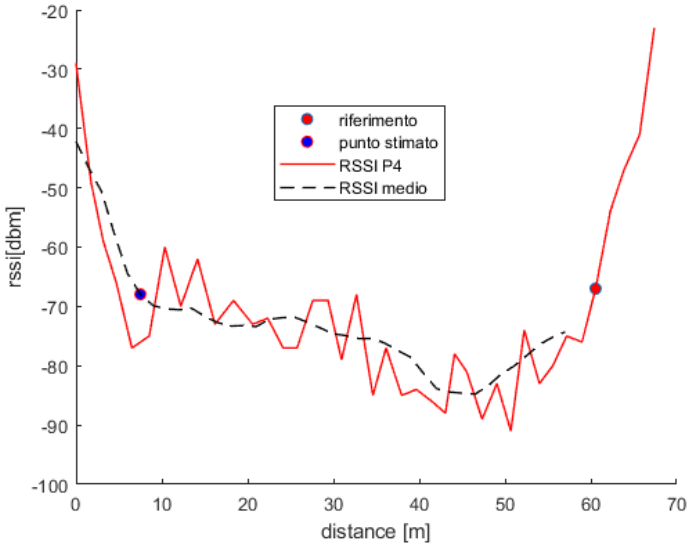


Fig. 13: Confronto tra il valore di RSSI misurato con riferimento alla distanza reale del soggetto lungo il percorso (cerchio rosso) e lo stesso valore di RSSI riferito invece alla distanza stimata dal sensore (cerchio blu). Con linea rossa è riportato il valore di RSSI medio e con linea nera tratteggiata i valori di RSSI misurati per il percorso 4.

6.3 Validazione

Per analizzare l'accuratezza del sensore si può applicare l'algoritmo precedente per ogni punto del percorso 4, ricavando i valori medi delle differenze (tra valori reali e valori stimati) e le deviazioni standard.

```
h=0;
for n=1:length(P4.id)
    %id0 indica la posizione del valore di sensore più vicina a quello
    %testato
    [~,idx0]=min(abs(sensore-P4.rssi(n)));
    %id1 indica la posizione più vicina
    [~,idx1]=min(abs(PALL.S0-S_interp(idx0)));
    %differenza in metri lungo il percorso
    metri_su_percorso(n)=abs(P4.S(n)-S_interp(idx0));

    if(metri_su_percorso(n)>(max(S_interp)/2))
        metri_su_percorso(n)=abs(max(S_interp)-metri_su_percorso(n));
    end
    if(metri_su_percorso(n)<11)
        h=h+1;
    end
    %metri in linea d'aria
```

```
metri_differenza_aerea(n)=distanza_aerea(P4.latitudine(n),P4.longitudine(n),PA
LL.latitudine(idx1),PALL.longitudine(idx1));

 end
 percentuale_riuscita=(h/length(P4.id))*100
```

```
percentuale_riuscita = 72.5000 %
```

```
 media_su_percorso=mean(metri_su_percorso)
```

```
media_su_percorso = 8.3880 [m]
```

```
 dev_std_su_percorso=std(metri_su_percorso)
```

```
dev_std_su_percorso = 8.2248 [m]
```

```
 media_aerea=mean(metri_differenza_aerea)
```

```
media_aerea = 10.1888 [m]
```

```
 dev_std_aerea=std(metri_differenza_aerea)
```

```
dev_std_aerea = 7.5505 [m]
```

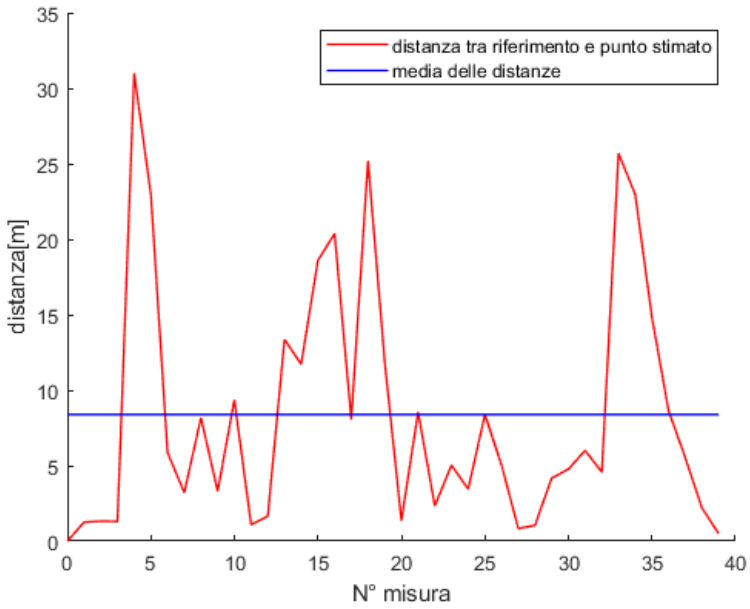


Fig. 14: In rosso la distanza in metri tra il punto di riferimento e quello ricavato dal sensore lungo il percorso mappato, per ogni valore del percorso di test. In blu la media dei valori.

6.4 Considerazioni

Questo primo sensore software realizzato può essere considerato “senza memoria”, dal momento che fa riferimento ad un singolo dato di misura, senza considerare gli stati precedenti, ovvero i valori misurati in precedenza dallo stesso sensore.
Essendo la funzione caratteristica del sensore non monotona, diversi valori di RSSI possono corrispondere a punti a distanze diverse lungo il percorso (Fig. 13). L’algoritmo utilizzando la funzione “min”, restituirà l’indice del dato che si trova a una distanza minima rispetto al dato di test. Tuttavia, ciò non implica necessariamente che l’indice restituito sia quello corretto. Questa problematica non è risolta da questo test dal momento è basato solamente su un unico valore di input.

7. Sensore software – Seconda realizzazione

7.1 Obiettivo

Dato in ingresso un valore misurato di RSSI, si desidera realizzare un sensore capace di stimare il raggio di ricerca e le regioni di tolleranza entro le quale si trova il punto di trasmissione del segnale. In seguito, al fine di validare il sensore realizzato, si procederà al calcolo delle discrepanze tra il raggio di ricerca stimato e quello effettivo, ricavato tramite il calcolo della distanza tra le coordinate GPS del punto e la base.

7.2 Svolgimento

Dai dati GPS misurati per i tre percorsi usati per la mappatura, si calcola la distanza in linea d'aria di ogni punto del percorso rispetto al punto iniziale. Interpolando i dati si trova un'equazione di secondo grado come in Fig. 6. Dato quindi un valore misurato di RSSI, si risolve l'equazione (mediante la funzione esterna "root()") trovando la distanza approssimativa in linea d'aria e le regioni di tolleranza. In Fig. 15 è riportato un esempio di risultato ottenuto con questa soluzione. La circonferenza in blu rappresenta il raggio stimato dal sensore mentre in rosso le circonferenze di tolleranza entro le quali si troverà il punto. È evidente che non è possibile stimare un singolo punto poiché le circonferenze di intersecano in due punti distinti lungo il percorso.

```
% Calcola la curva di adattamento
[p,s] = polyfit(PALL_sorted.S0, PALL_sorted.rssi, 2);
% Calcola la curva di adattamento
[rssi_fit,delta] = polyval(p, PALL_sorted.S0,s);
%dato di test
h=36; %indice dell cella contenente il valore di RSSI da testare scelto
casualmente
%cerco il punto più vicino alla line di interpolazione
[~,idx2]=min(abs(rssi_fit-P4.rssi(h)));
%calcolo la soluzione per il raggio esterno
raggio_up= root2(p(1),p(2),p(3)-rssi_fit(idx2)+1.5*delta(idx2));
%calcolo la soluzione per il raggio interno
raggio_down= root2(p(1),p(2),p(3)-rssi_fit(idx2)-1.5*delta(idx2));
%calcolo la soluzione per il raggio effettivo
raggio_cent= PALL_sorted.S0(idx2)
```

```
raggio_cent = 10.4113 [m]
```

```
%distanza in line d'aria
distanza_linea_d_aria = abs(P4.S0(h)-raggio_cent)
```

```
distanza_linea_d_aria = 0.6273 [m]
```

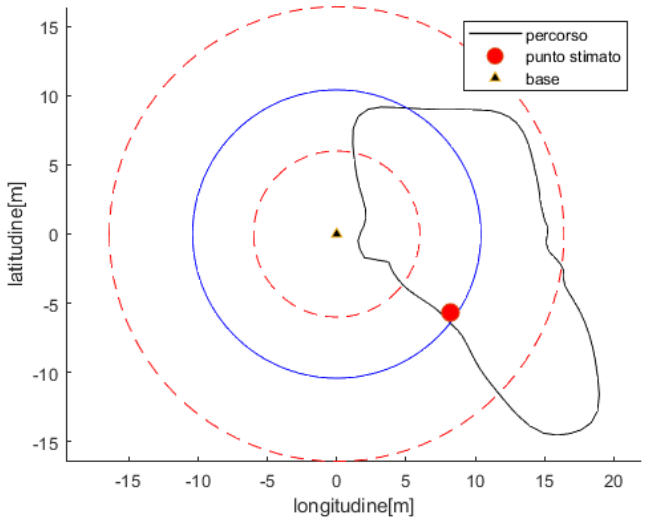


Fig. 15: In questa figura si illustra il funzionamento del sensore software. Con il cerchio rosso è riportata la posizione reale del soggetto (ottenuta usando le coordinate GPS). In nero è riportato invece il percorso di riferimento per il sensore. A partire dal valore di rssi misurato, il sensore stima la distanza in linea d'aria dalla base (circonferenza blu) e la regione di tolleranza, compresa tra le due circonferenze tratteggiate

7.3 Validazione

Per analizzare l'accuratezza della stima fornita dal sensore si applica l'algoritmo precedente per ogni punto del percorso, ricavando i valori medi delle differenze e le deviazioni standard.

```
for n=1:length(P4.id)
[~,idx2]=min(abs(rssi_fit-P4.rssi(n)));
raggio_cent= PALL_sorted.S0(idx2);
distanza_linea_d_aria(n) = abs(P4.S0(n)-raggio_cent);
end

distanza_media=mean(distanza_linea_d_aria)
```

```
distanza_media = 3.3279 [m]
```

```
distanza_dev=std(distanza_linea_d_aria)
```

```
distanza_dev = 2.0734 [m]
```

