



## UNIVERSITÀ DI PISA

UNIVERSITÀ DEGLI STUDI DI PISA  
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE  
Corso di laurea in Ingegneria Robotica e dell'Automazione

# Studio e caratterizzazione del sistema Pozyx

Progetto di Sistemi di guida e navigazione

Candidati:  
**Cristian Tesconi**  
**Luca Crosato**

A.A. 2017-2018

# Indice

|                                                                          |           |
|--------------------------------------------------------------------------|-----------|
| <b>1 Obiettivi del progetto</b>                                          | <b>6</b>  |
| <b>2 Introduzione</b>                                                    | <b>6</b>  |
| 2.1 Il sistema Pozyx . . . . .                                           | 6         |
| 2.1.1 Dove collocare le ancore (regole empiriche) . . . . .              | 7         |
| 2.1.2 Setup . . . . .                                                    | 8         |
| 2.2 Dispositivo seriale e dispositivo remoto . . . . .                   | 8         |
| 2.3 Parametri ultra-wideband . . . . .                                   | 9         |
| 2.4 Accorgimenti . . . . .                                               | 9         |
| 2.5 Caratterizzazione di un fallimento nel positioning . . . . .         | 10        |
| <b>3 Aspetti teorici</b>                                                 | <b>11</b> |
| 3.1 Algoritmo di autocalibrazione . . . . .                              | 11        |
| 3.2 Algoritmo RANSAC . . . . .                                           | 12        |
| 3.2.1 Vantaggi e svantaggi RANSAC . . . . .                              | 12        |
| 3.2.2 Implementazione Python e risultati sperimentali . . . . .          | 13        |
| 3.3 Propagazione dell'errore . . . . .                                   | 16        |
| 3.3.1 Applicazione alla calibrazione manuale . . . . .                   | 17        |
| <b>4 Esperimento 1: verifica di bias</b>                                 | <b>18</b> |
| 4.1 Conclusioni . . . . .                                                | 21        |
| <b>5 Esperimento 2: auto-calibrazione 2D</b>                             | <b>22</b> |
| 5.1 Descrizione dell'esperimento . . . . .                               | 22        |
| 5.2 Risultati . . . . .                                                  | 22        |
| 5.3 Nuova configurazione . . . . .                                       | 23        |
| <b>6 Esperimento 3: autocalibrazione 3D mediante webapp</b>              | <b>25</b> |
| 6.1 Conclusioni . . . . .                                                | 26        |
| <b>7 Esperimento 4: autocalibrazione mediante algoritmo algebrico</b>    | <b>27</b> |
| 7.1 Conclusioni . . . . .                                                | 27        |
| <b>8 Esperimento 5: posizionamento tag</b>                               | <b>29</b> |
| 8.1 Conclusioni . . . . .                                                | 32        |
| <b>9 Esperimento 6: parametri UWB e frequenza di lavoro</b>              | <b>34</b> |
| 9.1 Conclusioni . . . . .                                                | 34        |
| <b>10 Esperimento 7: Multipositioning</b>                                | <b>39</b> |
| <b>11 Esperimento 8: massima distanza di comunicazione doPositioning</b> | <b>41</b> |
| 11.1 Descrizione dell'esperimento . . . . .                              | 41        |
| 11.2 Caratterizzazione di un fallimento nel positioning . . . . .        | 41        |
| 11.3 Plot grafici e tabelle dei risultati . . . . .                      | 41        |
| <b>12 Esperimento 9: test outdoor</b>                                    | <b>46</b> |
| 12.1 Plot grafici e tabelle dei risultati . . . . .                      | 47        |
| <b>A Script di base</b>                                                  | <b>58</b> |
| A.1 autocalibration.py . . . . .                                         | 58        |
| A.2 Descrizione procedura di autocalibrazione . . . . .                  | 59        |
| A.3 Autocalibrazione senza RANSAC . . . . .                              | 60        |
| A.4 Algoritmo RANSAC . . . . .                                           | 64        |
| A.5 Autocalibrazione con RANSAC . . . . .                                | 68        |
| A.6 Positioning . . . . .                                                | 77        |
| A.7 Calibrazione e positioning . . . . .                                 | 81        |
| <b>B Codice esperimento 1</b>                                            | <b>82</b> |

|                                          |            |
|------------------------------------------|------------|
| <b>C Codice esperimento 2</b>            | <b>84</b>  |
| <b>D Codice esperimento 4</b>            | <b>86</b>  |
| <b>E Codice esperimento 5</b>            | <b>87</b>  |
| <b>F Codice esperimento 6</b>            | <b>92</b>  |
| <b>G Codice propagazione dell'errore</b> | <b>100</b> |

## Elenco delle figure

|    |                                                                                                                                                                                                                                                                                                                           |    |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1  | Diagramma del Pozyx tag . . . . .                                                                                                                                                                                                                                                                                         | 6  |
| 2  | Regola 2 . . . . .                                                                                                                                                                                                                                                                                                        | 7  |
| 3  | Regola 3 . . . . .                                                                                                                                                                                                                                                                                                        | 7  |
| 4  | Applicazione dell'algoritmo RANSAC sui dati grezzi di autoranging: prova 1 ( $n=20\%$ ,<br>$k=10^3, t = 10^5, d = 60\%$ ) . . . . .                                                                                                                                                                                       | 13 |
| 5  | Applicazione dell'algoritmo RANSAC sui dati grezzi di autoranging: prova 2 ( $n=20\%$ ,<br>$k=10^4, t = 5 \cdot 10^4, d = 60\%$ ) . . . . .                                                                                                                                                                               | 14 |
| 6  | Applicazione dell'algoritmo RANSAC sui dati grezzi di autoranging: prova 3 ( $n=10\%$ ,<br>$k=2 \cdot 10^5, t = 10^4, d = 30\%$ ) . . . . .                                                                                                                                                                               | 14 |
| 7  | Applicazione dell'algoritmo RANSAC sui dati grezzi di autoranging: prova 3 ( $n=10\%$ ,<br>$k=2 \cdot 10^5, t = 2500, d = 30\%$ ) . . . . .                                                                                                                                                                               | 15 |
| 8  | Propagazione dell'errore di misura . . . . .                                                                                                                                                                                                                                                                              | 16 |
| 9  | Iistogramma prova 1 . . . . .                                                                                                                                                                                                                                                                                             | 18 |
| 10 | Iistogramma prova 2 . . . . .                                                                                                                                                                                                                                                                                             | 18 |
| 11 | Iistogramma prova 3 . . . . .                                                                                                                                                                                                                                                                                             | 19 |
| 12 | Iistogramma prova 4 . . . . .                                                                                                                                                                                                                                                                                             | 19 |
| 13 | Iistogramma prova 5 . . . . .                                                                                                                                                                                                                                                                                             | 19 |
| 14 | Iistogramma prova 6 . . . . .                                                                                                                                                                                                                                                                                             | 19 |
| 15 | Iistogramma prova 7 . . . . .                                                                                                                                                                                                                                                                                             | 19 |
| 16 | Iistogramma prova 8 . . . . .                                                                                                                                                                                                                                                                                             | 19 |
| 17 | Iistogramma prova 9 . . . . .                                                                                                                                                                                                                                                                                             | 19 |
| 18 | Iistogramma prova 1_1 . . . . .                                                                                                                                                                                                                                                                                           | 20 |
| 19 | Iistogramma prova 2_1 . . . . .                                                                                                                                                                                                                                                                                           | 20 |
| 20 | Iistogramma prova 3_1 . . . . .                                                                                                                                                                                                                                                                                           | 20 |
| 21 | Iistogramma prova 4_1 . . . . .                                                                                                                                                                                                                                                                                           | 20 |
| 22 | Iistogramma prova 5_1 . . . . .                                                                                                                                                                                                                                                                                           | 20 |
| 23 | Fit lineare (relativo alla Tabella 1) . . . . .                                                                                                                                                                                                                                                                           | 21 |
| 24 | Esperimento 3: gruppi delle misure relative alle 3 ancore. Le misure sono espresse<br>in mm. I numeri indicano a quale ancora corrispondono i gruppi di dati, i colori<br>corrispondono alla configurazioni corrispondenti. La configurazione verde è quella<br>corrispondente alla reale posizione delle ancore. . . . . | 22 |
| 25 | Esperimento 3: gruppi delle misure relative alle 3 ancore. Le misure sono espresse<br>in mm. I numeri indicano a quale ancora corrispondono i gruppi di dati, i colori<br>corrispondono alla configurazioni corrispondenti. La configurazione verde è quella<br>corrispondente alla reale posizione delle ancore. . . . . | 23 |
| 26 | Calibrazione mediante app . . . . .                                                                                                                                                                                                                                                                                       | 25 |
| 27 | Calibrazione mediante app: vista frontale . . . . .                                                                                                                                                                                                                                                                       | 25 |
| 28 | Posizione ancore, vista in pianta. . . . .                                                                                                                                                                                                                                                                                | 29 |
| 29 | Posizione ancore 3D. . . . .                                                                                                                                                                                                                                                                                              | 30 |
| 30 | Posizione tag, vista in pianta. . . . .                                                                                                                                                                                                                                                                                   | 30 |
| 31 | Posizione tag 3D. . . . .                                                                                                                                                                                                                                                                                                 | 31 |
| 32 | Posizione tag e ancore, vista in pianta. . . . .                                                                                                                                                                                                                                                                          | 31 |
| 33 | Posizione tag e ancore 3D. . . . .                                                                                                                                                                                                                                                                                        | 32 |
| 34 | Coordinata x del tag, calibrazione automatica. . . . .                                                                                                                                                                                                                                                                    | 32 |
| 35 | Coordinata x del tag, calibrazione manuale. . . . .                                                                                                                                                                                                                                                                       | 32 |
| 36 | Coordinata y del tag, calibrazione automatica. . . . .                                                                                                                                                                                                                                                                    | 33 |
| 37 | Coordinata y del tag, calibrazione manuale. . . . .                                                                                                                                                                                                                                                                       | 33 |
| 38 | Coordinata z del tag, calibrazione automatica. . . . .                                                                                                                                                                                                                                                                    | 33 |
| 39 | Coordinata z del tag, calibrazione manuale. . . . .                                                                                                                                                                                                                                                                       | 33 |
| 40 | Durata del positioning: PROTOCOL PRECISION, canale 5 . . . . .                                                                                                                                                                                                                                                            | 35 |
| 41 | Durata del positioning: PROTOCOL FAST, canale 5 . . . . .                                                                                                                                                                                                                                                                 | 35 |
| 42 | Durata del positioning: PROTOCOL PRECISION canale 1 . . . . .                                                                                                                                                                                                                                                             | 37 |
| 43 | Durata del positioning: PROTOCOL FAST, canale 1 . . . . .                                                                                                                                                                                                                                                                 | 37 |
| 44 | Traiettoria seguita dall'asta . . . . .                                                                                                                                                                                                                                                                                   | 39 |
| 45 | Iistogramma della distanza fra i due tag . . . . .                                                                                                                                                                                                                                                                        | 39 |
| 46 | Iistogramma distanza relativa x . . . . .                                                                                                                                                                                                                                                                                 | 40 |

|     |                                                                                  |    |
|-----|----------------------------------------------------------------------------------|----|
| 47  | Iistogramma distanza relativa y . . . . .                                        | 40 |
| 48  | Iistogramma distanza relativa z . . . . .                                        | 40 |
| 49  | Iistogramma coordinata x del tag: distanza 5 m . . . . .                         | 42 |
| 50  | Iistogramma coordinata y del tag: distanza 5 m . . . . .                         | 42 |
| 51  | Iistogramma coordinata z del tag: distanza 5 m . . . . .                         | 42 |
| 52  | Iistogramma coordinata x del tag: distanza 10 m . . . . .                        | 43 |
| 53  | Iistogramma coordinata y del tag: distanza 10 m . . . . .                        | 43 |
| 54  | Iistogramma coordinata z del tag: distanza 10 m . . . . .                        | 43 |
| 55  | Iistogramma coordinata x del tag: distanza 15 m . . . . .                        | 43 |
| 56  | Iistogramma coordinata y del tag: distanza 15 m . . . . .                        | 43 |
| 57  | Iistogramma coordinata z del tag: distanza 15 m . . . . .                        | 43 |
| 58  | Iistogramma coordinata x del tag: distanza 20 m . . . . .                        | 44 |
| 59  | Iistogramma coordinata y del tag: distanza 20 m . . . . .                        | 44 |
| 60  | Iistogramma coordinata z del tag: distanza 20 m . . . . .                        | 44 |
| 61  | Iistogramma coordinata x del tag: distanza 25 m . . . . .                        | 44 |
| 62  | Iistogramma coordinata y del tag: distanza 25 m . . . . .                        | 44 |
| 63  | Iistogramma coordinata z del tag: distanza 25 m . . . . .                        | 44 |
| 64  | Iistogramma coordinata x del tag: distanza 30 m . . . . .                        | 45 |
| 65  | Iistogramma coordinata y del tag: distanza 30 m . . . . .                        | 45 |
| 66  | Iistogramma coordinata z del tag: distanza 30 m . . . . .                        | 45 |
| 67  | Iistogramma coordinata x del tag: distanza 35 m . . . . .                        | 45 |
| 68  | Iistogramma coordinata y del tag: distanza 35 m . . . . .                        | 45 |
| 69  | Iistogramma coordinata z del tag: distanza 35 m . . . . .                        | 45 |
| 70  | Iistogramma coordinata x del tag: distanza 40 m . . . . .                        | 46 |
| 71  | Iistogramma coordinata y del tag: distanza 40 m . . . . .                        | 46 |
| 72  | Iistogramma coordinata z del tag: distanza 40 m . . . . .                        | 46 |
| 73  | Iistogramma coordinata x del tag: distanza 5 m . . . . .                         | 47 |
| 74  | Iistogramma coordinata y del tag: distanza 5 m . . . . .                         | 47 |
| 75  | Iistogramma coordinata z del tag: distanza 5 m . . . . .                         | 47 |
| 76  | Iistogramma coordinata x del tag: distanza 10 m . . . . .                        | 48 |
| 77  | Iistogramma coordinata y del tag: distanza 10 m . . . . .                        | 48 |
| 78  | Iistogramma coordinata z del tag: distanza 10 m . . . . .                        | 48 |
| 79  | Iistogramma coordinata x del tag: distanza 15 m . . . . .                        | 48 |
| 80  | Iistogramma coordinata y del tag: distanza 15 m . . . . .                        | 48 |
| 81  | Iistogramma coordinata z del tag: distanza 15 m . . . . .                        | 48 |
| 82  | Iistogramma coordinata x del tag: distanza 20 m . . . . .                        | 49 |
| 83  | Iistogramma coordinata y del tag: distanza 20 m . . . . .                        | 49 |
| 84  | Iistogramma coordinata z del tag: distanza 20 m . . . . .                        | 49 |
| 85  | Iistogramma coordinata x del tag: distanza 25 m . . . . .                        | 49 |
| 86  | Iistogramma coordinata y del tag: distanza 25 m . . . . .                        | 49 |
| 87  | Iistogramma coordinata z del tag: distanza 25 m . . . . .                        | 49 |
| 88  | Iistogramma coordinata x del tag: distanza 30 m . . . . .                        | 50 |
| 89  | Iistogramma coordinata y del tag: distanza 30 m . . . . .                        | 50 |
| 90  | Iistogramma coordinata z del tag: distanza 30 m . . . . .                        | 50 |
| 91  | Iistogramma coordinata x del tag: distanza 40 m . . . . .                        | 50 |
| 92  | Iistogramma coordinata y del tag: distanza 40 m . . . . .                        | 50 |
| 93  | Iistogramma coordinata z del tag: distanza 40 m . . . . .                        | 50 |
| 94  | Iistogramma coordinata x del tag: distanza 45 m . . . . .                        | 51 |
| 95  | Iistogramma coordinata y del tag: distanza 45 m . . . . .                        | 51 |
| 96  | Iistogramma coordinata z del tag: distanza 45 m . . . . .                        | 51 |
| 97  | Iistogramma coordinata x del tag: distanza 50 m, orientazione 0 gradi . . . . .  | 52 |
| 98  | Iistogramma coordinata y del tag: distanza 50 m, orientazione 0 gradi . . . . .  | 52 |
| 99  | Iistogramma coordinata z del tag: distanza 50 m, orientazione 0 gradi . . . . .  | 52 |
| 100 | Iistogramma coordinata x del tag: distanza 50 m, orientazione 90 gradi . . . . . | 53 |
| 101 | Iistogramma coordinata y del tag: distanza 50 m, orientazione 90 gradi . . . . . | 53 |
| 102 | Iistogramma coordinata z del tag: distanza 50 m, orientazione 90 gradi . . . . . | 53 |
| 103 | Iistogramma coordinata x del tag: distanza 55 m . . . . .                        | 53 |
| 104 | Iistogramma coordinata y del tag: distanza 55 m . . . . .                        | 53 |

|     |                                                                                  |    |
|-----|----------------------------------------------------------------------------------|----|
| 105 | Iistogramma coordinata x del tag: distanza 60 m, orientazione 0 gradi . . . . .  | 54 |
| 106 | Iistogramma coordinata y del tag: distanza 60 m, orientazione 0 gradi . . . . .  | 54 |
| 107 | Iistogramma coordinata z del tag: distanza 60 m, orientazione 0 gradi . . . . .  | 54 |
| 108 | Iistogramma coordinata x del tag: distanza 60 m, orientazione 90 gradi . . . . . | 55 |
| 109 | Iistogramma coordinata y del tag: distanza 60 m, orientazione 90 gradi . . . . . | 55 |
| 110 | Iistogramma coordinata z del tag: distanza 60 m, orientazione 90 gradi . . . . . | 55 |
| 111 | Iistogramma coordinata x del tag: distanza 65 m . . . . .                        | 56 |
| 112 | Iistogramma coordinata y del tag: distanza 65 m . . . . .                        | 56 |
| 113 | Iistogramma coordinata z del tag: distanza 65 m . . . . .                        | 56 |
| 114 | Iistogramma coordinata x del tag: distanza 70 m . . . . .                        | 56 |
| 115 | Iistogramma coordinata y del tag: distanza 70 m . . . . .                        | 56 |
| 116 | Iistogramma coordinata z del tag: distanza 70 m . . . . .                        | 56 |
| 117 | Traiettoria seguita in blu, ancore in rosso. Vista in pianta . . . . .           | 57 |
| 118 | Positioning a varie distanze . . . . .                                           | 57 |