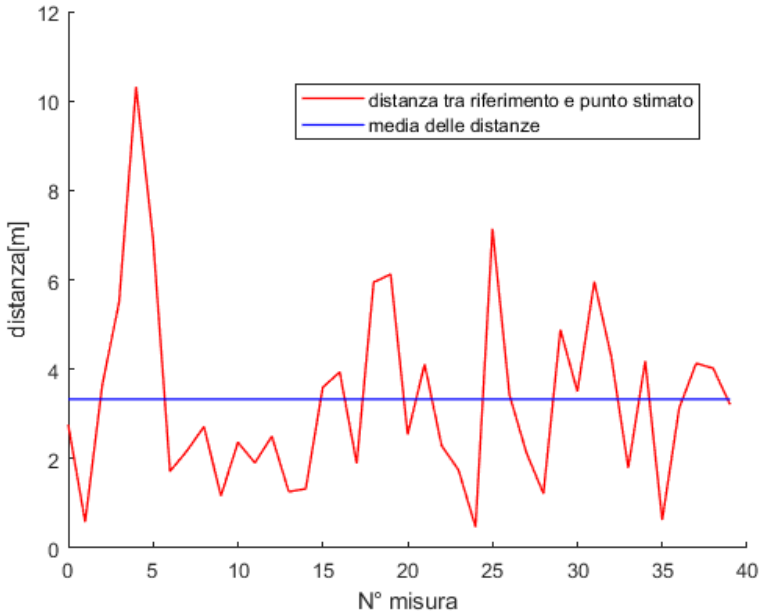


Fig. 16: In rosso la distanza in metri tra il punto di riferimento e quello ricavato dal sensore per ogni valore del percorso di test. In blu la media dei valori.

7.4 Considerazioni

Dalla media e deviazione standard si può notare come questo test possa sembrare preciso, però bisogna tenere in considerazione che esso restituisce la distanza in linea d'aria, di conseguenza, se viene sovrapposto al percorso effettivo si trovano due possibili punti. Le circonferenze rosse indicano inoltre le tolleranze del sistema disegnando così un'area di ricerca. Potrebbe risultare utile per la ricerca su vaste aree dove il percorso non è ben delimitato.

8. Sensore software – Terza realizzazione con memoria

8.1 Obiettivo

Questo sensore, a differenza dei precedenti, sarà dotato di una memoria dati che consentirà di sfruttare due misure consecutive al fine di discriminare punti successivi che presentano una notevole distanza tra loro. Tale approccio mira a migliorare complessivamente l'accuratezza del sensore.

8.2 Svolgimento

Il sensore utilizza una ricerca a tolleranza variabile che verrà descritta in dettaglio nel seguito. I valori misurati verranno confrontati con i valori di riferimento del sensore con una tolleranza iniziale minima, in modo da partire con una regione di ricerca molto ristretta. Si analizzeranno in seguito le distanze relative tra i valori stimanti, incrementando la tolleranza, in metri, della regione di ricerca per considerare quello più vicino. Se essi sono inferiori ad una soglia, fissata a 3 metri, il dato sarà considerato valido, altrimenti si aumenterà la tolleranza del RSSI nel segnale mappato. Si procede così fino a che non si trova un dato valido.

```
 h=34;                      %indice dell cella contenente il valore di RSSI da
testare scelto casualmente
 X = [];                    %array di dati validi
 RSSI_test= P4.rssi(h)      %RSSI attuale
```

```
RSSI_test = -75 [dBbm]
```

```
 RSSIprev=P4.rssi(h-1)      %RSSI precedente
```

```
RSSIprev = -80 [dBm]
```

```
 distanza_reale= P4.S(h)
```

```
distanza_reale = 57.1807 [m]
```

```
 tolleranza1=0;              %tolleranza RSSI attuale
 tolleranza2=0;              %tolleranza RSSI precedente
 x_max=0;                    %tolleranza metri
 index = find(abs(sensore-RSSI_test)< tolleranza1);
 index_prev = find(abs(sensore-RSSIprev)< tolleranza2);
 %ciclo per trovare il primo dato RSSI Attuale
 while(length(index)<1)
     tolleranza1=tolleranza1+0.5;
     index = find(abs(sensore-RSSI_test)<= tolleranza1);
```

```
end
%ciclo per trovare il primo dato RSSI precedente
while(length(index_prev)<1)
    tolleranza2=tolleranza2+0.7;
    index_prev = find(abs(sensore-RSSIprev)<= tolleranza2);
end

dati_selezionati = [];
%fino a che non trovo un dato valido esegue il ciclo
while(isempty(X))
    x_max=0;
    while(isempty(X) && x_max<=3)
        x_hat=S_interp(index);              %distanza attuale
        x_prev=S_interp(index_prev);        %distanza precedente
        th1=x_prev-x_max;
        th2=x_prev+x_max;
        X = [];
        for n = 1:length(th1)
            dati_selezionati = x_hat(x_hat >= th1(n) & x_hat <= th2(n));
            X = [X, dati_selezionati];
        end
        x_max=x_max+1.5;
    end
    tolleranza2=tolleranza2+0.5;
    tolleranza1=tolleranza1+0.7;
    index_prev = find(abs(sensore-RSSIprev)<= tolleranza2);
    index = find(abs(sensore-RSSI_test)<= tolleranza1);
end
%calcolo della media tra i davi validi, trovando la distanza
distanza=mean(X)
```

```
distanza = 54 [m]
```

```
%differenza in metri tra punto stimato e reale
metri_su_percorso=abs(distanza-distanza_reale);
if(metri_su_percorso>max(PALL.S)/2)
    metri_su_percorso=abs(max(PALL.S)-metri_su_percorso);
end
metri_su_percorso
```

```
metri_su_percorso = 3.1807 [m]
```

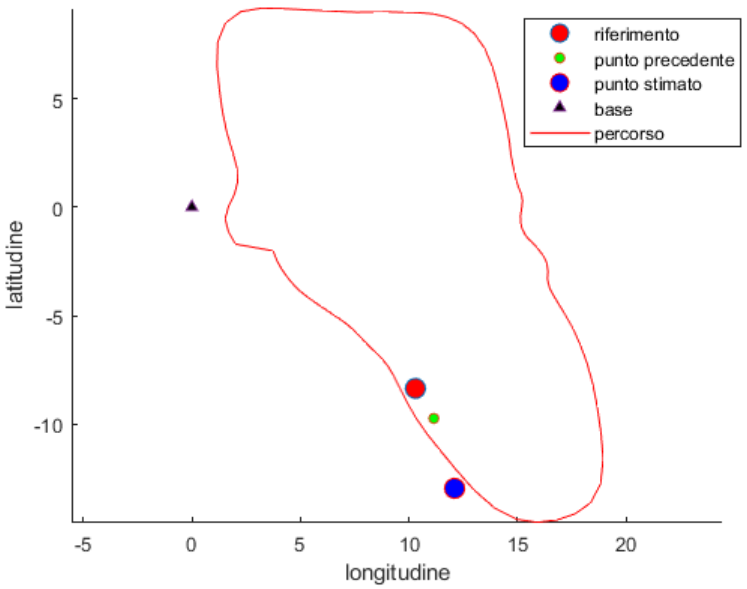
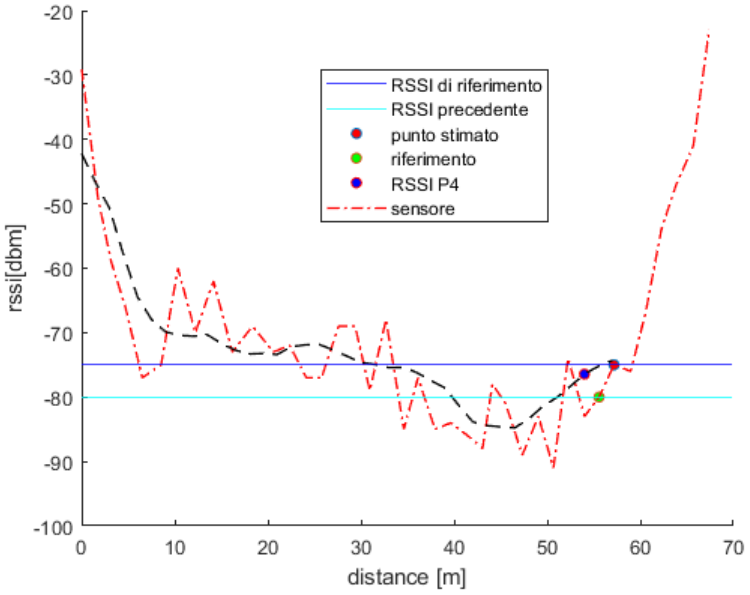


Fig. 17: In questa figura viene rappresentato il funzionamento del sensore software. La linea rossa rappresenta il percorso mappato e il triangolo nero la posizione della base. Con il cerchio rosso è riportato la posizione reale del soggetto e con il cerchio verde la posizione precedente (ottenute usando le coordinate GPS). A partire da questi dal valore di rssi dei due punti il sensore stima la posizione del soggetto (circonferenza blu).



v.s

Fig. 18: Rappresentazione del funzionamento del sensore con memoria: la linea in blu rappresenta il valore di RSSI attuale, in azzurro il valore precedente. Risulta evidente che gli valori uguali di RSSI si ripetono lungo il percorso. Il riferimento con la distanza reale è rappresentato in rosso mentre il valore del punto stimato dal sensore col cerchio in blu. Il punto stimato dal sensore si troverà sempre lungo la linea tratteggiata in nero e compreso tra le rette di riferimento iniziale (in blu e azzurro).

8.3 Validazione

Come nei sensori precedenti, viene testato ogni punto del percorso 4, raccogliendo i dati in una tabella.

```
dati=table(); %tabella con tutti i dati del sensore
f=1;
for h=2:length(P4.rssi)
    X = [];                     %array di dati validi
    RSSI_test= P4.rssi(h);      %RSSI attuale
    RSSIprev=P4.rssi(h-1);      %RSSI precedente
    distanza_reale= P4.S(h);
    tolleranza1=0;              %tolleranza RSSI attuale
    tolleranza2=0;              %tolleranza RSSI precedente
    x_max=0;                    %tolleranza metri
    index = find(abs(sensore-RSSI_test)< tolleranza1);
    index_prev = find(abs(sensore-RSSIprev)< tolleranza2);
    %ciclo per trovare il primo dato RSSI Attuale
    while(length(index)<1)
        tolleranza1=tolleranza1+0.5;
        index = find(abs(sensore-RSSI_test)<= tolleranza1);
    end
    %ciclo per trovare il primo dato RSSI precedente
    while(length(index_prev)<1)
        tolleranza2=tolleranza2+0.7;
        index_prev = find(abs(sensore-RSSIprev)<= tolleranza2);
    end

    dati_selezionati = [];
    %fino a che non trovo un dato valido esegue il ciclo
    while(isempty(X))
        x_max=0;
        while(isempty(X) && x_max<=3)
            x_hat=S_interp(index);              %distanza attuale
            x_prev=S_interp(index_prev);        %distanza precedente
            th1=x_prev-x_max;
            th2=x_prev+x_max;
            X = [];
            for n = 1:length(th1)
                dati_selezionati = x_hat(x_hat >= th1(n) & x_hat <= th2(n));
                X = [X, dati_selezionati];
            end
            x_max=x_max+1.5;
        end

        tolleranza2=tolleranza2+0.5;
        tolleranza1=tolleranza1+0.9;
        index_prev = find(abs(sensore-RSSIprev)<= tolleranza2);
        index = find(abs(sensore-RSSI_test)<= tolleranza1);
```

```
    end
    %calcolo della media tra i davi validi
    distanza=mean(X);
    %differenza in metri tra punto stimato e reale
    metri_su_percorso=abs(distanza-distanza_reale);
    if(metri_su_percorso>max(PALL.S)/2)
        metri_su_percorso=abs(max(PALL.S)-metri_su_percorso);
    end
    %tabella di dati del sensore
    dati.h(f)=h;
    dati.distanza(f)=distanza;
    dati.differenza(f)=metri_su_percorso;
    f=f+1;
end

media=mean(dati.differenza)
```

```
media = 6.2493 [m]
```

```
deviazione_standard=std(dati.differenza)
```

```
deviazione_standard = 6.3340 [m]
```

```
dati_validi=dati(dati.differenza<11, :);  %considero validi solo i dati inferiori a 11m
perc_riuscita=(length(dati_validi.h)/length(P4.id))*100
```

```
perc_riuscita = 82.500 %
```

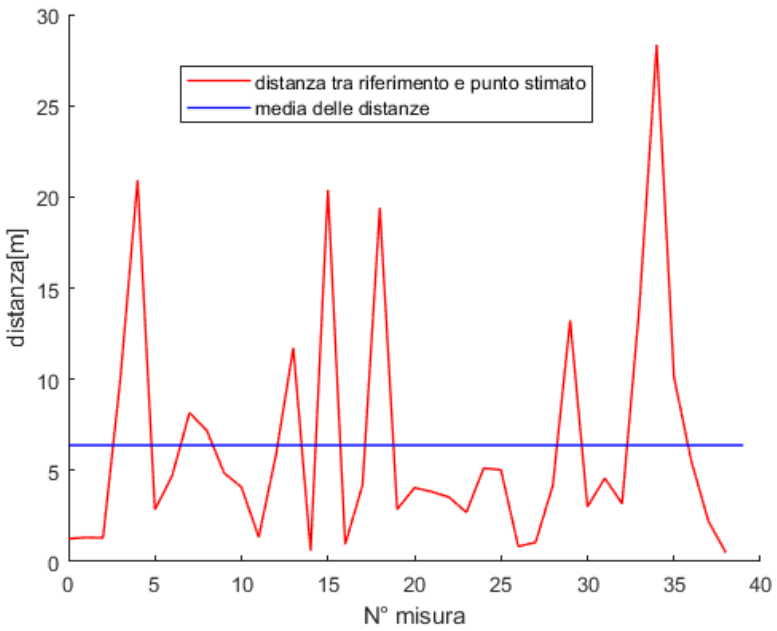


Fig. 19: Distanza (in rosso) in metri tra il punto di riferimento e quello ricavato dal sensore per ogni valore del percorso di test. In blu la media dei valori.

8.4 Considerazioni

Questo sensore si dimostra particolarmente accurato, in quanto evidenzia valori di media e deviazione standard inferiori ai casi precedenti, oltre a una percentuale di successo pari all'80%. Si osserva che, modificare l'incremento di tolleranza di RSSI può influenzare la precisione del sistema, di conseguenza questo valore di incremento va scelto accuratamente e può essere visto come un parametro del sistema da ottimizzare. Tuttavia, possono sorgere problemi quando si confrontano valori di input uguali ma con distanza diverse o in presenza di variazioni eccessive nel segnale.

9. Conclusioni e sviluppi futuri

L'obiettivo di questo lavoro di tesi consiste nella progettazione e implementazione di un soft-sensor finalizzato al tracciamento di individui in ambienti esterni, sfruttando un trasmettitore e un singolo ricevitore basati sulla tecnologia a basso consumo Lora. Sono emerse evidenze che dimostrano come la presenza di vegetazione e l'orografia del territorio influenzino i valori di RSSI misurati, diminuendo l'efficienza del sensore.

I diversi esempi di sensori sviluppati possono essere considerati adeguati in base alle ipotesi iniziali che richiedevano una tolleranza di circa una decina di metri per un tracking valido.

L'ultimo test è risultato essere il più accurato. Tuttavia, va notato che richiede l'utilizzo di due dati consecutivi come input per la sua corretta esecuzione, assumendo che la distanza percorsa dal soggetto monitorato tra le due misurazioni sia trascurabile.

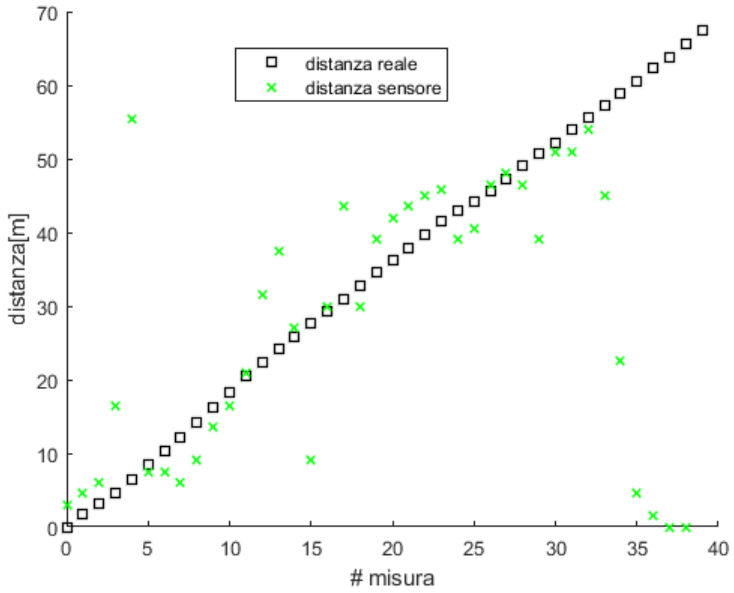


Fig. 20: In Questa figura viene rappresento il risultato della migliore implementazione del soft-sensor: la distanza stimata dal sensore (asterischi verde) viene confrontata con la distanza reale (quadrati nero). Si può notare che, salvo qualche dato non lineare, l'andamento per le misure iniziali (#misura <30) segue l'andamento del segnale di riferimento.

In Fig. 20 è possibile confrontare distanza reale percorsa dal soggetto (quadrati neri) e la distanza stimata usando il terzo tipo di sensore (asterischi verdi). Si nota come alcuni valori stimati si discostano di molto rispetto ai valori reali; questo errore di misura è dovuto alle fluttuazioni del RSSI ad esempio per valori successivi si può avere una variazione di 10 dBm. Per quanto riguarda i valori finali (#misura > 30), il sensore fornisce come stima delle stime di distanza molto basse, quasi nulle. Per spiegare questo comportamento, è necessario considerare

un aspetto fondamentale relativo al Soft-Sensor in questione. Il Soft-Sensor restituisce la distanza percorsa dalla persona lungo il sentiero precedentemente mappato, considerando la direzione seguita dal soggetto durante la mappatura iniziale. Pertanto, si verifica la possibilità che i punti situati in prossimità della base (inizio del percorso) possano essere interpretati dal sensore sia come punti iniziali (a distanza molto bassa, quasi nulla) che, come punti finali (distanza pari alla lunghezza complessiva del percorso mappato).

Gli sviluppi futuri di questo soft-sensor si potranno concentrare sulla correzione di valori stimati in queste zone.

Per migliorarne l'efficienza del sensore e superare il problema di punti con lo stesso RSSI si potrebbe aumentare il numero di gateway che ricevono lo stesso segnale RSSI. Con tre gateway si ottiene un sensore ideale adottando un qualsiasi algoritmo di triangolazione: questa soluzione però richiede un'infrastruttura più complessa e un aumento significativo dei costi di implementazione.

Bibliografia

[1] ST-Microelectronics, «st.com,» [Online]. Available: https://www.st.com/en/evaluation-tools/nucleo-l476rg.html.

[2] hoperf, «www.hoperf.com,» [Online]. Available: https://www.hoperf.com/modules/lora/RFM96.html.

[3] «https://datasheethub.com/gy-neo6mv2-flight-control-gps-module/,» [Online].

[4] K. Gade, «A Non-singular Horizontal Position,» *Norwegian Defence Research Establishment (FFI).*

[5] u-blox, [Online]. Available: https://content.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf.