# 1 Obiettivi del progetto

L'obiettivo primario è quello di verificare le performance del sistema in termini di accuratezza e frequenza con cui il sistema calcola la posizione del dispositivo mobile.

Vengono effettuati test relativamente al ranging (calcolo della distanza euclidea fra una o più copie di dispositivi), alla calibrazione delle dei dispositivi pozzyx fissi e al positioning (determinazione delle coordinate cartesiane del dispositivo mobile sulla base delle informazioni ottenute dal ranging).

È stato inoltre implementato un algoritmo, nominato RANSAC, finalizzato alla detezione ed eliminazione degli outlier nelle misure di autoranging delle ancore.

È realizzato uno script python volto ad agevolare le operazioni sopra descritte, il quale consente all'utente di effettuare una procedura guidata di calibrazione delle ancore.

I linguaggi di programmazione utilizzati sono *python* e *matlab*.

In appendice viene riportato il codice utilizzato in ciascun test eseguito.

## 2 Introduzione

### 2.1 Il sistema Pozyx

Il Pozyx è un sistema di localizzazione alternativo al GPS, basato su tecnologia ultra-wideband (UWB).

In una configurazione base si utilizzano 4 ancore (dispositivi fissi) e un tag (dispositivo mobile). Basandosi su un algoritmo di triangolazione il tag riesce a localizzarsi rispetto alle ancore. In teoria la tecnologia ultra-wideband consente di avere posizionamento accurato al centimetro anche in ambienti indoor e in presenza di ostacoli di natura non metallica.

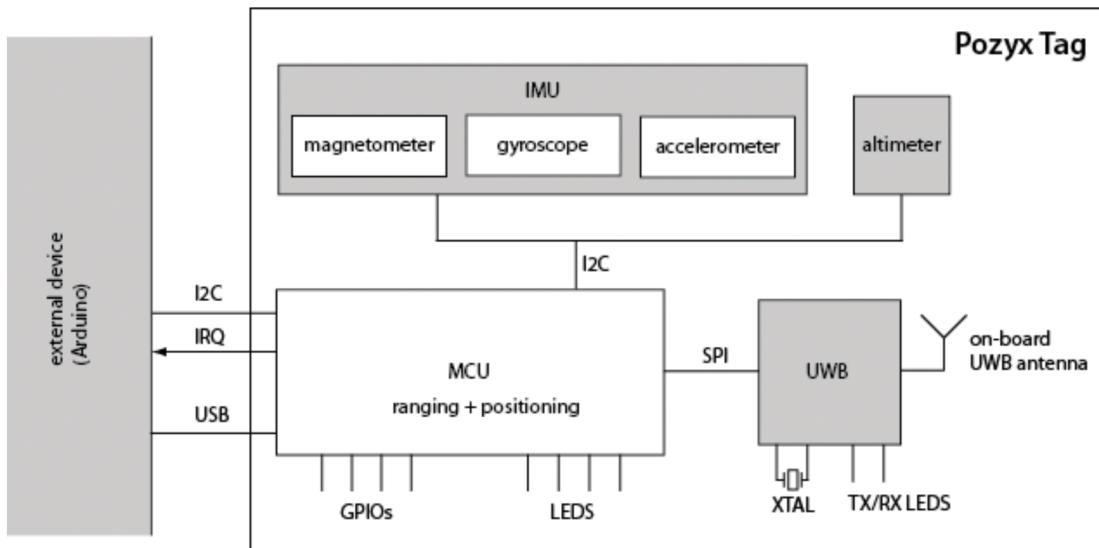


Figura 1: Diagramma del Pozyx tag

In figura 1 è rappresentato il diagramma concettuale dell'hardware del tag. Il cuore della scheda elettronica è il microcontrollore, indicato con la sigla MCU (Microcontroller Unit) che svolge la maggior parte delle funzionalità, come gli algoritmi di triangolazione, auto-calibrazione e ranging. Inoltre è responsabile della gestione dell'interfaccia con dispositivi esterni (come Arduino), con i quali comunica attraverso un protocollo USB o I2C e una linea di interrupt. A bordo del tag sono inoltre equipaggiati svariati sensori: un magnetometro, un giroscopio, un accelerometro e un sensore di pressione (altimetro). I dati dei sensori vengono utilizzati dal microcontrollore per la triangolazione e sono disponibili all'utente. Naturalmente è anche presente un chip munito di antenna UWB che è fondamentale per avere le misurazioni di distanza necessarie a far funzionare gli algoritmi di posizionamento. Infine precisiamo che la differenza tra un ancora ed

un tag consiste nella presenza dei sensori di moto, che risultano superflui per l'ancora in quanto quest'ultime devono rimanere fisse. Eventualmente è pertanto possibile utilizzare un dispositivo o come ancora o come tag. Se si utilizza un'ancora come tag si però deve rinunciare all'ausilio dei sensori di moto.

### 2.1.1 Dove collocare le ancore (regole empiriche)

- Posizionare le ancore in alto e sulla line-of-sight dell'utente; questa prima regola è semplice: il posizionamento in alto (sul soffitto o sui muri) aumenta la possibilità di ricevere un buon segnale perché ci sono meno ostacoli. Le ostruzioni hanno generalmente un influenza negativa sulle misurazioni dei range e di conseguenza sul posizionamento.
- Non collocare le ancore sulla stessa linea; è consigliato distribuire le ancore in modo che esse coprano tutte le direzioni. Nel caso in cui gli ancoraggi sono su una linea retta, l'errore di posizionamento sarà molto grande. Questo può essere visto nella figura 2.

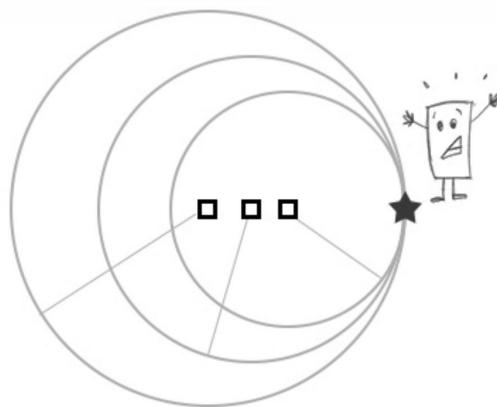


Figura 2: Regola 2

- Posizionare le ancore verticalmente con l'antenna UWB rivolta verso l'alto come in figura 3;

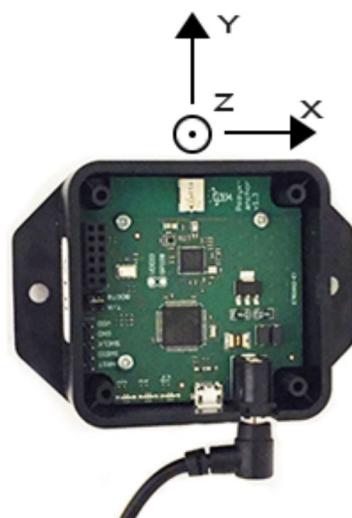


Figura 3: Regola 3

- Per un posizionamento 3D posizionare le ancore ad altezze differenti; se infatti si usano le ancore tutte nello stesso piano orizzontale non sarà possibile avere una buona accuratezza sull'altezza.

### 2.1.2 Setup

1. Scaricare l'ultima versione del firmware del software Pozyx (<http://tray.cloud.pozyxlabs.com/>)
2. Alla fine dell'installazione sarà possibile effettuare il login; nel caso in cui l'utente non sia ancora registrato, è possibile creare un account a quell'istante.
3. A questo punto è richiesto aggiornare il firmware (si rimanda al seguente link per una descrizione più dettagliata: [https://www.pozyx.io/Documentation/firmware\\_update](https://www.pozyx.io/Documentation/firmware_update)).
4. Collega il tag o un'ancora al proprio PC mediante un cavo USB.
5. Adesso è possibile aprire la web app collegandosi al seguente link: (<https://bapp.cloud.pozyxlabs.com/login>) oppure utilizzando l'app precedentemente installata sul PC.
6. Effettuare il log-in ed iniziare ad interagire con il sistema. Sul sito pozzyx sono presenti alcuni tutorial utili per prendere confidenza con il sistema stesso.
7. Installazione librerie:
  1. Su Windows, se non si dispone di un ambiente di sviluppo basato su python si consiglia di scaricare Pyzo all'indirizzo: <https://pyzo.org/start.html>:
  2. Effettuare eventualmente l'upgrade di pip con il comando `pip install --upgrade pip`, la libreria pypozyx è scritta in python3 pertanto è necessario utilizzare pip3;
  3. Installare le seguenti librerie (si suggerisce il comando `pip3 install mylibrary`):
    - numpy
    - scipy
    - matplotlib
    - pypozyx
  4. Su Linux, se si hanno problemi con l'installazione di Matplotlib dopo l'aggiornamento di pip3 provare a modificare il file `/usr/bin/pip` in:

```
import sys
from pip import __main__
if __name__ == '__main__':
    sys.exit(__main__.main())
```
  8. Se si desidera effettuare una calibrazione automatica del sistema, è presente uno script da noi creato che si chiama `autocalibration.py`, presente nella cartella `/Script` di base.

## 2.2 Dispositivo seriale e dispositivo remoto

In una normale configurazione sono presenti dispositivi collegati semplicemente all'alimentazione e un dispositivo collegato alla porta seriale del PC. Questo dispositivo è un dispositivo privilegiato, in quanto in grado di comunicare direttamente con il PC e quindi di eseguire le istruzioni indicategli da uno script python. Esso verrà chiamato dispositivo seriale e quando si chiamano le funzioni di libreria che lo riguardano occorre prestare particolare attenzione in quanto esse vanno chiamate con l'identificativo `None`. Generalmente le funzioni di libreria prendono in ingresso i seguenti parametri: `destination` e `remote_id`.

Vediamo ad esempio un caso specifico:

```
doRanging(destination, device_range, remote_id=None)
```

Questa funzione effettua una misurazione della distanza tra i dispositivi i cui identificativi sono `remote_id` e `destination`. Il valore di default per `remote_id` è `None`. Quando questo avviene, la distanza viene automaticamente effettuata tra il dispositivo connesso alla seriale del PC e il dispositivo il cui identificativo è `destination`. Se si desidera invece che il ranging venga effettuato tra due diversi dispositivi della rete basterà inserire l'opportuno valore di `remote_id`. Questa logica di funzionamento delle funzioni di libreria ricorre molto spesso ed è importante tenerne di conto quando si creano script Python che vogliono far interagire i dispositivi Pozyx.

## 2.3 Parametri ultra-wideband

La selezione dei parametri ultra-wideband (UWB) si ripercuote tipicamente sull'update rate del Positioning del tag e sulla distanza a cui riescono a comunicare le antenne. È importante sottolineare che dispositivi aventi differenti parametri UWB potrebbero non essere in grado di comunicare tra loro, quindi è importante assicurarsi che tutti i dispositivi utilizzino gli stessi parametri. Questi ultimi sono così definiti:

- **channel**: questo parametro impone il canale UWB. I dispositivi Pozyx possono utilizzare 6 canali UWB indipendenti. Dispositivi su differenti canali non possono comunicare e non interferiscono. I canali aventi un numero identificativo più basso lavorano a frequenze inferiori, ciò risulta in distanze di comunicazione aumentate ma in un update rate più lento;
- **bitrate**: sono possibili tre possibili valori: 110 kbit/sec, 850 kbit/sec, 6810 kbit/sec. Maggiore è il bitrate e più brevi risulteranno i messaggi e quindi la comunicazione risulterà più veloce. Ciò limiterà però le mie capacità in termini di distanze di lavoro tra le antenne;
- **pulse repetition frequency (PRF)**: questo impone ogni quanto avviene l'impulso che è alla base del funzionamento della comunicazione. Sono possibili due valori: 16 MHz e 64 MHz. Gli effetti sulla frequenza di lavoro sono indicati al paragrafo 9;
- **preamble length**: questo parametro impone la lunghezza del preamble che viene posto davanti a ciascun messaggio che si scambiano i dispositivi Pozyx. Sono possibili 8 differenti opzioni. Più è breve il preamble e più brevi risulteranno i messaggi, quindi avrà in generale una frequenza di lavoro più alta, sempre al costo di uno spazio operativo ridotto;
- **gain**: rappresenta l'intensità dell'impulso e può assumere un qualsiasi valore reale tra 0 e 67.1 dB;
- **ranging protocol**: rappresenta il protocollo utilizzato dai dispositivi Pozyx per effettuare il ranging tra due differenti dispositivi. Sono disponibili attualmente due possibilità: RANGING\_PROTOCOL\_FAST e RANGING\_PROTOCOL\_PRECISION;

Quando si settano i parametri UWB occorre chiamare rispettivamente le funzioni `setUWBSettings` e `setUWBChannel`. Occorre modificare per ultimi i parametri del dispositivo seriale, altrimenti questo non sarà più in grado di modificare i rimanenti dispositivi della rete.

I parametri di default all'accensione dei dispositivi sono: channel 5, bitrate 110 kbit/sec, PRF 64 MHz, preamble length 1024 symbols, gain 11.5 dB.

Per salvare permanentemente un set modificato di parametri, in modo tale da non dover cambiare i parametri ad ogni accensione, occorrerebbe chiamare la funzione di libreria `saveNetwork` (da usare con estrema cautela). Nel caso in cui si modificassero permanentemente i parametri UWB di un dispositivo e ci si dimenticasse quali fossero, si può sempre reinstallare da capo il firmware del dispositivo stesso.