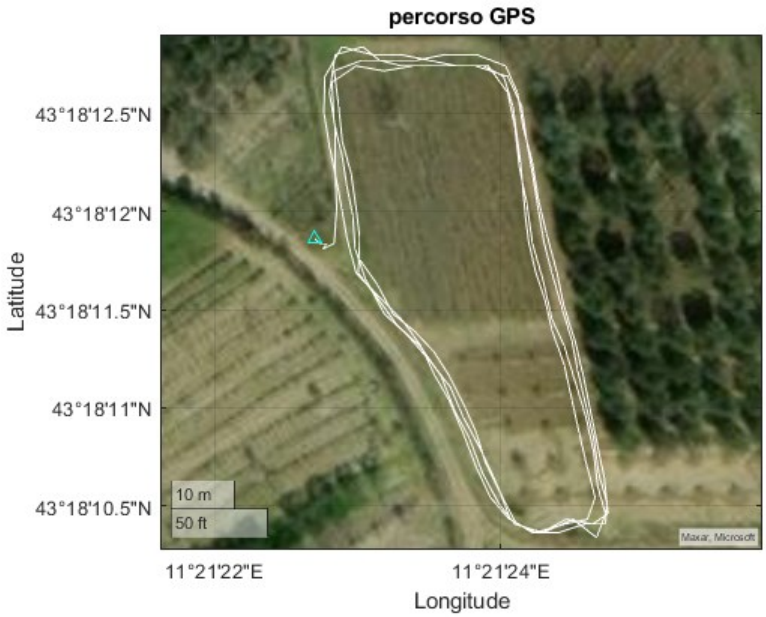
[6] T. J. Rouphael, «Chapter 4 - High-Level Requirements and Link Budget Analysis,» *RF and Digital Signal Processing for Software-Defined Radi,* pp. 87-122, 2009.

[7] G. D. Renzone, G. Giorgi e P. I. 2. p. 1.-6. d. 1. A. Pozzebon IEEE International Symposium on Measurements & Networking (M&N), «"Outdoor sports tracking by means of hybrid GPS-LoRaWAN localization",» *2022 IEEE International Symposium on Measurements & Networking (M&N),* pp. pp. 1-6, 2022.

[8] J. B. H. F. Phui San Cheong, «Comparison of LoRaWAN Classes and their Power Consumption».

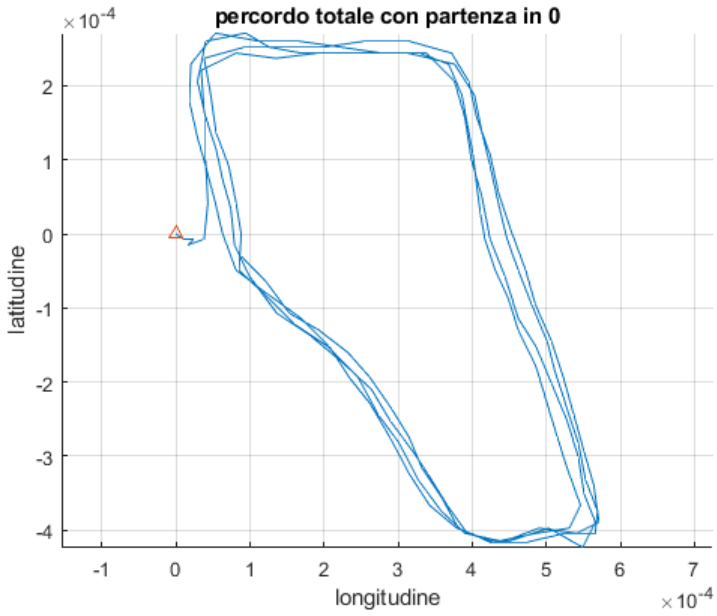
Codice Matlab

```
close all
clear
clc

%leggo dati da file
P = readtable('GPS_LRW_short_test.csv');
%creo due nuove colonne nella tabella di dati principale contenti la
%longitudine e latitudine separata
P.latitudine=latitudine(length(P.payload),P)';
P.longitudine=longitudine(length(P.payload),P)';

%calcolo delle logitudini e latitudini rispetto al punto iniziale
P.lat0=P.latitudine-P.latitudine(1);
P.long0=P.longitudine-P.longitudine(1);

% Create geoplot of P.latitudine and P.longitudine
figure;
geoplot(P.latitudine,P.longitudine,"w",P.latitudine(1),P.longitudine(1),'^c')
;
geobasemap satellite
title("percorso GPS");
```

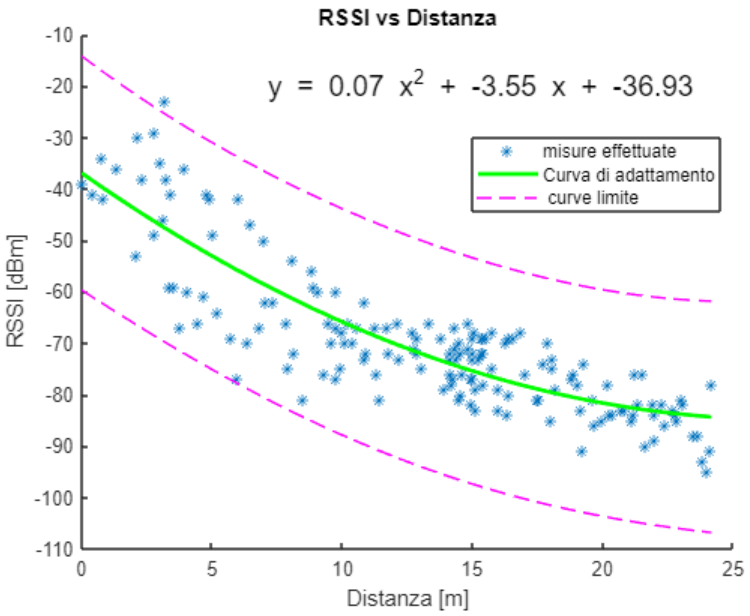



```
%grafico GPS del percorso rispetto al punto iniziale
figure;
hold on;
plot(P.long0,P.lat0);
plot(0,0,"^");
grid on;
ylabel('latitudine'); xlabel('longitudine');
```

```
 title('percordo totale con partenza in 0');
 axis equal
 hold off;
```




```
 %calcolo distanza aerea di ogni punto rispetto al punto iniziale
 for n=1:length(P.payload)

P.S0(n)=distanza_aerea(P.latitudine(1),P.longitudine(1),P.latitudine(n),P.long
itudine(n));
 end
 P.S=distanza(length(P.payload),P.latitudine,P.longitudine)';
 P_sorted = sortrows(P, 'S0');
 %polyfit restituisce i coefficenti del polinomio di grado n
 [p,s] = polyfit(P_sorted.S0, P_sorted.rssi, 2);
 % Calcola i valori della curva di adattamento
 [rssi_fit,delta] = polyval(p, P_sorted.S0,s);

 figure;
 hold on;
 % Traccia il grafico a punti
 plot(P.S0,P.rssi,'*');
 plot(P_sorted.S0, rssi_fit,'color','green','linewidth',2);
 plot(P_sorted.S0,rssi_fit+3*delta, 'm--' ,P_sorted.S0,rssi_fit-3*delta, 'm--'
)
 title('Grafico RSSI vs Distanza con curva di adattamento');
 xlabel('Distanza [m]');
 ylabel('RSSI [dBm]');
 legend('misure effettuate', 'Curva di adattamento', ' curve limite');
 text(7, -20, sprintf('y = %.2f x^2 + %.2f x + %.2f', p(1), p(2), p(3)),
'FontSize', 14);
 title('RSSI vs Distanza');
 hold off;
```



Analisi RSSI

costante di propagazione RSSI = -(10Nlog(d)-A)

A=potenza a 1 metro tramite formula [dB]

N= coefficente di attenuazione

d=distanza

```
 %creo una tabella rimuovendo i dati inferiori al metro di distanza, quindi
con indice maggiore di 6.
 x= table();
 x.rssi=P_sorted.rssi(6:end);
 x.S0=P_sorted.S0(6:end);

 %calcolo i coefficenti della formula di Friis che interpola i valori
sperimentali di RSSI misurati durante l'esperimento.
 p1=polyfit(-10*log10(x.S0),x.rssi,1); %funzionione interpolatrice ricavata
dalla formula di Friis
 N=p1(1)  %coefficiente di di path loss
```

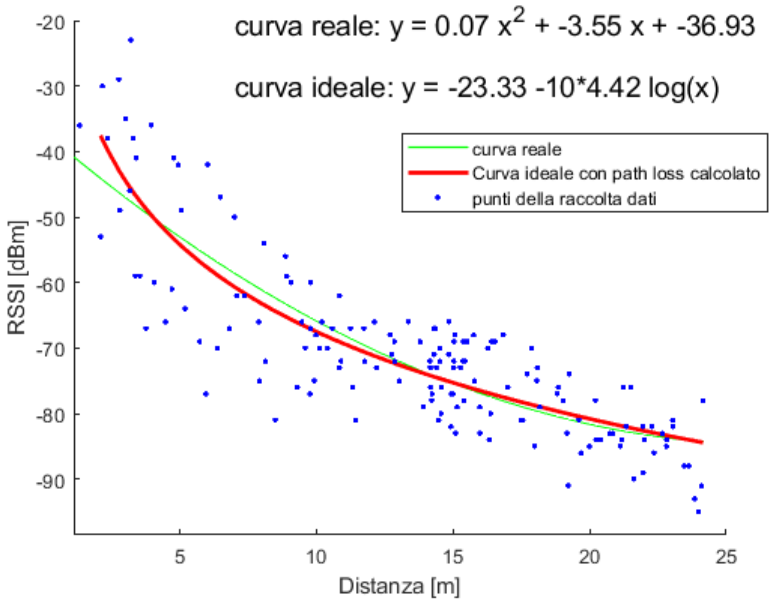
```
N = 4.4174
```

```
 A=p1(2)  %potenza a un metro di distanza
```

```
A = -23.3336
```

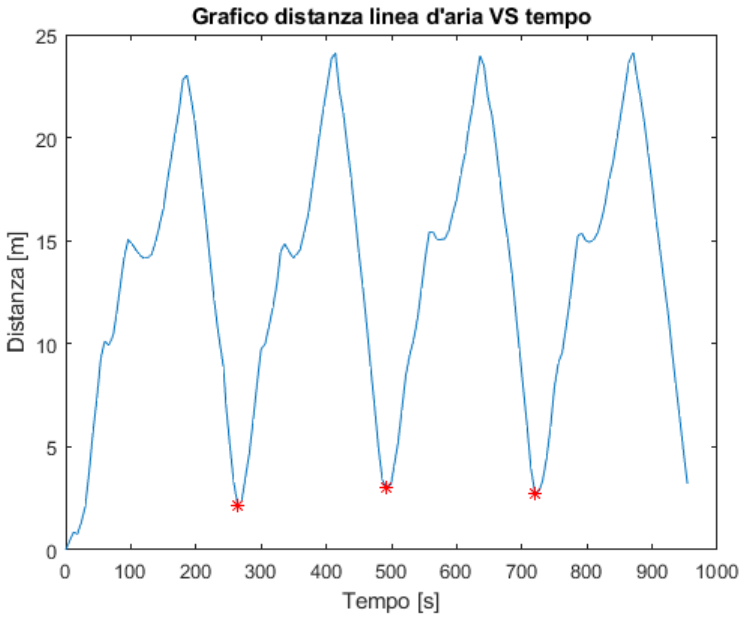
```
 Rssi = -(10*N*log10(x.S0)-A);
 figure;
 hold on;
 plot(P_sorted.S0, rssi_fit,'color','green','linewidth',1);
 plot(x.S0,Rssi,'color','red','linewidth',2);
 plot(P_sorted.S0,P_sorted.rssi,'.b','linewidth',1);
```

```
 xlabel('Distanza [m]');
 ylabel('RSSI [dBm]');
 legend('curva reale', 'Curva ideale con path loss calcolato', 'punti della
raccolta dati');
 text(7, -20, sprintf('curva reale: y = %.2f x^2 + %.2f x + %.2f', p(1), p(2),
p(3)), 'FontSize', 14);
 text(7, -30, sprintf('curva ideale: y = %.2f -10*%.2f log(x)', A, N),
'FontSize', 14);
 hold off;
```



Divisione dei percorsi

```
 %calcolo del tempo in secondi trascorso
 Dt0=tempo(P.timestamp);
 %trova i valori di minimo locali
 Vmin = find(islocalmin(P.S0,'MinProminence',2));

 figure;
 plot(Dt0,P.S0,Dt0(Vmin),P.S0(Vmin), 'r*' )
 title('Grafico distanza linea d''aria VS tempo');
 xlabel('Tempo [s]');
 ylabel('Distanza [m]');
```

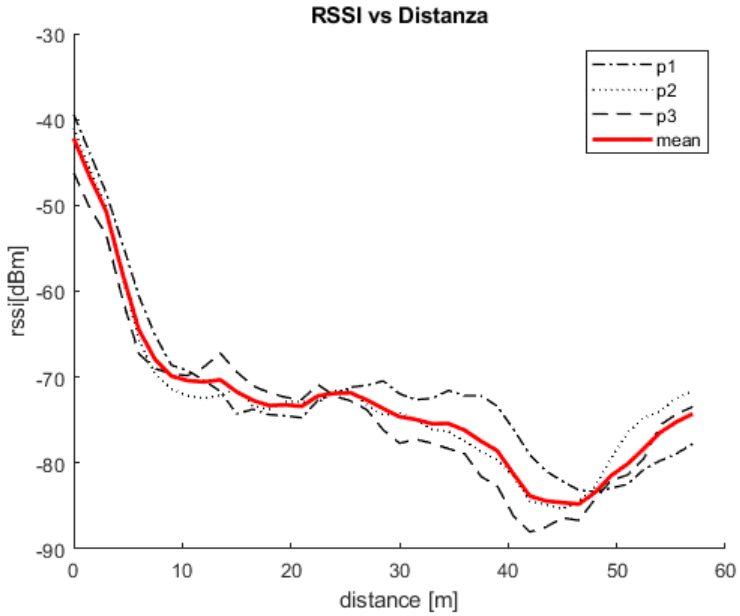



```
%separazione dei percorsi in base al punti più picino alla partenza
P1 = P(1:Vmin(1)-1,1:10);
P2 = P(Vmin(1):Vmin(2)-1,1:10);
P3 = P(Vmin(2):Vmin(3)-1,1:10);
P4 = P(Vmin(3):end,1:10);

%calcolo del percorso con meno punti
DimPerc=min([length(P1.id),length(P2.id),length(P3.id)]);
%tronco i percorsi in base alla dimensione del più piccolo
P1(DimPerc+1:end,:) = [];
P2(DimPerc+1:end,:) = [];
P3(DimPerc+1:end,:) = [];
%calcolo la distanza dal punto iniziale di ogni percorso
P1.S=distanza(DimPerc,P1.latitudine,P1.longitudine)';
P2.S=distanza(DimPerc,P2.latitudine,P2.longitudine)';
P3.S=distanza(DimPerc,P3.latitudine,P3.longitudine)';
P4.S=distanza(length(P4.id),P4.latitudine,P4.longitudine)';
%converto la latitudine in metri
P4.lat_m=lat2m(P4.lat0);
P4.lon_m=lon2m(P4.long0);
```

Creazione sensore

```
%creo un vettori di punti di spazio
S_interp= 0:1.5:57;
rssiP1_interp = interp1(P1.S, P1.rssi, S_interp);
rssiP2_interp = interp1(P2.S, P2.rssi, S_interp);
rssiP3_interp = interp1(P3.S, P3.rssi, S_interp);

%calcolo media mobile
avg_data_P1 = movmean(rssiP1_interp,5);
avg_data_P2 = movmean(rssiP2_interp,5);
```

```
avg_data_P3 = movmean(rssiP3_interp,5);
%calcolo RSSI medio dei 3 percorsi
sensore= mean([avg_data_P1',avg_data_P2',avg_data_P3'],2,"omitnan" );

figure;
hold on;
plot(S_interp, avg_data_P1 , '-.k','linewidth',1);
plot(S_interp, avg_data_P2, ':k','linewidth',1);
plot(S_interp, avg_data_P3,'--k','linewidth',1);
plot(S_interp, sensore,'-r','linewidth',2);
ylabel('rssi[dBm]'); xlabel('distance [m]');
legend('p1','p2','p3','mean');
title('RSSI vs Distanza');
hold off;
```



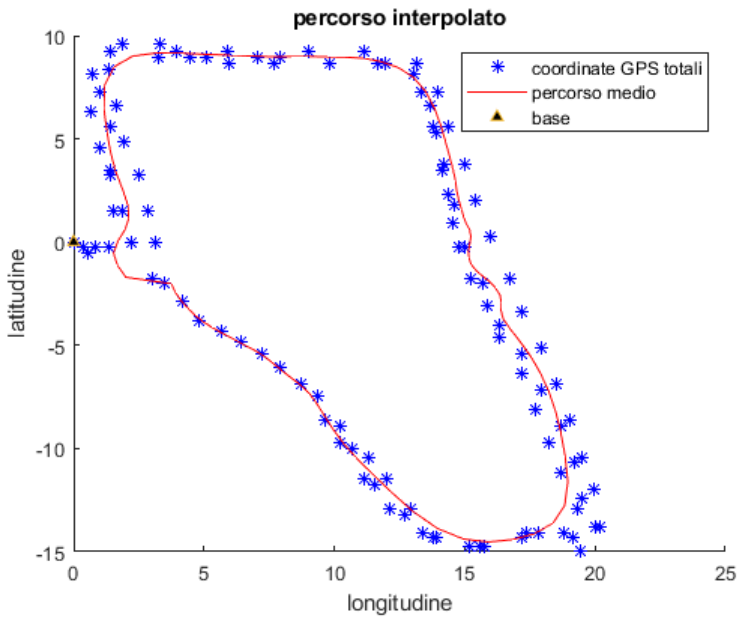
Unione dei dati dei 3 percorsi:

```
rssi= [P1.rssi ; P2.rssi ; P3.rssi];      %rssi
S0 = [P1.S0 ; P2.S0 ; P3.S0];             %distanza aerea dispetto allo 0
lat = [lat2m(P1.lat0); lat2m(P2.lat0) ; lat2m(P3.lat0)];     %lat in metri
lon = [ lon2m(P1.long0); lon2m(P2.long0); lon2m(P3.long0)];  %lon in metri
S= [ P1.S ; P2.S ; P3.S];
latitudine= [P1.latitudine ; P2.latitudine ; P3.latitudine];
longitudine = [ P1.longitudine ; P2.longitudine ; P3.longitudine];
PALL = table(rssi, S0, lat , lon, latitudine,longitudine,S,  'VariableNames',
{'rssi', 'S0', 'lat','lon','latitudine', 'longitudine','S'});
PALL_sorted = sortrows(PALL, 'S0');
```

Unione geografica e media delle cordinate dei 3 percorsi

```
lon_media = mean(PALL.lon);         %longitudine media
lat_media = mean(PALL.lat);         %latituidine media
```

```
 [theta,r] = cart2pol(PALL.lon-lon_media,PALL.lat-lat_media);  %converto in
coordinate polari
 [theta,ind] = sort(theta);  %ordino in base all'angolo
 r = r(ind);
 % rimuovo duplicati
 [theta,IA,~] = unique(theta);
 r = r(IA);      %inserisco dati in r
 theta_new = linspace(min(theta),max(theta),100);
 r_new = interp1(theta,r,theta_new);
 rs = smoothdata(r_new,'gaussian',7); %smooth dei dati con dev std 7
 % conversione in cartesiane
 [longp,latp] = pol2cart(theta_new,rs);
 longp = longp + lon_media;
 latp = latp + lat_media;
 %per chiudere il percorso
 longp(end+1) = longp(1);
 latp(end+1) = latp(1);

 figure;
 hold on;
 title('percorso interpolato');
 ylabel('latitudine'); xlabel('longitudine');
 plot(PALL.lon,PALL.lat,'b*',longp,latp,'r');
 plot(0,0,'^', 'MarkerFaceColor', 'black', 'MarkerSize', 5);
 legend('coordinate GPS totali','percorso medio', 'base');
 hold off;
```



Test 1: Iterato

```
 h=0;
 for n=1:length(P4.id)
     %id0 indica la posizione del valore di sensore più vicina a quello
```

```
    %testato
    [~,idx0]=min(abs(sensore-P4.rssi(n)));
    %id1 indica la posizione più vicina
    [~,idx1]=min(abs(PALL.S0-S_interp(idx0)));
    %differenza in metri lungo il percorso

    metri_su_percorso(n)=abs(P4.S(n)-S_interp(idx0));

    if(metri_su_percorso(n)>(max(PALL.S)/2))
        metri_su_percorso(n)=abs(max(PALL.S)-metri_su_percorso(n));
    end
    if(metri_su_percorso(n)<11)
        h=h+1;
    end
    %metri in linea d'aria

metri_differenza_aerea(n)=distanza_aerea(P4.latitudine(n),P4.longitudine(n),PA
LL.latitudine(idx1),PALL.longitudine(idx1));
 end
 percentuale_riuscita=(h/length(P4.id))*100
```

```
percentuale_riuscita = 72.5000
```

```
 media_su_percorso=mean(metri_su_percorso)
```

```
media_su_percorso = 8.3880
```

```
 dev_std_su_percorso=std(metri_su_percorso)
```

```
dev_std_su_percorso = 8.2248
```