## 2.4 Accorgimenti

- Ciascun dispositivo può possedere al suo interno una lista degli altri dispositivi presenti nella rete. Tale lista prevede che per ogni dispositivo sia fornito un flag (1 se si tratta di un'ancora e 0 se si tratta di un tag) e le coordinate. Tale lista può essere gestita mediante le funzioni "*Device list functions*" della libreria python `pypozyx`. Tali funzioni permettono di aggiungere dispositivi alla lista, cancellare la lista, leggere le coordinate dei dispositivi della lista, ecc. È importante dire che quando un dispositivo viene staccato dall'alimentazione perde la propria lista interna dei dispositivi. Se si vuole che tale lista venga permanentemente salvata sul dispositivo, occorre chiamare la funzione `saveNetwork`, la quale salva la lista sulla memoria flash del dispositivo. Se si vuole che ciascun dispositivo ricordi la propria posizione, occorre prima aggiungere il dispositivo alla device list del dispositivo stesso e poi chiamare la funzione `saveNetwork`.
- Variare il rate di aggiornamento dei dispositivi Pozyx richiede di variare opportunamente i parametri dell'Ultra Wide Band. I valori tipici di aggiornamento sono riportati nella sezione relativa all'esperimento sulla frequenza.

- I dispositivi Pozyx possono interagire solo con altri dispositivi che abbiano gli stessi parametri UWB. Ciò consente, per esempio, di avere più dispositivi pozyx nel medesimo ambiente senza che ci sia relativo disturbo.
- Accertarsi che non ci siano interferenze di natura elettromagnetica tra i dispositivi. Eventuali interferenze di questo tipo, specialmente se interposte tra le ancore, possono rendere difficile la comunicazione tra i dispositivi della rete.

## 2.5 Caratterizzazione di un fallimento nel positioning

In caso di fallimento del Positioning il sistema Pozyx può comportarsi in due modi:

- nel caso in cui una delle ancore della lista interna del dispositivo (cioè le ancore che il dispositivo sa di dover interrogare per effettuare il positioning) sia spenta, NON VIENE SEGNALATO UN POZYX\_FAILURE. Il valore di return della funzione doPositioning può infatti essere POZYX\_SUCCESS o POZYX\_FAILURE a seconda che il procedimento di Positioning abbia avuto successo o meno. Nel caso in cui un'ancora sia spenta però, NON verrà segnalato un POZYX\_FAILURE. E' comunque possibile determinare un fallimento del positioning in quanto le coordinate del tag risulteranno essere poste di default a (0,0,0), la distanza dall'ancora spenta è posta a 0, l'RSS ( Received signal strength) tra il tag e l'ancora è posto a 0 e infine la timestamp è posta al valore di default pari a 1705985.
- nel caso in cui tutte le ancore siano correttamente accese ma si fallisca a ricevere la distanza da una delle ancore (signal loss), il sistema Pozyx notificherà l'evento tramite un POZYX\_FAILURE. Se si vanno a leggere i dati relativi alle distanze delle ancore, risulterà evidente che i dati relativi all'ancora con cui è fallita la comunicazione sono identici ai precedenti. Ciò significa che quando il doPositioning chiama internamente la doRanging, quando una delle doRanging fallisce, viene segnalato un POZYX\_FAILURE della procedura di positioning.
- un'altra causa di fallimento potrebbe essere dovuto ai parametri UWB che possono essere impostati differentemente sui vari dispositivi. I dispositivi con parametri UWB differenti da quelli del dispositivo che effettua il positioning appariranno come se fossero spenti, come nel caso 1. Conseguentemente le coordinate del tag ottenute tramite il doPositioning risulteranno essere (0,0,0).

### 3 Aspetti teorici

#### 3.1 Algoritmo di autocalibrazione

L'algoritmo di autocalibrazione sviluppato da Fioretti e da noi leggermente modificato e riscritto in linguaggio Python viene qui descritto.

L'algoritmo è basato su un approccio algebrico al problema di calibrazione delle ancore ed è pensato per un setup in cui siano presenti 4 ancore. Sulla base delle 6 distanze relative tra le varie coppie di ancore e sulla base di una convenzione stabilita a priori sul sistema di riferimento utilizzato, vengono fornite le coordinate delle quattro ancore. Siamo interessati a determinare le coordinate incognite  $(x_0, y_0, z_0)$ ,  $(x_1, y_1, z_1)$ ,  $(x_2, y_2, z_2)$ ,  $(x_3, y_3, z_3)$  sulla base delle distanze relative  $(r_{01}, r_{02}, r_{03}, r_{12}, r_{13}, r_{23})$ . Abbiamo preferito modificare l'algoritmo già sviluppato affinché si adattasse meglio alle nostre esigenze su python. Infatti veniva precedentemente chiamato un risolutore di sistemi di equazioni di matlab che non è presente sul software python. Pertanto abbiamo ritenuto più opportuno modificare leggermente l'algoritmo, cercando manualmente la soluzione in forma chiusa del problema. La convenzione utilizzata impone che l'ancora 0 si trovi nell'origine, l'ancora 1 si trovi sull'asse y ad una ordinata positiva, l'ancora 2 abbia un'ascissa positiva e l'ancora 3 una quota positiva. Le prime tre ancore sono infine coplanari e determinano quindi la direzione dell'asse z. Ciò impone che:

$$(x_0, y_0, z_0) = (0, 0, 0)$$

$$(x_1, y_1, z_1) = (0, r_{01}, 0)$$

$$z_2 = 0$$

Il sistema da risolvere per determinare le coordinate della seconda ancora è il seguente:

$$\begin{cases} d(0, 2)^2 = r_{02}^2 \rightarrow x_2^2 + y_2^2 = r_{02}^2 \\ d(1, 2)^2 = r_{12}^2 \rightarrow x_2^2 + (y_2 - y_1)^2 = r_{12}^2 \end{cases} \quad (1)$$

Sottraendo la seconda equazione dalla prima si ottiene l'espressione esplicita:

$$y_2 = \frac{r_{01}^2 - r_{12}^2 + r_{02}^2}{2r_{01}}$$

Dalla convenzione sulla positività di  $x_2$  si ottiene:

$$x_2 = +\sqrt{r_{02}^2 - y_2^2}$$

Il sistema da risolvere per determinare le coordinate della terza ancora è il seguente:

$$\begin{cases} d(0, 3)^2 = r_{03}^2 \rightarrow x_3^2 + y_3^2 + z_3^2 = r_{03}^2 \\ d(1, 3)^2 = r_{13}^2 \rightarrow x_3^2 + (y_3 - y_1)^2 + z_3^2 = r_{13}^2 \\ d(2, 3)^2 = r_{23}^2 \rightarrow (x_3 - x_2)^2 + (y_3 - y_2)^2 + z_3^2 = r_{23}^2 \end{cases} \quad (2)$$

Sottraendo dalla prima equazione la seconda si ottiene l'espressione esplicita:

$$y_3 = \frac{r_{03}^2 - r_{13}^2 + y_1^2}{2y_1}$$

Sottraendo poi dalla seconda equazione la terza si ottiene l'espressione esplicita per  $x_2$ :

$$x_3 = \frac{x_2^2 - 2y_3y_2 + y_2^2 - r_{23}^2 + r_{03}^2}{2x_2}$$

Dalla convenzione sulla positività di  $z_3$  si ottiene:

$$z_3 = +\sqrt{r_{03}^2 - x_3^2 - y_3^2}$$

## 3.2 Algoritmo RANSAC

RANSAC (RANdom SAmple Consensus) è un metodo iterativo utilizzato per stimare i parametri di un modello matematico a partire da un insieme di osservazioni sperimentali contenente una grande percentuale di outliers.

Si tratta di un algoritmo non deterministico pubblicato per la prima volta nel 1981 da Fischler e Bolles.

L'algoritmo RANSAC necessita di due funzioni: una prima funzione che sia in grado di generare una stima dei parametri mediante un fit dei dati campione, una seconda funzione in grado di determinare sia l'errore tra un dato campione e il modello derivante dal fit, sia l'errore associato ad un intero set di campioni.

Il suo funzionamento si basa essenzialmente sui seguenti step:

1. Si selezionano in maniera casuale un insieme di campioni dal dataset di partenza (parametro  $n$ ), si genera un primo fit dei campioni ottenuti e si effettua una prima stima dei parametri del modello;
2. RANSAC controlla quali elementi dell'intero dataset sono consistenti con i parametri stimati: se il punto preso in esame è compatibile con il modello in uso (la distanza tra il punto e il modello deve essere minore di un valore soglia  $t$ ), si considera il punto un inliers, viceversa il punto viene classificato come outliers. Il set di dati classificati come inliers è chiamato Consensus Set. Se il numero di inliers così determinati è superiore ad un certo numero  $d$  fissato a priori, si procede con il passo 3, altrimenti si ripete il punto 1.
3. A questo punto viene effettuato un secondo fit sul Consensus Set e si determina l'errore complessivo del modello, ad esempio in termini di varianza ottenuta. Se l'errore così ottenuto è inferiore a quello ottenuto al passo precedente, i parametri del nuovo modello vengono salvati.

Questa procedura viene ripetuta un numero fissato di volte ( $k$ ), producendo ad ogni iterazione o un modello che viene rifiutato a causa del basso numero di punti classificati come inlier, o un modello corretto assieme ad una corrispondente misura dell'errore.

A differenza delle classiche tecniche che utilizzano la maggior quantità di dati possibile per ottenere una soluzione e solo successivamente procedono a eliminare i valori anomali, RANSAC utilizza un piccolo insieme di dati per determinare il modello e procede nell'ingrandire questo insieme con i punti di dati coerenti con il modello stimato.

Ricapitolando, i parametri da fornire all'algoritmo sono i seguenti:

1.  $n$  = numero di dati estratti randomicamente;
2.  $k$  = numero di iterazioni eseguite dall'algoritmo;
3.  $t$  = valore di soglia per determinare se un dato è conforme al modello;
4.  $d$  = numero di inliers necessari per asserire che un modello rappresenta bene i dati;

### 3.2.1 Vantaggi e svantaggi RANSAC

L'algoritmo RANSAC presenta vantaggi e svantaggi. Tra i vantaggi si menziona l'abilità a produrre stime robuste del modello dei parametri; tuttavia uno dei grandi svantaggi di RANSAC è che non esiste un limite superiore al tempo che potrebbe impiegare per calcolare tali parametri; inoltre quando il numero di iterazioni è limitato la soluzione trovata potrebbe essere subottima. Per ovviare a tale problema, si aumenta il numero di iterazioni aumentando così la probabilità di ottenere un modello ragionevole. Un altro svantaggio di RANSAC è che richiede il settaggio dei parametri in base al problema.

### 3.2.2 Implementazione Python e risultati sperimentali

Il codice implementativo dell'algoritmo RANSAC è stato scaricato dal seguente sito: <https://scipy-cookbook.readthedocs.io/items/RANSAC.html> e riadattato al nostro caso.

Nel nostro caso, il modello da fittare sono semplicemente sei valori costanti che rappresentano le distanze tra le varie coppie di ancore.

Al termine della procedura di autocalibrazione viene mostrato un grafico per ciascuna coppia di ancore. In tale grafico vengono riportati in ascissa l'indice del campione e in ordinate il valore della distanza ottenuta mediante la procedura di autoranging. Il dato è poi colorato di blu se appartiene al Consensus Set, in nero se invece è identificato come outlier e quindi scartato. Una linea rossa rappresenta i fit dell'algoritmo ransac, una linea verde invece rappresenta la media ottenuta sul dataset di partenza. Il codice viene anche riportato in appendice nella sezione *Script di base*.

Di seguito vengono riportati alcuni grafici relativi a prove sperimentali realizzate. Il dataset di partenza è sempre il medesimo e vengono fatti variare i parametri  $n$ ,  $k$ ,  $t$ ,  $d$ .

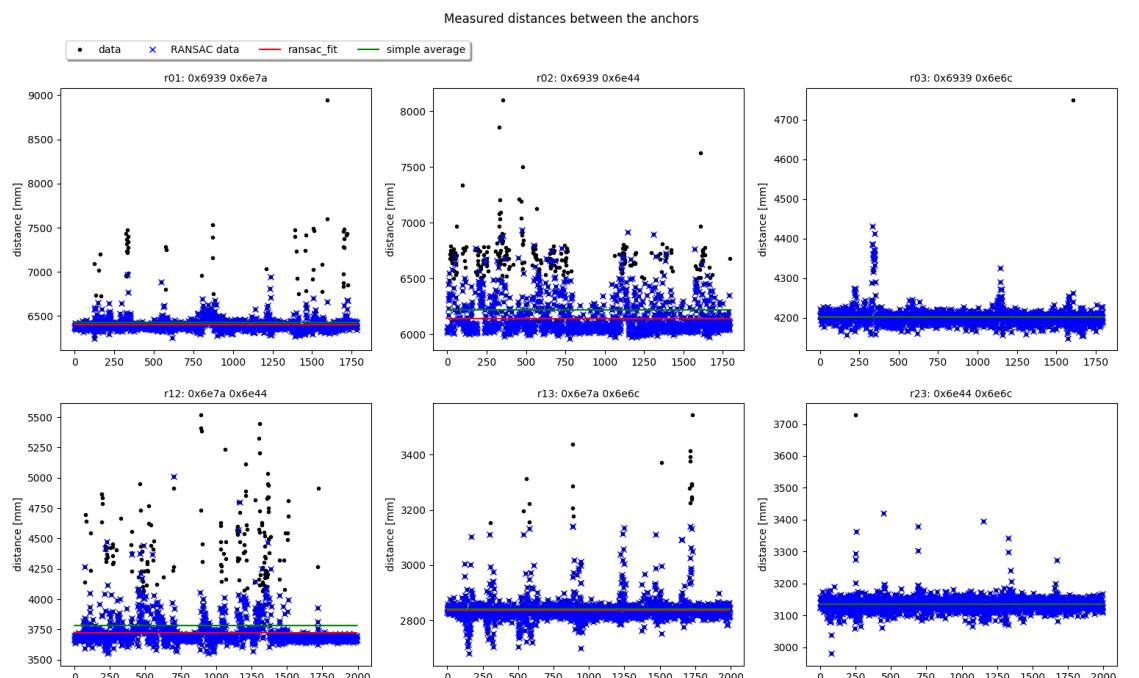


Figura 4: Applicazione dell'algoritmo RANSAC sui dati grezzi di autoranging: prova 1 ( $n=20\%$ ,  $k=10^3$ ,  $t = 10^5$ ,  $d = 60\%$ )

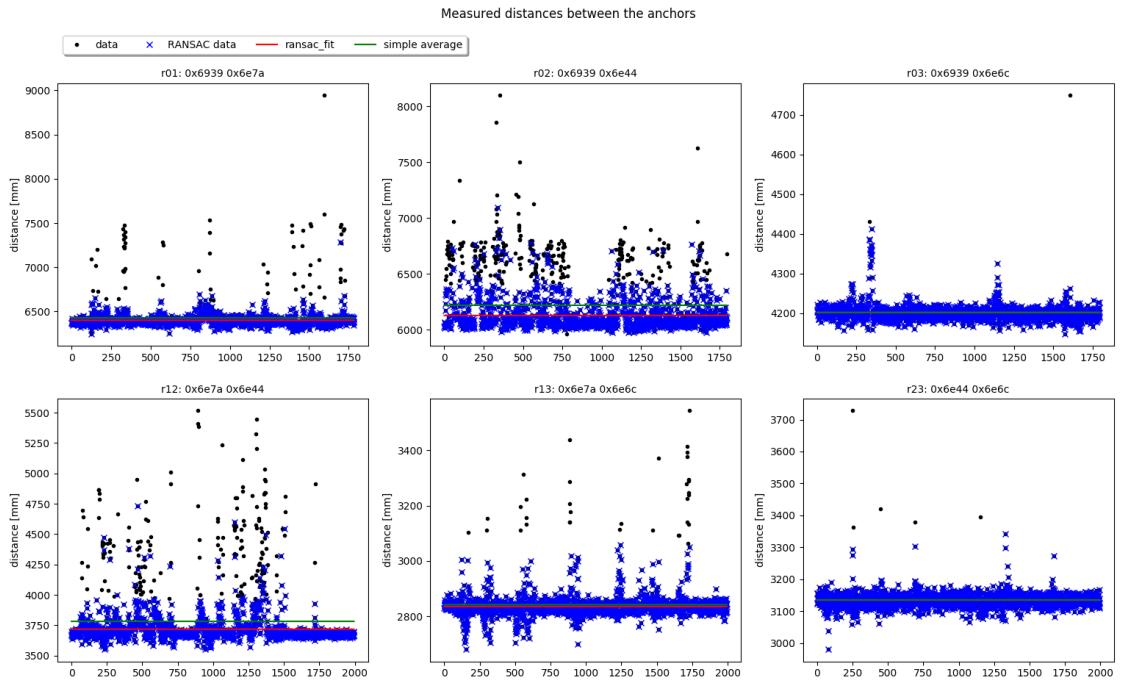


Figura 5: Applicazione dell'algoritmo RANSAC sui dati grezzi di autoranging: prova 2 ( $n=20\%$ ,  $k=10^4$ ,  $t = 5 \cdot 10^4$ ,  $d = 60\%$ )