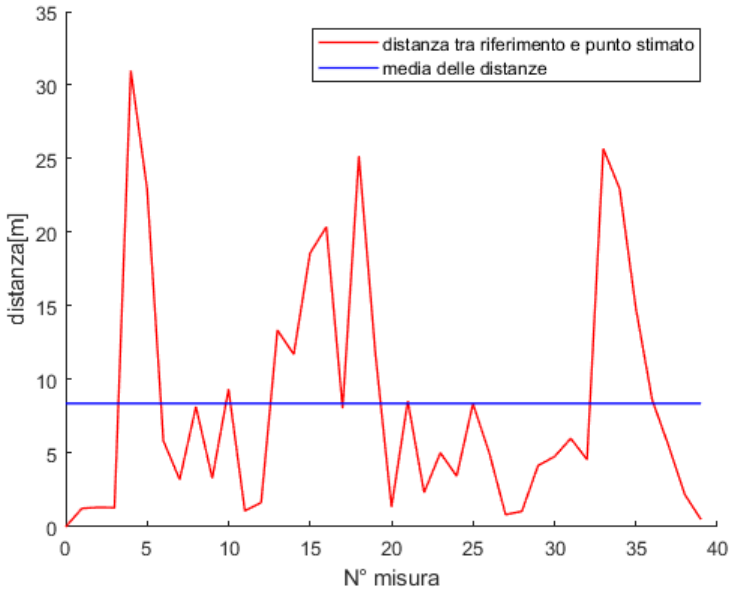
```
 media_aerea=mean(metri_differenza_aerea)
```

```
media_aerea = 10.1888
```

```
 dev_std_aerea=std(metri_differenza_aerea)
```

```
dev_std_aerea = 7.5505
```

```
 figure;
 hold on;
 plot((0:1:length(P4.S)-1), metri_su_percorso,'r','linewidth',1);
 plot((0:1:length(P4.S)-1),
(media_su_percorso)*ones(length(P4.S)),'b','linewidth',1);
 ylabel('distanza[m]'); xlabel('N° misura');
 legend('distanza tra riferimento e punto stimato','media delle distanze');
 hold off;
```



Test 1: base

```
 %test su singolo dato
 h=36; %indice dell cella contenente il valore di RSSI da testare scelto
casualmente
 valore_rssi=P4.rssi(h)       %valore RSSI misurato in dB
```

```
valore_rssi = -67
```

```
 % La funzione caratteristica f(rssi, distanza) realizzata dal sensore
software viene per semplicità rappresentata in
 % forma vettoriale. Ogni elemento dell'array fornisce il valore di rssi,
mentre l'indice corrispondente identifica la distanza lungo il percorso.
 %idx0 è l'indice della cella contente il valore di RSSI di test più vicino a
quello medio
 [~,idx0]=min(abs(sensore-P4.rssi(h)));
 %idx1 è l'indice della cella con lo spazio percorso più simile a quello reale
 %questo calcolo è necessario per calcolare la distanza in linea d'aria e
 %trovare le coordinate del punto ricavato
 [~,idx1]=min(abs(PALL.S-S_interp(idx0)));
 dist_di_riferimento=P4.S(h)         %distanza di rifierimento
```

```
dist_di_riferimento = 60.5564
```

```
 dist_stimata=S_interp(idx0)   %distanza stimata dal sensore
```

```
dist_stimata = 7.5000
```

```
 %differenza di posizione lungo il percorso
 metri_su_percorso=abs(dist_di_riferimento-dist_stimata);

 if(metri_su_percorso>(max(PALL.S)/2))
     metri_su_percorso=abs(max(PALL.S)-metri_su_percorso);
```

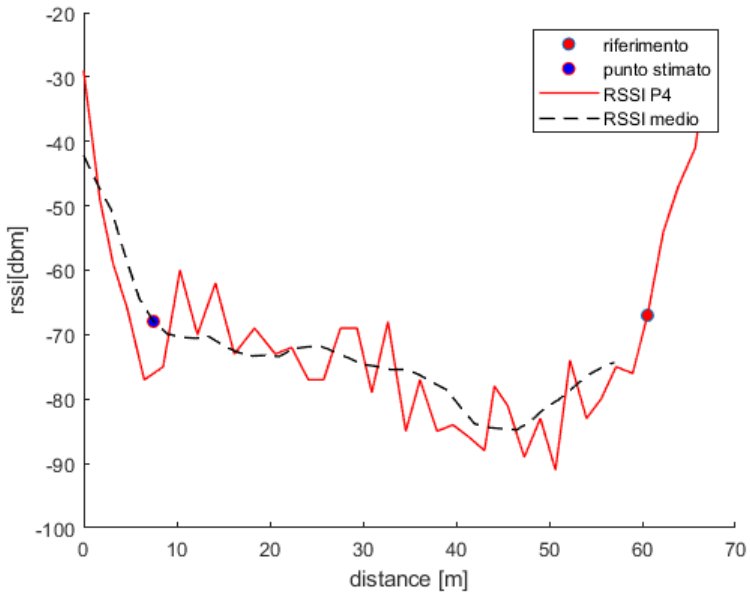
```
 end
 metri_su_percorso
```

```
metri_su_percorso = 14.8337
```

```
 %metri in linea d'aria
 %per ricavarla si utilizza funzione 'distanza_aerea'
 metri_differenza_aerei=distanza_aerea(P4.latitudine(h),P4.longitudine(h),PALL
.latitudine(idx1),PALL.longitudine(idx1))
```

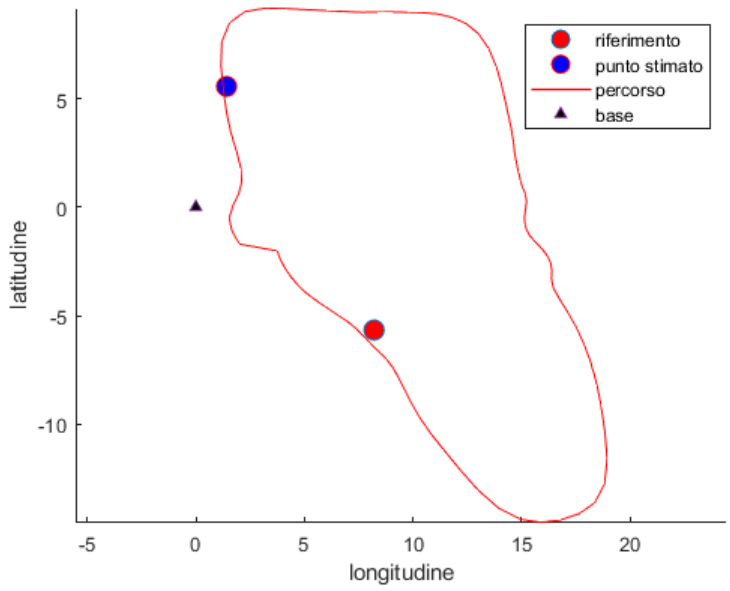
```
metri_differenza_aerei = 13.0209
```

```
 figure;
 hold on;
 plot(P4.S(h),P4.rssi(h),'o', 'MarkerFaceColor', 'red', 'MarkerSize', 6);
 plot(S_interp(idx0)',sensore(idx0),'ro','MarkerFaceColor', 'blue',
'MarkerSize', 6);
 plot(P4.S,P4.rssi,'-r','linewidth',1);
 plot(S_interp, sensore,'--k','linewidth',1);
 ylabel('rssi[dbm]'); xlabel('distance [m]');
 legend('riferimento','punto stimato','RSSI P4','RSSI medio');
 hold off;
```




```
 figure;
 hold on;
 plot(P4.lon_m(h),P4.lat_m(h),'o', 'MarkerFaceColor', 'red', 'MarkerSize',
10);
 plot(PALL.lon(idx1),PALL.lat(idx1),'ro','MarkerFaceColor', 'blue',
'MarkerSize', 10);
 plot(longp,latp,'-r','linewidth',1);
 plot(0,0,'^', 'MarkerFaceColor', 'black', 'MarkerSize', 5);
 ylabel('latitudine'); xlabel('longitudine');
 legend('riferimento','punto stimato','percorso','base');
```

```
axis equal
hold off;
```



TEST 2: base

```
% Calcola la curva di adattamento
[p,s] = polyfit(PALL_sorted.S0, PALL_sorted.rssi, 2);
% Calcola la curva di adattamento
[rssi_fit,delta] = polyval(p, PALL_sorted.S0,s);
%dato di test
h=36; %indice dell cella contenente il valore di RSSI da testare scelto
casualmente
%cerco il punto più vicino alla line di interpolazione
[~,idx2]=min(abs(rssi_fit-P4.rssi(h)));
%calcolo la soluzione per il raggio esterno
raggio_up= root2(p(1),p(2),p(3)-rssi_fit(idx2)+1.5*delta(idx2));
%calcolo la soluzione per il raggio interno
raggio_down= root2(p(1),p(2),p(3)-rssi_fit(idx2)-1.5*delta(idx2));
%calcolo la soluzione per il raggio effettivo
raggio_cent= PALL_sorted.S0(idx2)
```

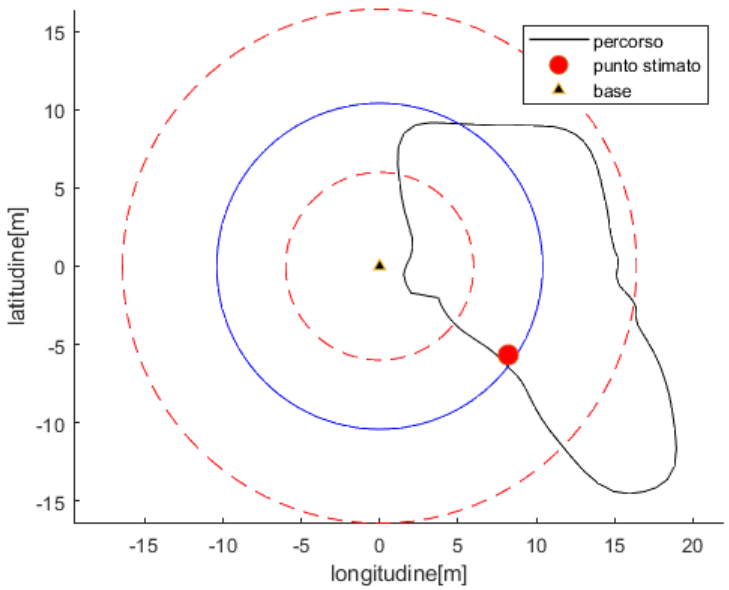
```
raggio_cent = 10.4113
```

```
%distanza in line d'aria
distanza_linea_d_aria = abs(P4.S0(h)-raggio_cent)
```

```
distanza_linea_d_aria = 0.6273
```

```
figure;
hold on;
plot(longp,latp,'-k','linewidth',1);
% Plotta il cerchio
```

```
 rectangle('Position', [-raggio_up, -raggio_up, 2*raggio_up, 2*raggio_up],
'Curvature', [1 1],'EdgeColor', 'r','LineStyle','--');
 rectangle('Position', [-raggio_down, -raggio_down, 2*raggio_down,
2*raggio_down], 'Curvature', [1 1],'EdgeColor', 'r','LineStyle','--');
 rectangle('Position', [-raggio_cent, -raggio_cent, 2*raggio_cent,
2*raggio_cent], 'Curvature', [1 1],'EdgeColor', 'b');
 plot(P4.lon_m(h),P4.lat_m(h),'o', 'MarkerFaceColor', 'red', 'MarkerSize',
10);
 plot(0,0,'^', 'MarkerFaceColor', 'black', 'MarkerSize', 5);
 ylabel('latitudine[m]'); xlabel('longitudine[m]');
 legend('percorso','punto stimato','base');
 axis equal
 hold off;
```



Test 2: Iterato

```
 for n=1:length(P4.id)
 [~,idx2]=min(abs(rssi_fit-P4.rssi(n)));
 raggio_cent= PALL_sorted.S0(idx2);
 distanza_linea_d_aria(n) = abs(P4.S0(n)-raggio_cent);
 end

 distanza_media=mean(distanza_linea_d_aria)
```

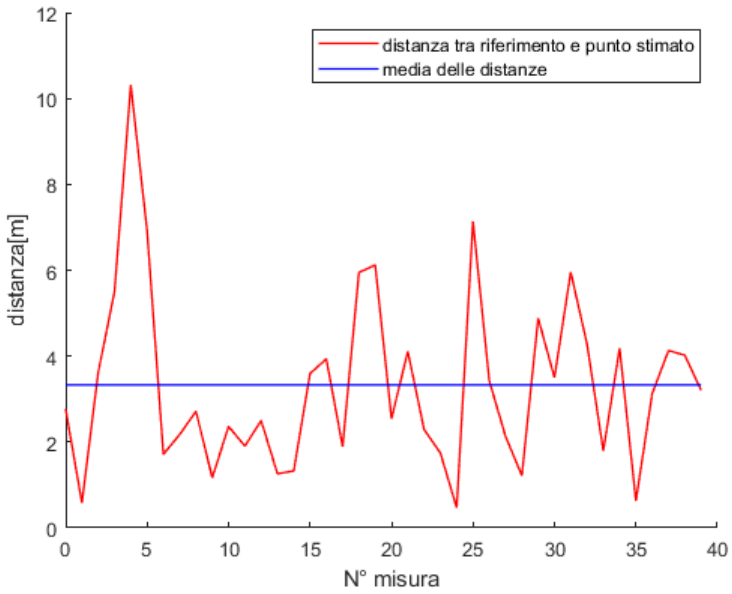
```
distanza_media = 3.3279
```

```
 distanza_dev=std(distanza_linea_d_aria)
```

```
distanza_dev = 2.0734
```

```
 figure;
 hold on;
 plot((0:1:length(P4.S)-1), distanza_linea_d_aria,'r','linewidth',1);
```

```
 plot((0:1:length(P4.S)-1),
(distanza_media)*ones(length(P4.S)),'b','linewidth',1);
 ylabel('distanza[m]'); xlabel('N° misura');
 legend('distanza tra riferimento e punto stimato','media delle distanze');
 hold off;
```



Test 3: base

```
 h=34; %indice dell cella contenente il valore di RSSI da testare scelto
casualmente

 X = [];                    %array di dati validi
 RSSI_test= P4.rssi(h)      %RSSI attuale
```

```
RSSI_test = -75
```

```
 RSSIprev=P4.rssi(h-1)      %RSSI precedente
```

```
RSSIprev = -80
```

```
 distanza_reale= P4.S(h)
```

```
distanza_reale = 57.1807
```

```
 tolleranza1=0;              %tolleranza RSSI attuale
 tolleranza2=0;              %tolleranza RSSI precedente
 x_max=0;                    %tolleranza metri
 index = find(abs(sensore-RSSI_test)< tolleranza1);
 index_prev = find(abs(sensore-RSSIprev)< tolleranza2);
 %ciclo per trovare il primo dato RSSI Attuale
 while(length(index)<1)
     tolleranza1=tolleranza1+0.5;
     index = find(abs(sensore-RSSI_test)<= tolleranza1);
```

```
 end
 %ciclo per trovare il primo dato RSSI precedente
 while(length(index_prev)<1)
     tolleranza2=tolleranza2+0.7;
     index_prev = find(abs(sensore-RSSIprev)<= tolleranza2);
 end

 dati_selezionati = [];
 %fino a che non trovo un dato valido esegue il ciclo
 while(isempty(X))
     x_max=0;
     while(isempty(X) && x_max<=3)
         x_hat=S_interp(index);              %distanza attuale
         x_prev=S_interp(index_prev);        %distanza precedente
         th1=x_prev-x_max;
         th2=x_prev+x_max;
         X = [];
         for n = 1:length(th1)
             dati_selezionati = x_hat(x_hat >= th1(n) & x_hat <= th2(n));
             X = [X, dati_selezionati];
         end
         x_max=x_max+1.5;
     end
     tolleranza2=tolleranza2+0.5;
     tolleranza1=tolleranza1+0.7;
     index_prev = find(abs(sensore-RSSIprev)<= tolleranza2);
     index = find(abs(sensore-RSSI_test)<= tolleranza1);
 end
 %calcolo della media tra i davi validi, trovando la distanza
 distanza=mean(X)
```

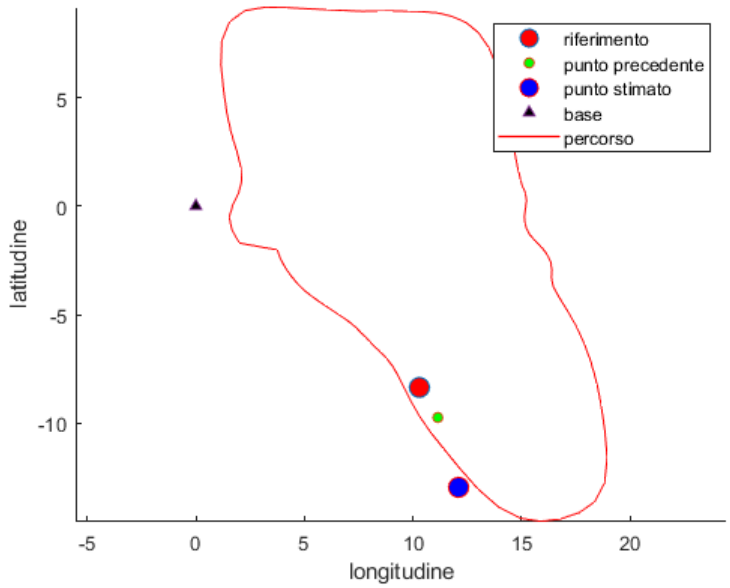
```
distanza = 54
```

```
 %differenza in metri tra punto stimato e reale
 metri_su_percorso=abs(distanza-distanza_reale);
 if(metri_su_percorso>max(PALL.S)/2)
     metri_su_percorso=abs(max(PALL.S)-metri_su_percorso);
 end
 metri_su_percorso
```

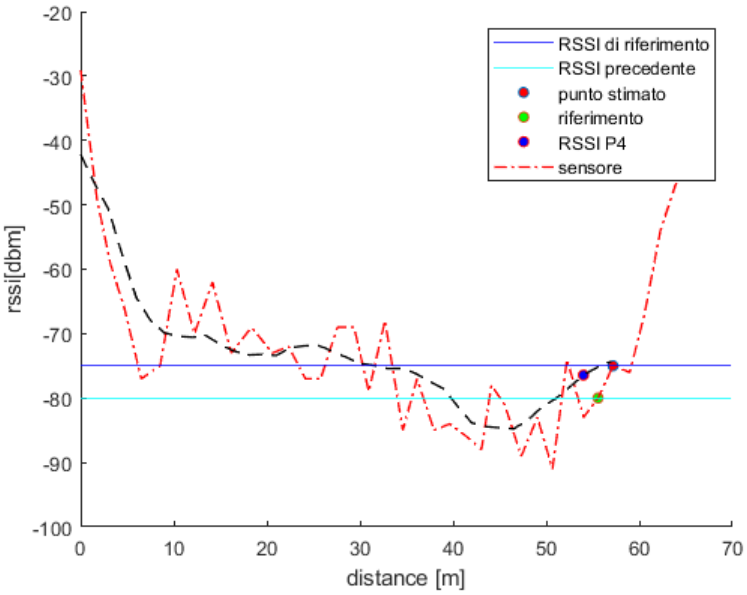
```
metri_su_percorso = 3.1807
```

```
 [~,idx1]=min(abs(PALL.S-distanza));   %idx1: punto con coordinate a distanza
più vicina
 [~,idx2]=min(abs(S_interp-distanza)); %idx2: indice del sensore a distanza
minima

 figure;
 hold on;
 plot(P4.lon_m(h),P4.lat_m(h),'o', 'MarkerFaceColor', 'red', 'MarkerSize',
10);
```

```
 plot(P4.lon_m(h-1),P4.lat_m(h-1),'o', 'MarkerFaceColor', 'green',
'MarkerSize', 5);
 plot(PALL.lon(idx1),PALL.lat(idx1),'ro','MarkerFaceColor', 'blue',
'MarkerSize', 10);
 plot(0,0,'^', 'MarkerFaceColor', 'black', 'MarkerSize', 5);
 plot(longp,latp,'-r','linewidth',1);
 ylabel('latitudine'); xlabel('longitudine');
 legend('riferimento','punto precedente','punto stimato','base','percorso');
 axis equal
 hold off;
```




```
 figure;
 hold on;
 yline(RSSI_test,'b');
 yline(RSSIprev,'c');
 plot(P4.S(h),P4.rssi(h),'o', 'MarkerFaceColor', 'red', 'MarkerSize', 5);
 plot(P4.S(h-1),P4.rssi(h-1),'o', 'MarkerFaceColor', 'green', 'MarkerSize',
5);
 plot(S_interp(idx2),sensore(idx2),'ro','MarkerFaceColor', 'blue',
'MarkerSize', 5);
 plot(P4.S,P4.rssi,'-.r','linewidth',1);
 plot(S_interp, sensore,'--k','linewidth',1);
 ylabel('rssi[dbm]'); xlabel('distance [m]');
 legend('RSSI di riferimento','RSSI precedente','punto
stimato','riferimento','RSSI P4', 'sensore');
 hold off;
```