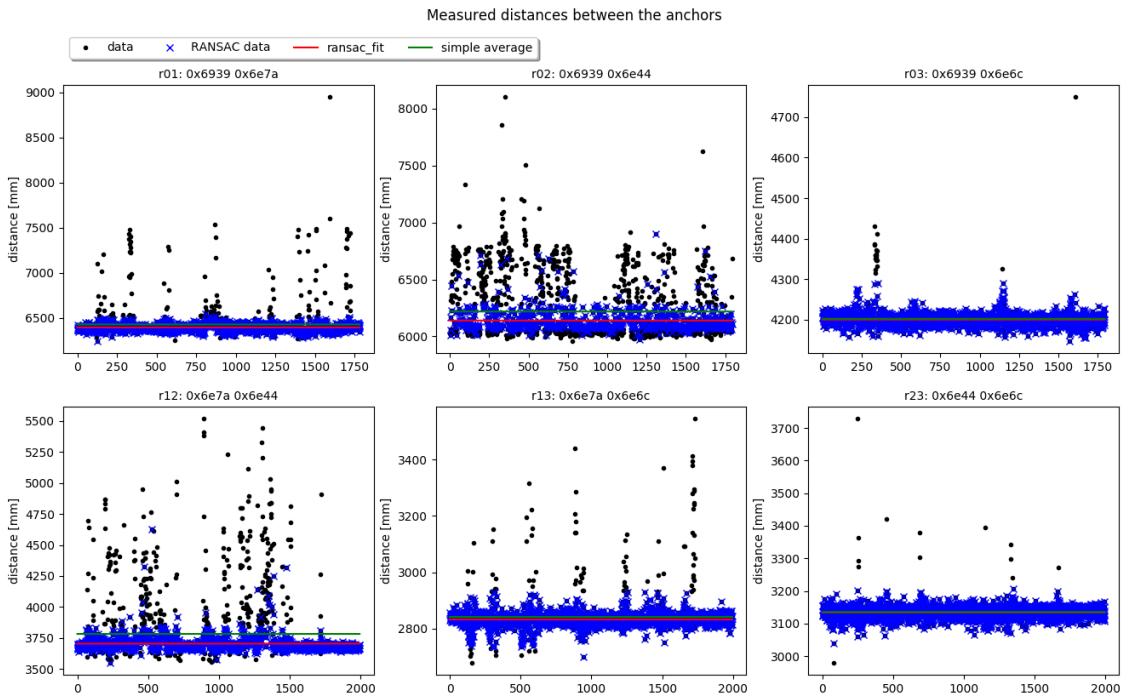


Figura 6: Applicazione dell'algoritmo RANSAC sui dati grezzi di autoranging: prova 3 ( $n=10\%$ ,  $k=2 \cdot 10^5$ ,  $t = 10^4$ ,  $d = 30\%$ )

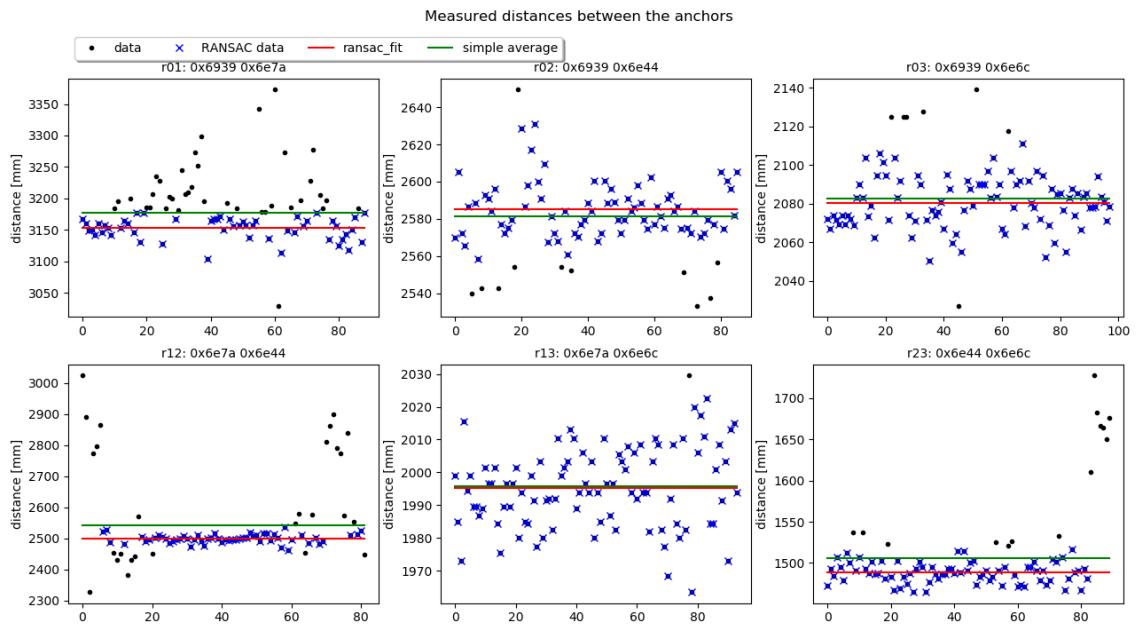


Figura 7: Applicazione dell'algoritmo RANSAC sui dati grezzi di autoranging: prova 3 ( $n=10\%$ ,  $k=2 \cdot 10^5$ ,  $t = 2500$ ,  $d = 30\%$ )

### 3.3 Propagazione dell'errore

La misura di una grandezza fisica è generalmente accompagnata dalla stima dell'errore ad essa associato. In particolare, le misure si distinguono in :

- misure dirette: quelle in cui uno strumento fornisce direttamente il risultato di una certa misura;
- misure indirette: quelle ottenute mediante combinazioni algebriche di misure dirette.

Noti gli errori sulle misure dirette, la propagazione di questi sulle misure indirette è basata sul concetto di derivata parziale.

Supponiamo di aver misurato una grandezza  $x_0$  e di aver misurato il suo errore  $\Delta x$ .

Siamo ora interessati a vedere come si propagherà l'incertezza di  $x_0$  sulla grandezza  $y = f(x)$ .

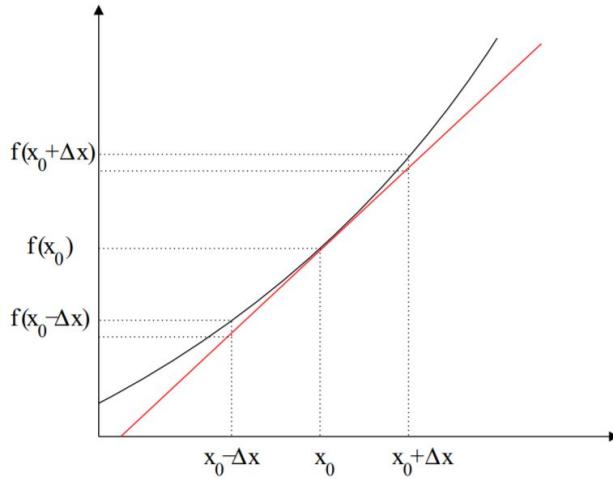


Figura 8: Propagazione dell'errore di misura

Il più grande valore probabile di  $x_0$  è  $x_0 + \Delta x$  mentre il più piccolo valore probabile è  $x_0 - \Delta x$ . Dal grafico si può vedere che il più grande e il più piccolo valore probabile di  $y$  è  $f(x_0 + \Delta x)$  e  $f(x_0 - \Delta x)$ . Se l'errore  $\Delta x$  è sufficientemente piccolo rispetto a  $x_0$ , allora il tratto di curva compreso fra  $x_0 - \Delta x$  e  $x_0 + \Delta x$  si può approssimare con la retta tangente in  $x_0$  e assumere che, essendo uguale l'ampiezza degli intervalli  $[x_0 - \Delta x, x_0]$  e  $[x_0, x_0 + \Delta x]$  :

$$\begin{cases} f(x_0 + \Delta x) - f(x_0) = \Delta f \\ f(x_0) - f(x_0 - \Delta x) = \Delta f \end{cases} \quad (3)$$

Sommando membro a membro si ricava che l'incertezza da associare alla funzione  $y$  risulta:

$$\Delta f = \frac{f(x_0 + \Delta x) + f(x_0 - \Delta x)}{2} \quad (4)$$

Poiché in generale  $f(x_0 + \Delta x)$  e  $f(x_0 - \Delta x)$  sono spesso incogniti o difficili da derivare, si deve ricorrere a qualche approssimazione.

Consideriamo l'espressione :

$$\Delta f = f(x_0 + \Delta x) - f(x_0) \quad (5)$$

Essendo  $\Delta x$  piccolo possiamo espandere  $y$  in serie di Taylor al primo ordine e scrivere:

$$f(x_0 + \Delta x) = f(x_0) + \frac{\partial f}{\partial x}(x_0 + \Delta x - x_0) = f(x_0) + \left. \frac{\partial f}{\partial x} \right|_{x=x_0} \Delta x \quad (6)$$

Quindi:

$$\Delta f = \left. \frac{\partial f}{\partial x} \right|_{x=x_0} \Delta x \quad (7)$$

Si faccia attenzione che non necessariamente  $(x_0 + \Delta x) > (x_0 - \Delta x)$  implica  $f(x_0 + \Delta x) > f(x_0 - \Delta x)$ . Nel caso in cui la pendenza della retta sia negativa, si avrà:

$$\Delta f = -\left. \frac{\partial f}{\partial x} \right|_{x=x_0} \Delta x \quad (8)$$

e quindi generalizzando si ha:

$$\Delta f = \left| \left. \frac{\partial f}{\partial x} \right| \right|_{x=x_0} \Delta x \quad (9)$$

Se la grandezza  $y$  è funzione di  $k$  grandezze  $x_i$  misurate direttamente, cioè  $y = f(x_1, \dots, x_k)$ , allora dovremo fare uso del concetto di differenziale di una funzione di più variabili: per variazioni infinitesime  $dx_i$  la variazione di  $y$  è data dal differenziale di  $f(x)$ :

$$df = \sum_{i=1}^k \frac{\partial f}{\partial x_i} dx_i \quad (10)$$

Se gli errori  $\Delta x_i$  sono sufficientemente piccoli da giustificare l'approssimazione lineare e le derivate sono non nulle, l'errore massimo di  $y$  è dato dal differenziale della funzione  $f(x_1, \dots, x_k)$  prendendo i moduli delle derivate:

$$\Delta f = \sum_{i=1}^k \left| \frac{\partial f}{\partial x_i} \right| \Delta x_i \quad (11)$$

### 3.3.1 Applicazione alla calibrazione manuale

Il calcolo della propagazione degli errori per la calibrazione manuale del sistema Pozyx è stato effettuato mediante uno script matlab riportato in appendice G.

## 4 Esperimento 1: verifica di bias

Con questo primo esperimento siamo interessati a verificare l'eventuale presenza o assenza di bias nella misurazione della distanza relativa fra due ancore. Per fare ciò abbiamo effettuato una misurazione manuale con un metro laser della distanza relativa tra di esse e ripetuto tale misurazione in varie configurazioni, come riportato in Tabella 1 e Tabella 2.

Le misure ottenute con il metro laser sono state confrontate con quelle derivanti da una procedura automatica di auto-ranging delle ancore. Nella procedura automatica abbiamo ottenuto una stima del valor medio e dello scarto quadratico medio delle misure tramite un ciclo di acquisizione con 1000 campioni.

In appendice è riportato lo script python utilizzato per fare le acquisizioni dei dati (B).

Ricordarsi, nell'eventualità di una replicazione della prova, di cambiare gli ID delle ancore. I parametri UWB non sono stati settati, pertanto sono pari ai valori di default: bitrate 110 kbit/sec, canale 5, PRF 64 MHz, gain 11.5 dB e preamble length 1024 symbols.

| Prova | Metro laser [mm] | Ancore [mm]    | Bias [mm] |
|-------|------------------|----------------|-----------|
| 1     | $1554 \pm 10$    | $1416 \pm 13$  | 138       |
| 2     | $2740 \pm 10$    | $2656 \pm 12$  | 84        |
| 3     | $3870 \pm 10$    | $3786 \pm 16$  | 84        |
| 4     | $4457 \pm 10$    | $4355 \pm 15$  | 102       |
| 5     | $5571 \pm 10$    | $5450 \pm 21$  | 121       |
| 6     | $6640 \pm 10$    | $6494 \pm 17$  | 146       |
| 7     | $8430 \pm 10$    | $8219 \pm 21$  | 211       |
| 8     | $9490 \pm 10$    | $9336 \pm 18$  | 154       |
| 9     | $11042 \pm 10$   | $10868 \pm 20$ | 174       |

Tabella 1: Distanza variabile - Angolo di vista fisso

| Prova | Angolazione             | Metro laser [mm] | Ancore [mm]   | Bias [mm] |
|-------|-------------------------|------------------|---------------|-----------|
| 1_1   | $0^\circ \pm 5^\circ$   | $5571 \pm 10$    | $5450 \pm 21$ | 121       |
| 2_1   | $45^\circ \pm 5^\circ$  | $5620 \pm 10$    | $5620 \pm 19$ | 150       |
| 3_1   | $90^\circ \pm 5^\circ$  | $5700 \pm 10$    | $5499 \pm 15$ | 201       |
| 4_1   | $135^\circ \pm 5^\circ$ | $5834 \pm 10$    | $5704 \pm 20$ | 130       |
| 5_1   | $180^\circ \pm 5^\circ$ | $5892 \pm 10$    | $5803 \pm 45$ | 89        |

Tabella 2: Distanza fissa - Angolo di vista variabile

Nella prima prova abbiamo lasciato le ancore ad una angolazione fissa, cioè le abbiamo poste frontalmente. Abbiamo ripetuto le acquisizioni a varie distanze come riportato in Tabella 1. Abbiamo poi lasciato le ancore ad una distanza costante e variato l'angolo di vista, come riportato in Tabella 2.

Di seguito sono riportati gli istogrammi relativi alle varie prove.

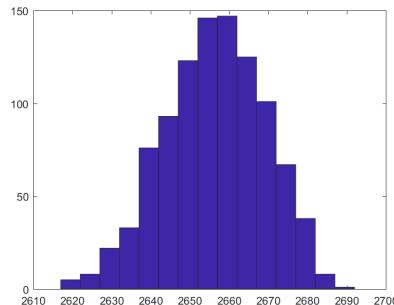


Figura 9: Istogramma prova 1

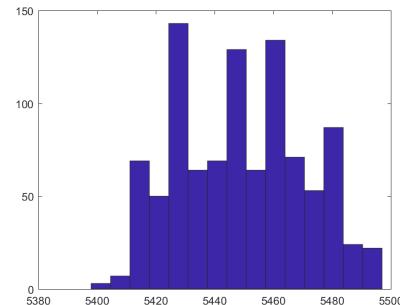


Figura 10: Istogramma prova 2

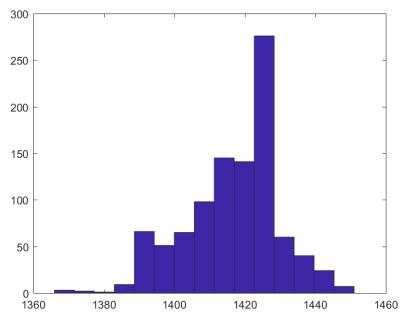


Figura 11: Istogramma prova 3

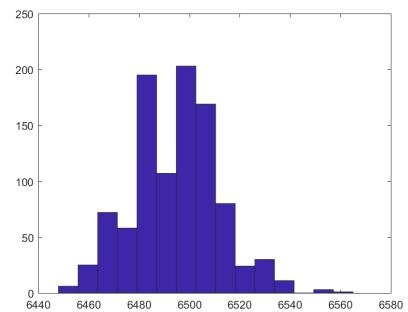


Figura 12: Istogramma prova 4

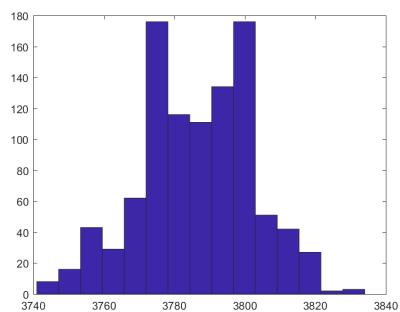


Figura 13: Istogramma prova 5

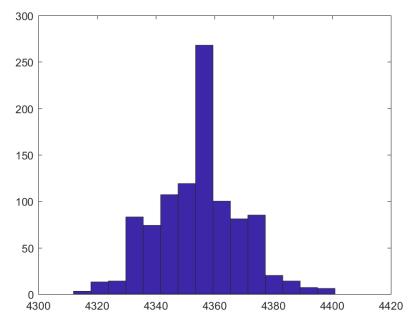


Figura 14: Istogramma prova 6

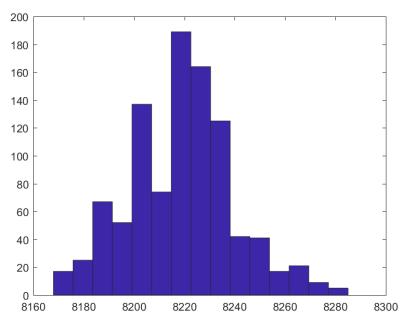


Figura 15: Istogramma prova 7

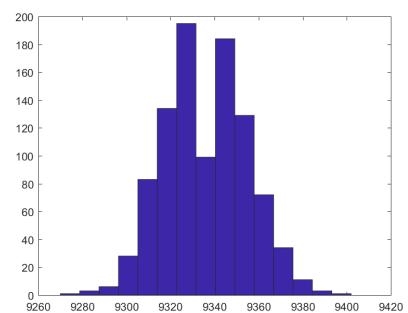


Figura 16: Istogramma prova 8

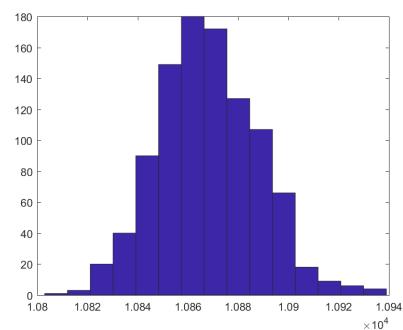


Figura 17: Istogramma prova 9

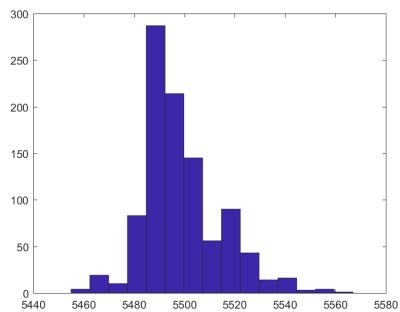


Figura 18: Istogramma prova 1\_1

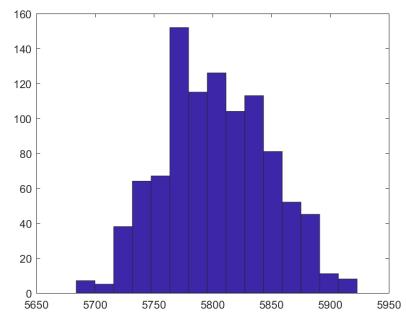


Figura 19: Istogramma prova 2\_1

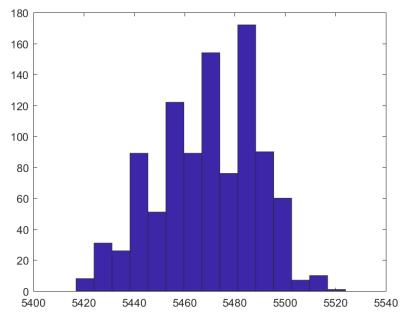


Figura 20: Istogramma prova 3\_1

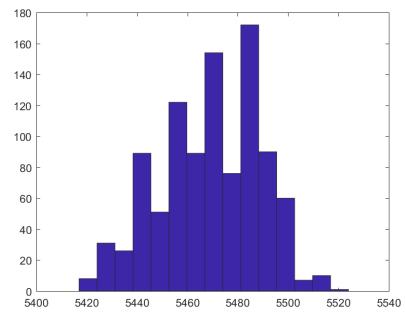


Figura 21: Istogramma prova 4\_1

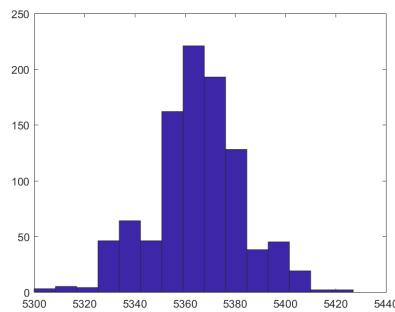


Figura 22: Istogramma prova 5\_1

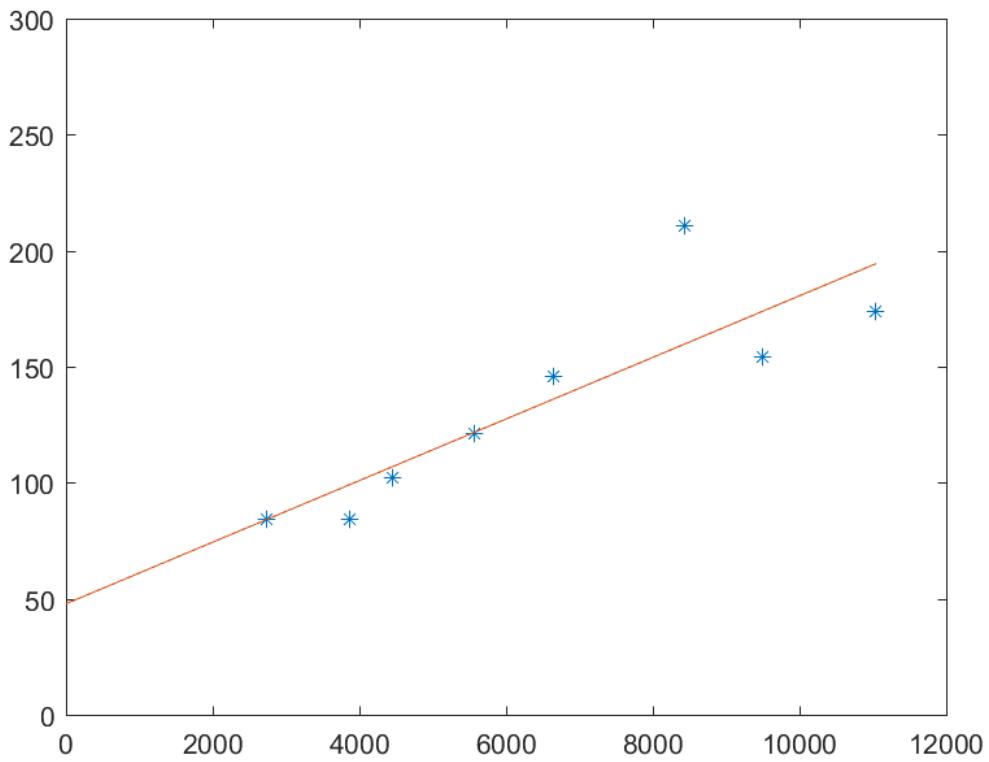


Figura 23: Fit lineare (relativo alla Tabella 1)

#### 4.1 Conclusioni

Dai dati riportati in Tabella 1 sembrerebbe esserci effettivamente un trend lineare nel bias tra le due antenne. Tuttavia si nota come il bias dipenda in maniera non trascurabile dall'angolazione relativa tra le ancore, oscillando tra i valori 89 e 201 mm quando la distanza è tenuta circa costante e pari a 5600 mm. Possiamo quindi concludere che è vero che c'è un trend lineare del bias quando le ancore sono mantenute frontalmente l'una rispetto all'altra, ma dato che il bias cambia in maniera non trascurabile quando l'angolo di vista tra le due ancore viene variato (anche del 100%), non avrà senso ripulire le misurazioni rimuovendo il trend lineare. Inoltre, i fattori da cui dipende il bias possono essere molteplici, ad esempio la distanza tra le antenne, l'angolazione relativa, il rumore ambientale, la geometria dell'ambiente in cui sono poste le antenne e i materiali in esso presenti. Pertanto è possibile considerare il bias lineare solo in una prima approssimazione, ritenuta molto grossolana.

Suggeriamo pertanto, quando possibile, una calibrazione manuale delle antenne, in quanto ritenuta più affidabile.

## 5 Esperimento 2: auto-calibrazione 2D

### 5.1 Descrizione dell'esperimento

Lo scopo di questo test è quello di verificare le performance dell'algoritmo di autocalibrazione 2D del sistema Pozyx. Tale algoritmo dovrebbe, come risultato, fornire le coordinate assolute di ciascuna ancora. La prova è stata effettuata indoor, dove le performance del sistema sono leggermente peggiori in quanto affette da un più marcato rumore ambientale e a causa della presenza di oggetti metallici. Le ancora sono state disposte ai vertici di un rettangolo e alla medesima altezza. Infatti, in questo algoritmo, non viene determinata l'altezza delle singole ancora poiché esso è pensato per una situazione in cui il tag si muove coplanare alle antenne. Se si volesse far muovere il tag parallelamente ad un piano contenente le antenne, i costruttori suggerirebbero invece l'utilizzo dell'algoritmo denominato 2.5D.

Abbiamo creato uno script in Python (vedi C) che fa partire l'autocalibrazione delle ancora e salva in un file di testo il risultato dell'algoritmo di autocalibrazione, ossia le coordinate delle ancora che sono state determinate. Lo script prevede di eseguire tale procedimento 100 volte, in modo tale da poter disporre di dati statistici per determinare valor medio e varianza della posizione di ciascuna ancora.

I parametri UWB sono stati settati ai valori di default: bitrate 110 kbit/sec, canale 5, PRF 64 MHz, gain 11.5 dB e preamble length 1024 symbols. È importante notare che l'ordine in cui vengono passati gli id delle ancora all'algoritmo di autocalibrazione è importante: la prima ancora avrà di default coordinate [0,0], ossia sarà l'origine del sistema di riferimento, la seconda ancorà avrà una prima coordinata positiva (asse X positivo) e la seconda coordinata nulla (ossia determinerà l'asse X del sistema di riferimento), la terza ancora avrà infine una coordinata Y positiva.

Il risultato dell'autocalibrazione è stato poi confrontato con le misurazioni del metro laser.

### 5.2 Risultati

L'analisi dei dati raccolti è stata effettuata con Matlab. Il numeroso insieme di punti ottenuto è stato rappresentato graficamente nel piano. Come c'era da aspettarsi, per ciascuna ancora non si è ottenuta esattamente la stessa posizione ad ogni iterazione, tuttavia i dati non si addensavano esattamente attorno ad un punto nello spazio, ma per ciascuna ancora è stato possibile identificare tre possibili punti attorno ai quali si addensano i dati sperimentali, come mostrato in figura 25.

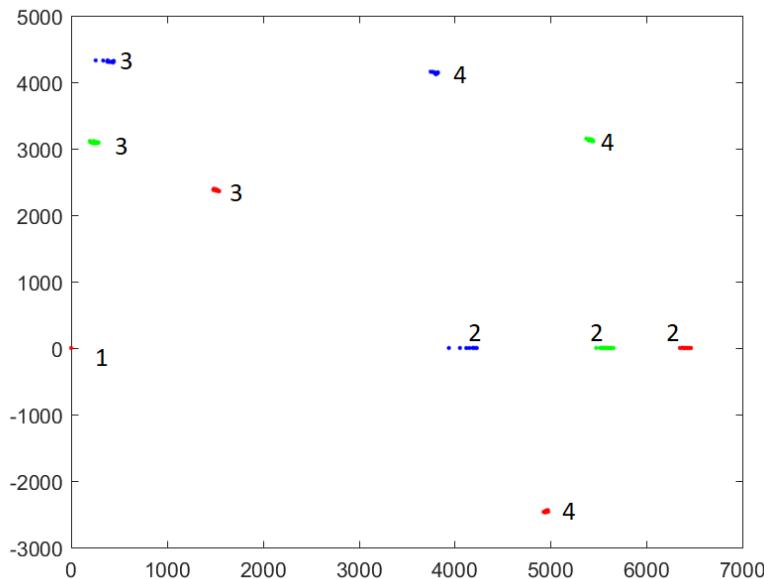


Figura 24: Esperimento 3: gruppi delle misure relative alle 3 ancora. Le misure sono espresse in mm. I numeri indicano a quale ancora corrispondono i gruppi di dati, i colori corrispondono alla configurazioni corrispondenti. La configurazione verde è quella corrispondente alla reale posizione delle ancora.

Abbiamo raggruppato i risultati, ossia le posizioni di tutte le ancore determinate dopo ogni iterazione dell'algoritmo di autocalibrazione, in tre gruppi. Solamente uno di tali gruppi è quello che effettivamente corrisponde alla geometria reale delle ancore, pertanto, possiamo concludere che l'algoritmo di autocalibrazione 2D, nelle condizioni specificate, necessita di qualche conoscenza a priori delle geometrie delle ancore per poter essere significativo.

Per il gruppo le cui distanze corrispondono a quelle determinate con il metro laser, sono stati determinati valor medio e varianza di ciascuna antenna.

| Ancora | $\mu_x$ | $\sigma_x$ | $\mu_y$ | $\sigma_y$ |
|--------|---------|------------|---------|------------|
| 1      | 0       | 0          | 0       | 0          |
| 2      | 5588    | 36         | 0       | 0          |
| 3      | 249     | 23         | 3093    | 67         |
| 4      | 5143    | 13         | 3130    | 76         |

Tabella 3: Autocalibrazione: configurazione rettangolare. Valor medio e deviazione standard [mm]

### 5.3 Nuova configurazione

Abbiamo provato una nuova geometria delle ancore, le quali sono state disposte ai vertici di un quadrilatero irregolare. Stavolta l'algoritmo converge verso due possibili insiemi di misure invece che tre. La configurazione che corrisponde a quella calcolata tramite il metro laser è la blu. Dal confronto tra le due possibili configurazioni sembrerebbe che la seconda sia la migliore, in quanto dai dati si ottengono due diversi gruppi invece che 3. Sulla base di una conoscenza a priori del setup è possibile vedere quale dei due gruppi rispecchia il reale posizionamento delle ancore.

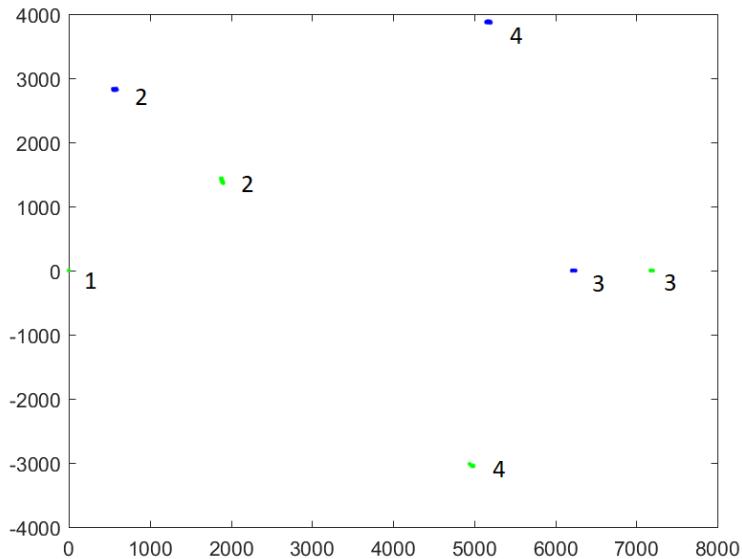


Figura 25: Esperimento 3: gruppi delle misure relative alle 3 ancore. Le misure sono espresse in mm. I numeri indicano a quale ancora corrispondono i gruppi di dati, i colori corrispondono alla configurazioni corrispondenti. La configurazione verde è quella corrispondente alla reale posizione delle ancore.

| <b>Ancora</b> | $\mu_x$ | $\sigma_x$ | $\mu_y$ | $\sigma_y$ |
|---------------|---------|------------|---------|------------|
| 1             | 0       | 0          | 0       | 0          |
| 2             | 6220    | 7          | 0       | 0          |
| 3             | 564     | 10         | 2828    | 5          |
| 4             | 5176    | 9          | 3880    | 5          |

Tabella 4: Autocalibrazione: configurazione a quadrilatero irregolare. Valor medio e deviazione standard [mm]

## 6 Esperimento 3: autocalibrazione 3D mediante webapp

In questa sezione vengono riportati i risultati derivanti da calibrazione delle ancore effettuata mediante applicazione web.