Test 3 iterato

```
dati=table(); %tabella con tutti i dati del sensore
f=1;
for h=2:length(P4.rssi)
    X = [];                     %array di dati validi
    RSSI_test= P4.rssi(h);      %RSSI attuale
    RSSIprev=P4.rssi(h-1);      %RSSI precedente
    distanza_reale= P4.S(h);
    tolleranza1=0;              %tolleranza RSSI attuale
    tolleranza2=0;              %tolleranza RSSI precedente
    x_max=0;                    %tolleranza metri
    index = find(abs(sensore-RSSI_test)< tolleranza1);
    index_prev = find(abs(sensore-RSSIprev)< tolleranza2);
    %ciclo per trovare il primo dato RSSI Attuale
    while(length(index)<1)
        tolleranza1=tolleranza1+0.5;
        index = find(abs(sensore-RSSI_test)<= tolleranza1);
    end
    %ciclo per trovare il primo dato RSSI precedente
    while(length(index_prev)<1)
        tolleranza2=tolleranza2+0.7;
        index_prev = find(abs(sensore-RSSIprev)<= tolleranza2);
    end
    dati_selezionati = [];
    %fino a che non trovo un dato valido esegue il ciclo
    while(isempty(X))
        x_max=0;
        while(isempty(X) && x_max<=3)
            x_hat=S_interp(index);              %distanza attuale
            x_prev=S_interp(index_prev);        %distanza precedente
            th1=x_prev-x_max;                   %range di ricerca
```

```
            th2=x_prev+x_max;
            X = [];
            for n = 1:length(th1)
                dati_selezionati = x_hat(x_hat >= th1(n) & x_hat <= th2(n));
                X = [X, dati_selezionati];
            end
            x_max=x_max+1.5;
        end
        tolleranza2=tolleranza2+0.5;
        tolleranza1=tolleranza1+0.9;
        index_prev = find(abs(sensore-RSSIprev)<= tolleranza2);
        index = find(abs(sensore-RSSI_test)<= tolleranza1);
    end
    %calcolo della media tra i davi validi
    distanza=mean(X);
    %differenza in metri tra punto stimato e reale
    metri_su_percorso=abs(distanza-distanza_reale);
    if(metri_su_percorso>max(PALL.S)/2)
        metri_su_percorso=abs(max(PALL.S)-metri_su_percorso);
    end
    %tabella di dati del sensore
    dati.h(f)=h;
    dati.distanza(f)=distanza;
    dati.differenza(f)=metri_su_percorso;
    f=f+1;
end
media=mean(dati.differenza)
```

```
media = 6.2493
```

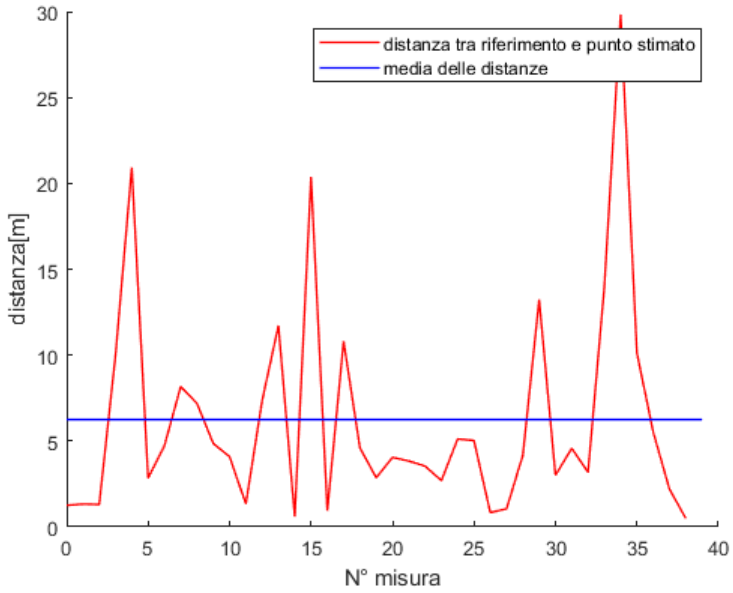
```
deviazione_standard=std(dati.differenza)
```

```
deviazione_standard = 6.3340
```

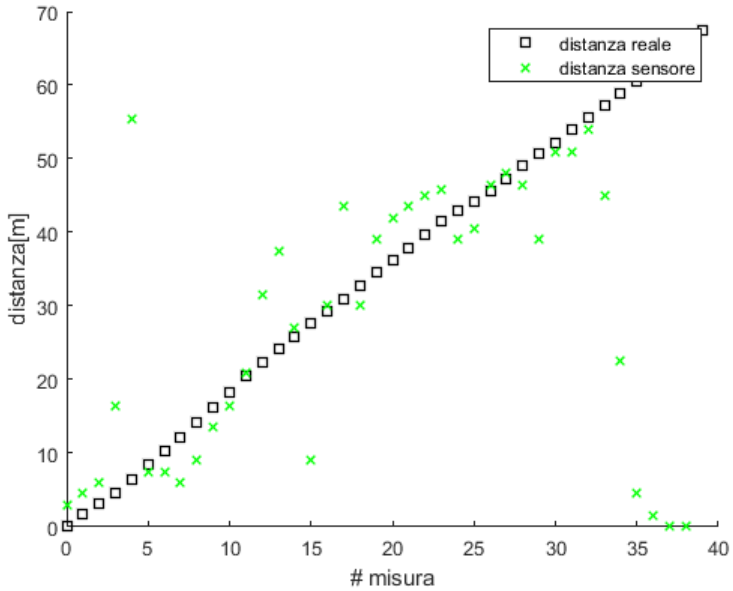
```
dati_validi=dati(dati.differenza<11, :);  %considero validi solo i dati
inferiori a 11m
perc_riuscita=(length(dati_validi.h)/length(P4.id))*100
```

```
perc_riuscita = 82.5000
```

```
figure;
hold on;
plot((0:1:length(P4.S)-2), dati.differenza,'r','linewidth',1);
plot((0:1:length(P4.S)-1), (media)*ones(length(P4.S)),'b','linewidth',1);
ylabel('distanza[m]'); xlabel('N° misura');
legend('distanza tra riferimento e punto stimato','media delle distanze');
hold off;
```




```
figure;
hold on;
plot((0:1:length(P4.S)-1), P4.S,'sk','linewidth',1);
plot((0:1:length(dati.differenza)-1), dati.distanza,'xg','linewidth',1);
ylabel('distanza[m]'); xlabel('# misura');
legend('distanza reale','distanza sensore');
hold off;
```



Funzioni esterne

Calcolo della radice reale di un polinomio di secondo grado.

```
function [r_real] = root2(x2,x1,x0)
p = [x2 x1 x0];
```

```
r = roots(p);
r_real1 = r(imag(r)==0 & r>0 & r<30);
 r_real = min(r_real1);
if isempty(r_real)
        r_real = 0;
end
```

Calcolo della distanza in linea d'aria tra due coordinate.

```
function [c] = distanza_aerea(latitude_i,longitude_i,latitude_f,longitude_f)
q=1/180;
 R=6372.795477598*1000;  %raggio della terra
dlat = (latitude_f-latitude_i)*q;
dlon = (longitude_f-longitude_i)*q;
lat1 =latitude_i*q;
lat2 = latitude_f*q;
a = (sin(dlat/2))^2 + cos(lat1)* cos(lat2)* (sin(dlon/2))^2;
c = 2*R*asin(sqrt(a));
end
```

Calcolo della lunghezza di un percorso tra punto iniziale e coordinate.

```
function [k] = distanza(N,latitude,longitude)
q=1/180;
%Calcolo distanza percorsa tra una misura e la successiva, in linea d'aria.
%distanza (A,B) = R * arccos(sin(latA) * sin(latB) + cos(latA) * cos(latB) *
cos(lonA-lonB))
%angoli in radianti, R = 6372,795,477598 Km
 R=6372.795477598*1000; k(1)=0;
for n=2:N
dlat = (latitude(n)-latitude(n-1))*q;
dlon = (longitude(n)-longitude(n-1))*q;
lat1 =latitude(n-1)*q;
lat2 = latitude(n)*q;
a = (sin(dlat/2))^2 + cos(lat1)* cos(lat2)* (sin(dlon/2))^2;
c = 2*R*asin(sqrt(a));
k(n)= k(n-1)+c;
end
```

Conversione latitudine e longitudine in metri.

```
function [c] = lat2m(latitude_f)
q=1/180;
 R=6372.795477598*1000;
dlat = (latitude_f)*q;
c = 2*R*asin(sin(dlat/2));
```

```
function [c] = lon2m(longitude_f)
q=1/180;
R=6372.795477598*1000;
dlon = (longitude_f)*q;
c = 2*R*asin(sin(dlon/2));
```