In particolare sono state lanciate 30 calibrazioni successive e per ognuna di esse sono state salvate le coordinate delle ancore in un file di testo. È da notare come non sia presente la misurazione della quota della quarta ancora né l'incertezza ad essa relativa. Infatti nell'autocalibrazione 3D sviluppata da Pozyx è previsto che l'utente inserisca manualmente la quota della terza ancora. È inoltre possibile bloccare alcune coordinate delle ancore in modo tale che al termine dell'algoritmo non vengano variate. Ciò consente, ad esempio, di avere un'ancora posizionata nell'origine del sistema di riferimento e di fissare l'asse  $y$  in base alla posizione di una seconda ancora a scelta dell'utente.

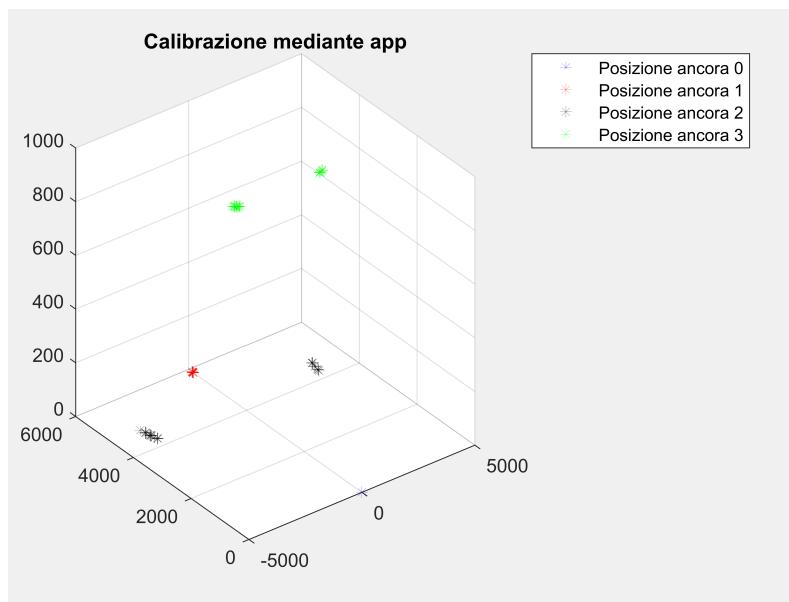


Figura 26: Calibrazione mediante app

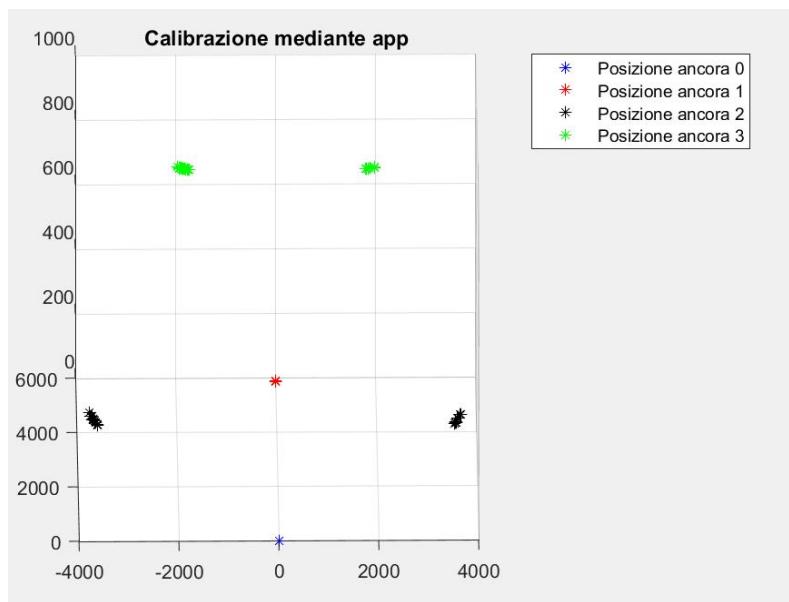


Figura 27: Calibrazione mediante app: vista frontale

| Ancora | $\mu_x$ [mm] | $\sigma_x$ [mm] | $\mu_y$ [mm] | $\sigma_y$ [mm] |
|--------|--------------|-----------------|--------------|-----------------|
| 2      | 0            | 0               | 5856         | 14              |
| 3      | 3651         | 50              | 4475         | 138             |
| 4      | 1877         | 70              | 2880         | 27              |

Tabella 5: Autocalibrazione

## 6.1 Conclusioni

Dal grafico 27 è possibile concludere che la calibrazione 3D effettuata mediante applicazione web non fornisce sempre il medesimo risultato; in particolare le configurazioni dell'ancora 2 e 3 risultano simmetrizzate rispetto a un piano passante per i punti dello spazio definiti dalle coordinate dell'ancora 0 e dell'ancora 1 .

## 7 Esperimento 4: autocalibrazione mediante algoritmo algebrico

In questo test vogliamo confrontare i risultati ottenuti mediante il medesimo algoritmo algebrico di autocalibrazione quando questo riceve in ingresso le misurazioni delle distanze relative fra le ancore effettuate con il metro laser o con i ranging automatici delle ancove.

I parametri UWB sono stati settati ai valori di default: bitrate 110 kbit/sec, canale 5, PRF 64 MHz, gain 11.5 dB e preamble length 1024 symbols. L'algoritmo di autocalibrazione prende in ingresso le 6 distanze in una configurazione minima tra tutte le coppie di ancove. Una configurazione minima presenta infatti 4 ancove a cui corrispondono 6 coppie possibili. L'output dell'algoritmo è costituito dalle coordinate  $[x, y, z]$  di ciascuna ancora. La convenzione adottata è basata sulle seguenti condizioni:

- ancora 0 posta in  $[0, 0, 0]$ ;
- ancora 1 definisce la direzione positiva dell'asse y ed ha quindi  $x = 0$  e  $z = 0$ ;
- ancora 2 possiede  $z = 0$ . Le prime tre ancove definiscono quindi il piano orizzontale della configurazione;
- ancora 3 posta nel semispazio avente  $z > 0$

Nel caso in cui in ingresso all'algoritmo siano date le distanze ottenute al metro laser si è calcolata l'incertezza con il metodo della propagazione degli errori. In particolare si è dapprima associato un errore a ciascuna misurazione della distanza al metro laser, il quale tiene di conto dell'incertezza della posizione dell'antenna nell'ancora stessa e di eventuali errori manuali. Poi abbiamo creato uno script matlab che calcola le incertezze sulle posizioni delle ancove sulla base delle prime. La funzione utilizzata è la seguente (aspetto teorico in sezione 3.3):

$$\Delta y = \sum_{i=1}^N \left| \frac{\partial y}{\partial x_i} \right| \Delta x_i$$

Tali incertezze possono essere calcolate mediante un apposito script matlab. Successivamente abbiamo effettuato 1000 acquisizioni delle distanze relative tra le varie coppie di ancove mediante uno script python. Queste 1000 6-uple di misurazioni vengono passate come argomento allo script matlab che effettua l'algoritmo algebrico. Ottenendo quindi 1000 diverse coordinate per ciascuna ancora. L'analisi statistica dei risultati è qui riportata.

| Ancora | $x$           | $y$           | $z$          |
|--------|---------------|---------------|--------------|
| 0      | 0             | 0             | 0            |
| 1      | 0             | $5950 \pm 10$ | 0            |
| 2      | $3783 \pm 15$ | $4573 \pm 19$ | 0            |
| 3      | $1814 \pm 37$ | $2917 \pm 23$ | $881 \pm 78$ |

Tabella 6: Metro laser

| Ancora | $x$           | $y$           | $z$           |
|--------|---------------|---------------|---------------|
| 0      | 0             | 0             | 0             |
| 1      | 0             | $6004 \pm 22$ | 0             |
| 2      | $3664 \pm 25$ | $4646 \pm 46$ | 0             |
| 3      | $1829 \pm 40$ | $2957 \pm 18$ | $543 \pm 139$ |

Tabella 7: Misurazione automatica

### 7.1 Conclusioni

Come era da aspettarsi, il bias che era presente sulle misurazioni della distanza tra due singole ancove si ripercuote sul bias presente nella procedure di autocalibrazione. È evidente inoltre come

la quota della quarta (ancora 3) ancora risulti essere la misurazione più incerta, sia in termini di bias che di varianza. Come è noto, infatti, le ancore forniscono misurazioni peggiori se sono poste ad altezze differenti in quanto le antenne sono meno sensibili lungo l'asse Z.

## 8 Esperimento 5: posizionamento tag

Lo scopo di questo esperimento è studiare le misurazioni del posizionamento del tag sotto diverse condizioni di setup delle ancore.

I parametri UWB sono stati settati ai valori di default: bitrate 110 kbit/sec, canale 5, PRF 64 MHz, gain 11.5 dB e preamble length 1024 symbols. In particolare, relativamente alla calibrazione delle ancore, viene fatto un confronto fra i risultati ottenuti applicando l'algoritmo algebrico (da noi modificato) a due diversi casi. Ricordiamo che l'algoritmo algebrico riceve come ingresso le distanze tra le coppie di antenne e fornisce in output le coordinate delle singole ancore. Nel primo caso le misurazioni delle distanze sono state effettuate con il metro laser, nell'altro caso sono stati dati come ingressi all'algoritmo le distanze fra le coppie di ancore ottenute mediante una procedura automatica.

Relativamente al positioning del tag, sono confrontati tre risultati:

- posizione del tag misurata col metro laser;
- posizione ottenuta dallo script python di positioning che prende in ingresso le coordinate delle ancore ottenute da calibrazione manuale;
- posizione ottenuta dallo script python di positioning che prende in ingresso le coordinate delle ancore ottenute da calibrazione automatica;

Di seguito vengono riportati i grafici ottenuti dalle simulazioni.

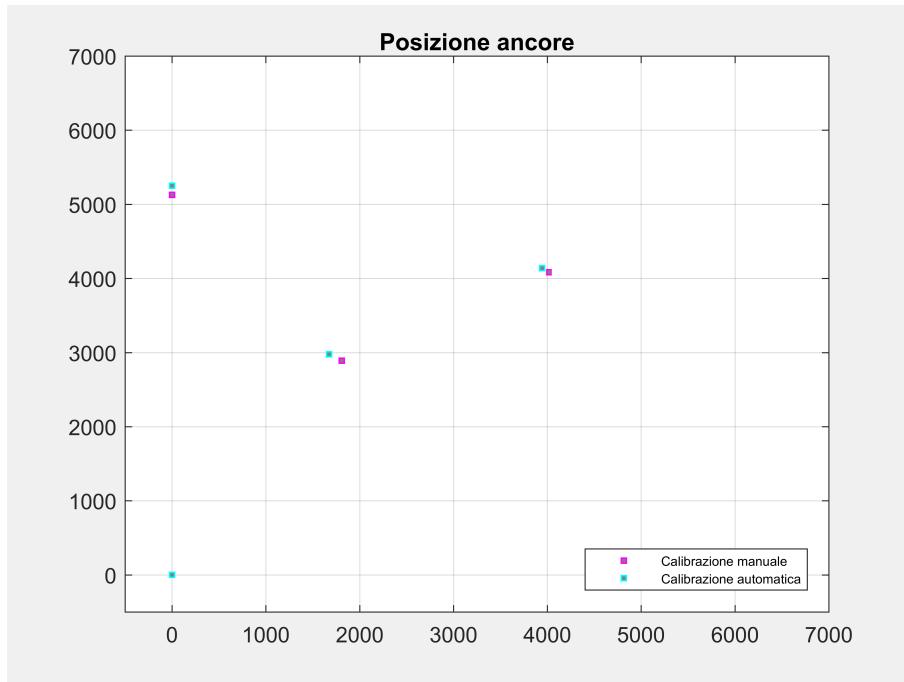


Figura 28: Posizione ancore, vista in pianta.

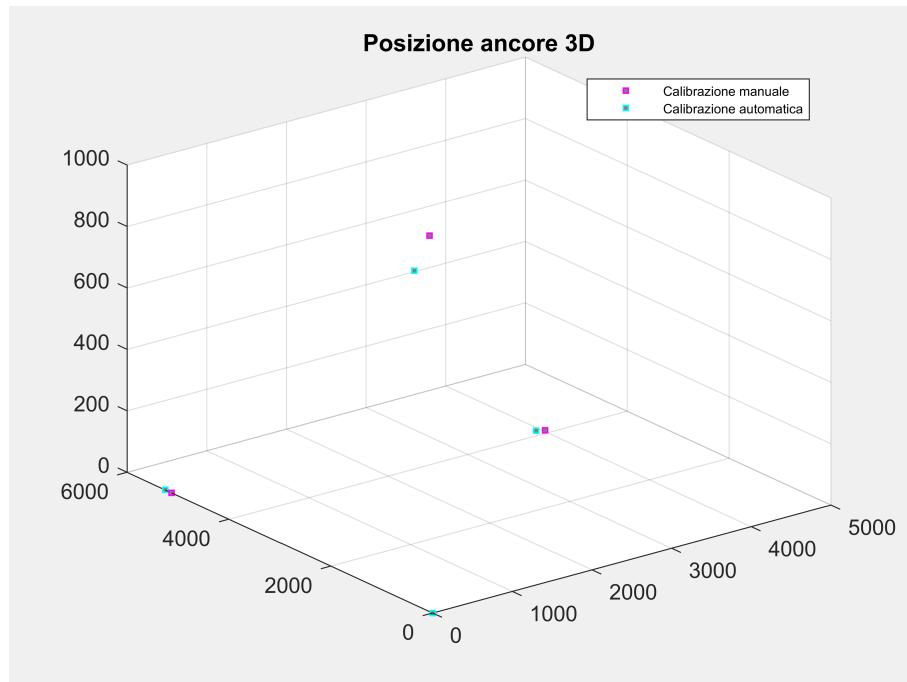


Figura 29: Posizione ancore 3D.

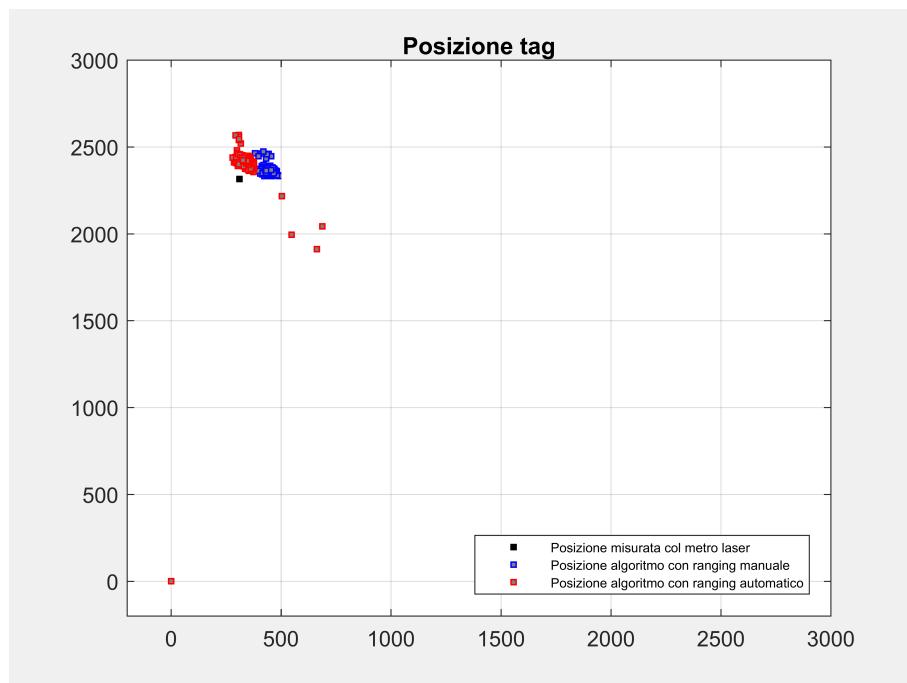


Figura 30: Posizione tag, vista in pianta.

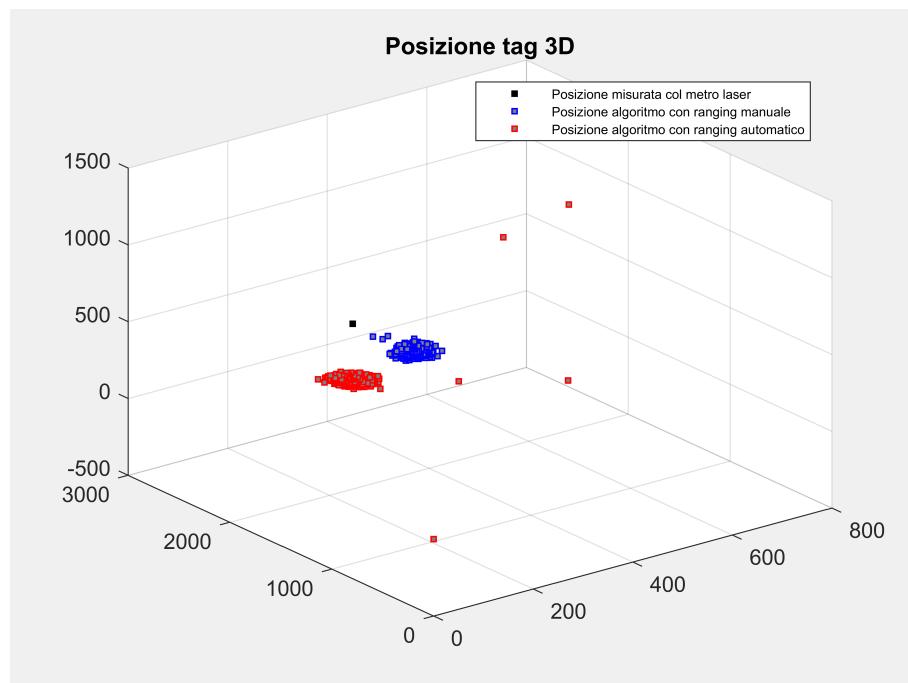


Figura 31: Posizione tag 3D.

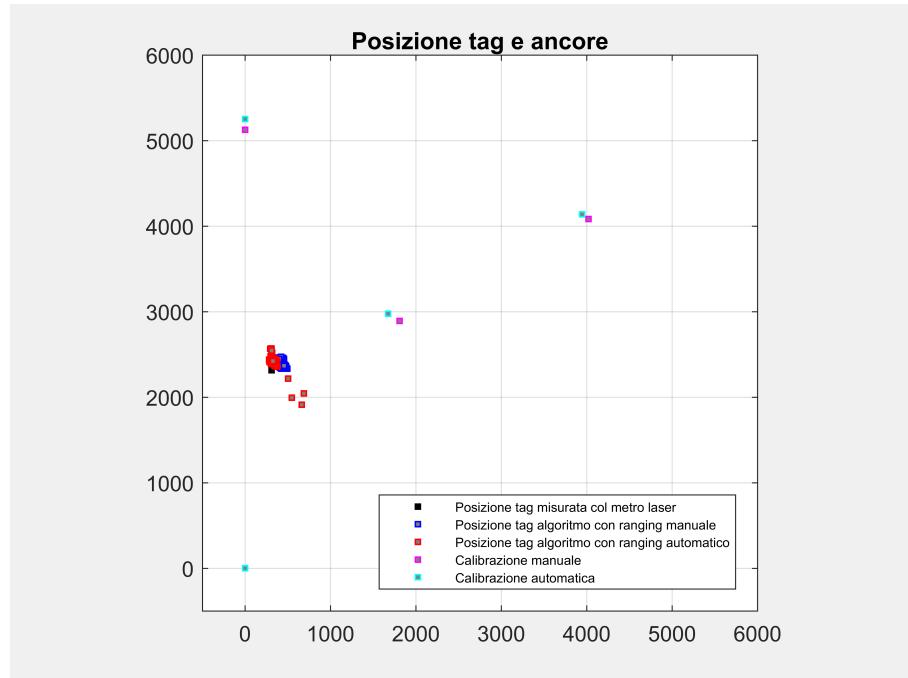


Figura 32: Posizione tag e ancora, vista in pianta.

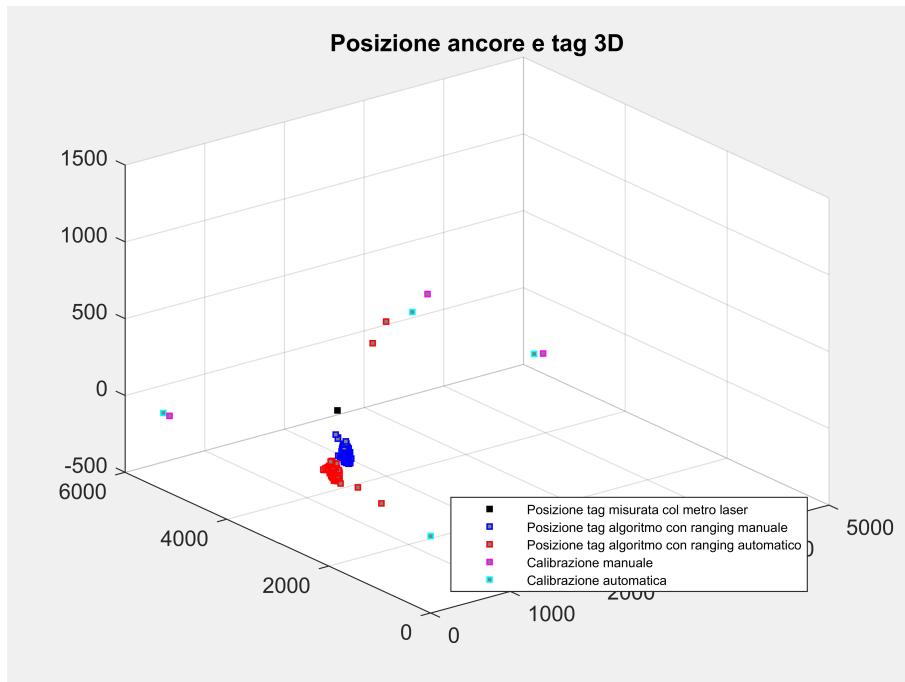


Figura 33: Posizione tag e ancore 3D.

## 8.1 Conclusioni

Per quanto riguarda l'errore di posizionamento del tag, sembrerebbe che la procedura automatica e manuale non differiscano di molto limitatamente alle coordinate x e y, ossia alle coordinate lungo il piano orizzontale. La stessa cosa non si può dire per l'altezza, infatti, come si evince dalla figura 33 e dai dati in tabella 8, entrambe le misurazioni sono fortemente affette da bias. In ogni caso, i valori risultati da una calibrazione manuale delle ancore risultano essere leggermente migliori.

| Tag         | x            | y             | z            |
|-------------|--------------|---------------|--------------|
| manuale     | $443 \pm 15$ | $2361 \pm 12$ | $113 \pm 25$ |
| automatico  | $333 \pm 16$ | $2421 \pm 17$ | $3 \pm 21$   |
| metro laser | $311 \pm 36$ | $2315 \pm 16$ | $422 \pm 60$ |

Tabella 8: Esperimento 5: Posizionamento del Tag

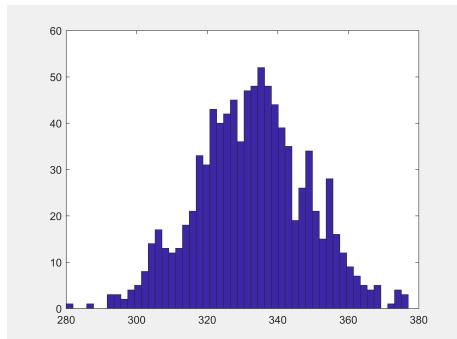


Figura 34: Coordinata x del tag, calibrazione automatica.

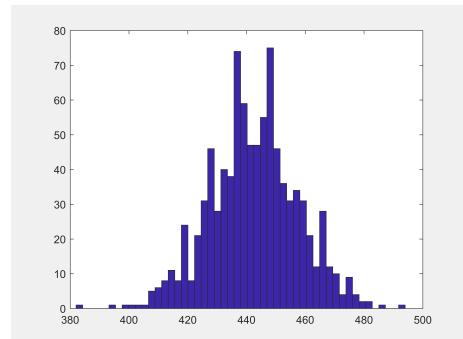


Figura 35: Coordinata x del tag, calibrazione manuale.

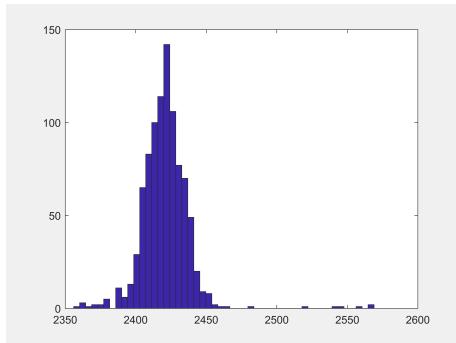


Figura 36: Coordinata y del tag, calibrazione automatica.

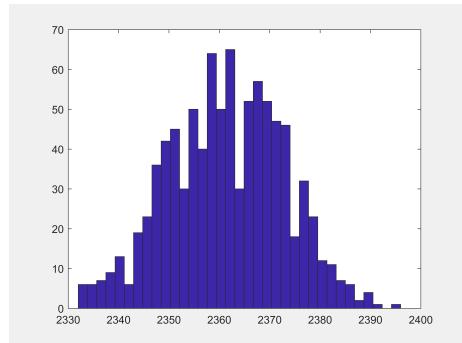


Figura 37: Coordinata y del tag, calibrazione manuale.

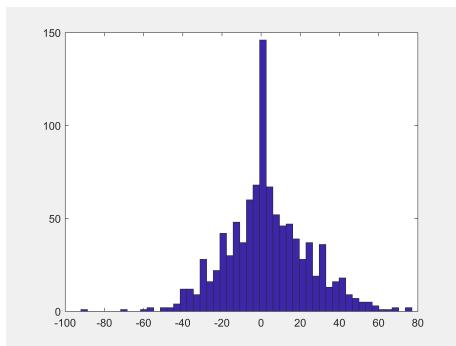


Figura 38: Coordinata z del tag, calibrazione automatica.

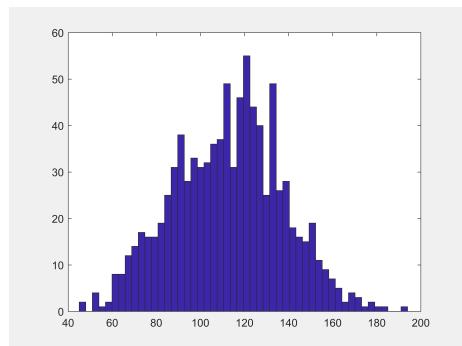


Figura 39: Coordinata z del tag, calibrazione manuale.

## 9 Esperimento 6: parametri UWB e frequenza di lavoro

In questo esperimento abbiamo voluto vedere come varia la frequenza di lavoro del sistema Pozyx al variare dei parametri UWB e del protocollo di ranging.

Il sistema Pozyx permette di utilizzare due differenti algoritmi per il positioning:

il RANGING\_PROTOCOL\_FAST e il RANGING\_PROTOCOL\_PRECISION.

Il primo dovrebbe avere una velocità di convergenza maggiore e, di conseguenza, riuscire ad avere una frequenza di lavoro superiore. In termini di precisione i due algoritmi forniscono, a detta dei costruttori, gli stessi risultati, anche se l'algoritmo FAST necessita di più tempo per essere accurato. In generale nei primi 100 ms l'informazione di ranging non sarà troppo accurata. Le prove sono state effettuate al variare dei seguenti parametri:

- **Canale:** sono stati utilizzati i valori 1,5,7;
- **Bitrate:** valori 110 kbit/sec, 850 kbit/sec e 6810 kbit/sec;
- **PRF:** 16 MHz e 64 MHz;
- **Preamble Length:** 64, 128, 256, 512, 1024, 1536, 2048, 4096 symbols;
- **Gain:** 11.5 dB;

Le prove sono state svolte in un ambiente indoor e dove sono presenti materiali metallici, pertanto i risultati potrebbero essere fortemente affetti da errori dovuti a rumore di misura e da disturbi elettromagnetici. I valori ottenuti sono quindi da ritenersi puramente indicativi.

È da sottolineare che in caso di fallimento di positioning del tag viene allungata di fatto la durata effettiva del positioning. Quindi se avessimo una percentuale di fallimento del 50%, la frequenza risulterebbe dimezzata e il tempo di positioning raddoppiato. Pertanto, qualora ci fossero rumori ambientali troppo marcati, anche le frequenze di lavoro risulterebbero notevolmente ridotte. La durata del positioning riportata in figura 40 è pertanto da intendersi come l'intervallo medio che intercorre tra due risultati positivi consecutivi dell'algoritmo di positioning.

### 9.1 Conclusioni

Dall'esperimento è emerso che esistono combinazioni dei parametri che sono migliori di altre in termini di percentuale di fallimenti e varianza dell'intervallo di aggiornamento del positioning. Tipicamente a valori di varianza elevati sono associate delle alte percentuali di fallimento. Ciò è coerente con il fatto che quando il Pozyx fallisce un algoritmo di positioning considera l'intervallo di aggiornamento come la somma dei singoli intervalli di aggiornamento. Perciò se avessimo  $n$  fallimenti consecutivi ciascuno avente tempo di calcolo  $\Delta T$ , si avrebbe un intervallo di aggiornamento complessivo pari a  $n\Delta T$ .

Un altro importante fattore di cui tenere conto quando si scelgono i parametri UWB è la dimensione caratteristica del sistema. Il sistema da noi analizzato aveva una lunghezza caratteristica dell'ordine di 5 m, dove si intende per lunghezza caratteristica la massima distanza tra due anchor. Tali distanze sono ritenute piccole rispetto ai range operativi delle antenne Pozyx. A detta dei costruttori infatti il sistema può funzionare a distanze di circa 100 m. È perciò naturale che i bitrates più elevati e Preamble length più corti, i quali sono più indicati per piccole distanze, diano dei risultati più soddisfacenti in ambienti simili a quello in esame.

Quello che appare controintuitivo ma è emerso dai dati è che la durata dell'intervallo di aggiornamento cresce circa linearmente con la lunghezza del Preamble Length, ma ciò non è seguito da una più ridotta percentuale di fallimenti. Sembrerebbe quindi più affidabile spingere il sistema al massimo delle sue prestazioni.

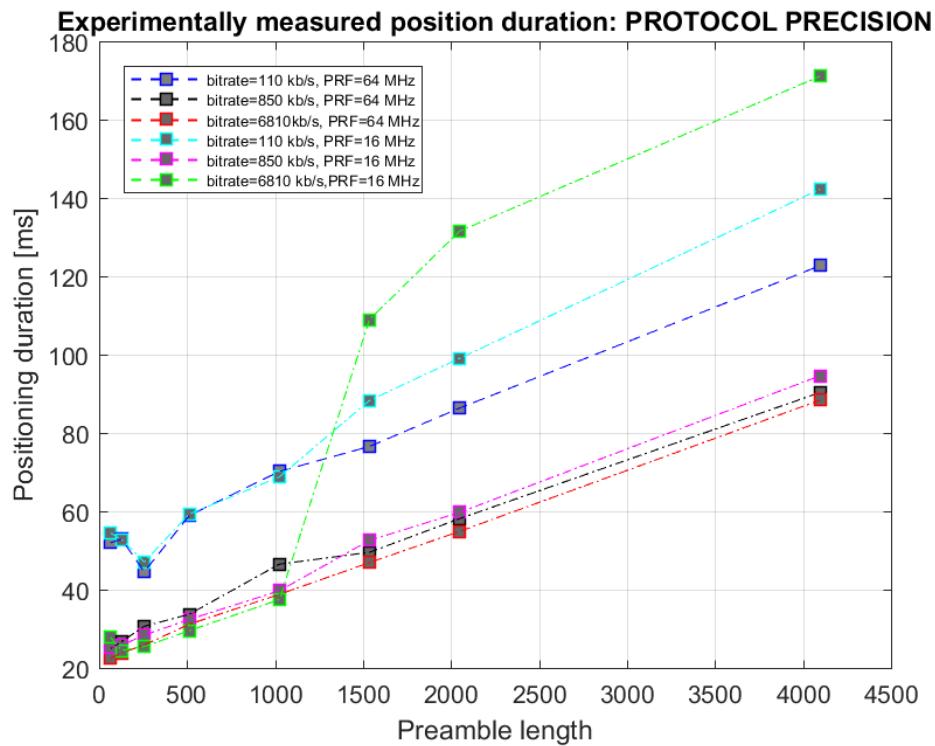


Figura 40: Durata del positioning: PROTOCOL PRECISION, canale 5

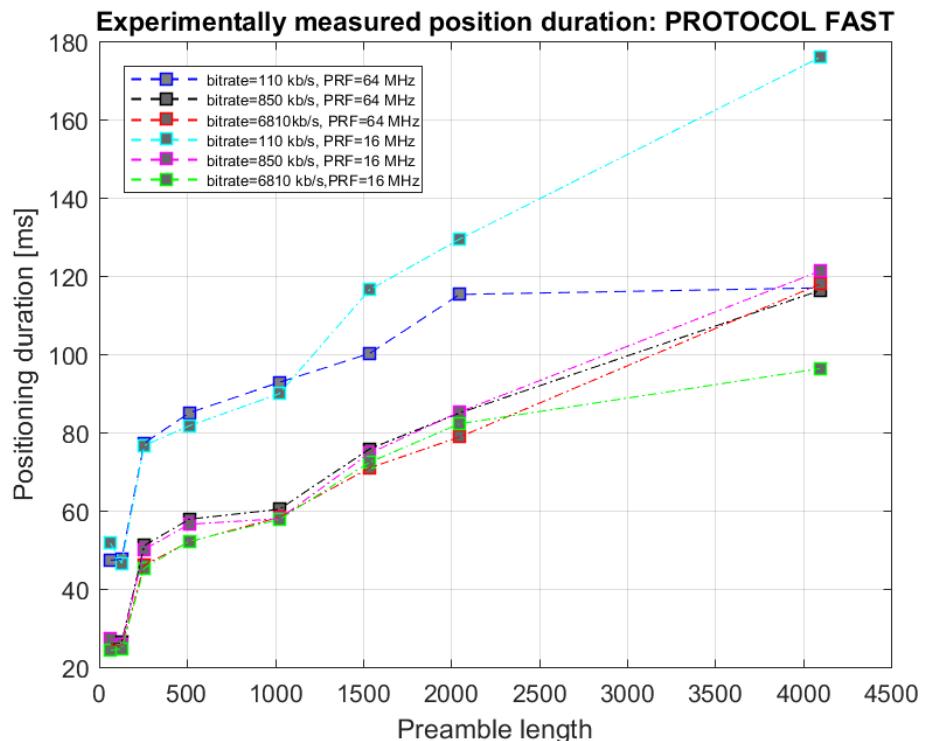


Figura 41: Durata del positioning: PROTOCOL FAST, canale 5

| PRF | Bitrate | Preamble length |       |       |       |       |        |        |        |
|-----|---------|-----------------|-------|-------|-------|-------|--------|--------|--------|
|     |         | 64              | 128   | 256   | 512   | 1024  | 1536   | 2048   | 4096   |
| 64  | 110     | 52±16           | 53±14 | 45±19 | 59±15 | 70±23 | 77±22  | 86±28  | 123±41 |
|     | 850     | 25±5            | 27±6  | 31±9  | 34±10 | 46±20 | 49±14  | 58±16  | 90±23  |
|     | 6810    | 22±5            | 24±5  | 26±6  | 31±7  | 38±10 | 46±12  | 54±15  | 88±24  |
| 16  | 64      | 54±19           | 53±15 | 47±19 | 59±17 | 69±21 | 88±40  | 99±49  | 142±76 |
|     | 850     | 25±5            | 26±6  | 28±6  | 32±9  | 39±9  | 52±19  | 60±20  | 95±35  |
|     | 6810    | 28±16           | 24±8  | 25±5  | 29±6  | 37±8  | 109±52 | 132±56 | 171±95 |

Tabella 9: Durata del positioning (valor medio e varianza): PROTOCOL PRECISION, canale 5, valori in [ms]

| PRF | Bitrate | Preamble length |       |       |       |       |        |        |        |
|-----|---------|-----------------|-------|-------|-------|-------|--------|--------|--------|
|     |         | 64              | 128   | 256   | 512   | 1024  | 1536   | 2048   | 4096   |
| 64  | 110     | 47±35           | 48±34 | 77±35 | 85±35 | 93±36 | 100±36 | 115±28 | 116±29 |
|     | 850     | 26±28           | 26±29 | 51±35 | 57±37 | 60±36 | 76±38  | 85±36  | 116±26 |
|     | 6810    | 26±29           | 25±28 | 46±34 | 51±35 | 58±35 | 70±37  | 78±36  | 117±31 |
| 16  | 110     | 51±37           | 46±34 | 76±35 | 81±36 | 89±33 | 116±64 | 129±52 | 175±94 |
|     | 850     | 27±30           | 26±29 | 50±35 | 56±36 | 58±34 | 74±39  | 85±40  | 121±36 |
|     | 6810    | 24±27           | 25±28 | 45±34 | 52±35 | 58±35 | 72±39  | 82±43  | 96±47  |

Tabella 10: Durata del positioning (valor medio e varianza): PROTOCOL FAST, canale 5, valori in [ms]

| PRF | Bitrate | Preamble length |     |      |     |      |      |      |      |
|-----|---------|-----------------|-----|------|-----|------|------|------|------|
|     |         | 64              | 128 | 256  | 512 | 1024 | 1536 | 2048 | 4096 |
| 64  | 110     | 5.8             | 4.4 | 54   | 3.6 | 7.8  | 5.0  | 6.6  | 8.2  |
|     | 850     | 0.0             | 0.0 | 6.2  | 5.6 | 15.4 | 3.8  | 3.8  | 2.2  |
|     | 6810    | 0.2             | 0.0 | 1.0  | 0.6 | 3.2  | 2.2  | 3.2  | 2.8  |
| 16  | 110     | 8.6             | 5.6 | 34.2 | 5.6 | 6.4  | 21   | 22.6 | 25.4 |
|     | 850     | 0.4             | 0.2 | 0.8  | 3   | 0.8  | 11   | 8.6  | 9.4  |
|     | 6810    | 25.8            | 7.4 | 0.6  | 0.0 | 1.0  | 12.2 | 13.4 | 64.6 |

Tabella 11: Percentuale di fallimenti nel Positioning, PROTOCOL\_PRECISION, Canale 5.

| PRF | Bitrate | Preamble length |     |     |     |      |      |      |      |
|-----|---------|-----------------|-----|-----|-----|------|------|------|------|
|     |         | 64              | 128 | 256 | 512 | 1024 | 1536 | 2048 | 4096 |
| 64  | 110     | 2.6             | 0.8 | 0.6 | 0.4 | 2.0  | 3.0  | 1.4  | 2.8  |
|     | 850     | 0.6             | 0.6 | 0.8 | 1.8 | 3.8  | 2.8  | 2.2  | 1.0  |
|     | 6810    | 1.2             | 0.6 | 0.2 | 0.2 | 1.4  | 2.6  | 1.8  | 2.8  |
| 16  | 110     | 11.6            | 1.4 | 0.0 | 0.6 | 0.4  | 18.4 | 15.8 | 20.2 |
|     | 850     | 9.6             | 0.8 | 2.0 | 1.6 | 3.0  | 5.4  | 6.2  | 6.2  |
|     | 6810    | 9.0             | 0.8 | 0.2 | 0.4 | 0.4  | 10.8 | 8.8  | 9.0  |

Tabella 12: Percentuale di fallimenti nel Positioning, PROTOCOL\_FAST, Canale 5.

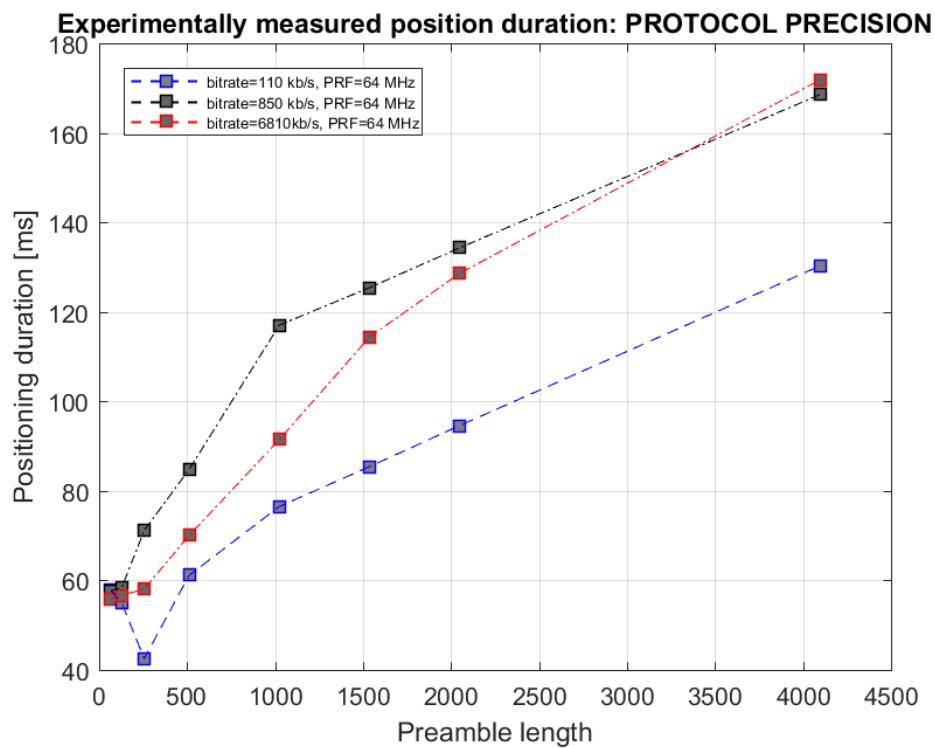


Figura 42: Durata del positioning: PROTOCOL PRECISION canale 1

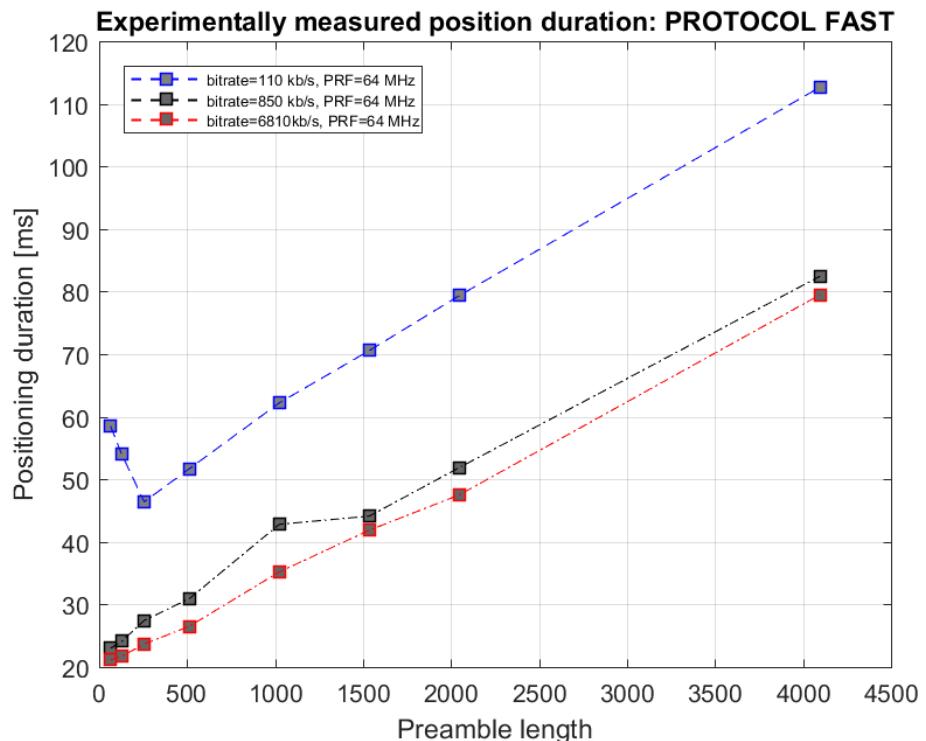


Figura 43: Durata del positioning: PROTOCOL FAST, canale 1

| PRF | Bitrate | Preamble length |       |       |       |        |        |        |        |
|-----|---------|-----------------|-------|-------|-------|--------|--------|--------|--------|
|     |         | 64              | 128   | 256   | 512   | 1024   | 1536   | 2048   | 4096   |
| 64  | 110     | 58±24           | 55±17 | 42±32 | 61±19 | 77±34  | 85±35  | 95±39  | 130±56 |
|     | 850     | 58±35           | 58±35 | 71±42 | 85±44 | 117±56 | 125±46 | 134±58 | 168±68 |
|     | 6810    | 56±36           | 57±36 | 58±35 | 70±36 | 91±41  | 114±44 | 129±50 | 172±76 |

Tabella 13: Durata del positioning (valor medio e varianza): PROTOCOL\_PRECISION, Canale 1, valori in [ms]

| PRF | Bitrate | Preamble length |       |       |       |       |       |        |        |
|-----|---------|-----------------|-------|-------|-------|-------|-------|--------|--------|
|     |         | 64              | 128   | 256   | 512   | 1024  | 1536  | 2048   | 4096   |
| 64  | 110     | 59±26           | 54±16 | 46±31 | 52±18 | 62±28 | 71±32 | 79±34  | 112±43 |
|     | 850     | 23±5            | 24±5  | 27±10 | 31±11 | 43±22 | 44±17 | 52pm19 | 82±30  |
|     | 6810    | 21±5            | 22±5  | 24±5  | 26±6  | 35±7  | 42±16 | 48±17  | 79±32  |

Tabella 14: Durata del positioning (valor medio e varianza): PROTOCOL\_FAST, Canale 1, valori in [ms]

| PRF | Bitrate | Preamble length |     |      |      |      |      |      |      |
|-----|---------|-----------------|-----|------|------|------|------|------|------|
|     |         | 64              | 128 | 256  | 512  | 1024 | 1536 | 2048 | 4096 |
| 64  | 110     | 14.8            | 8   | 84.4 | 6.4  | 16.0 | 15.6 | 16.0 | 31.6 |
|     | 850     | 0.2             | 0.8 | 13.6 | 10.4 | 22.2 | 10.8 | 14.4 | 9.8  |
|     | 6810    | 0.0             | 0.0 | 0.6  | 1.8  | 6.4  | 9.6  | 10.8 | 11.6 |

Tabella 15: Percentuale di fallimenti nel Positioning, PROTOCOL\_PRECISION, Canale 1

| PRF | Bitrate | Preamble length |     |      |      |      |      |      |      |
|-----|---------|-----------------|-----|------|------|------|------|------|------|
|     |         | 64              | 128 | 256  | 512  | 1024 | 1536 | 2048 | 4096 |
| 64  | 110     | 16.4            | 5.8 | 81.4 | 11.4 | 17.4 | 17.0 | 16.6 | 14.2 |
|     | 850     | 0.4             | 0.6 | 10.0 | 9.4  | 24.4 | 11.2 | 13.2 | 10.2 |
|     | 6810    | 0.2             | 0.2 | 1.6  | 1.0  | 10.4 | 11.2 | 9.2  | 10.8 |

Tabella 16: Percentuale di fallimenti nel Positioning, PROTOCOL\_FAST, Canale 1

## 10 Esperimento 7: Multipositioning

In questa sezione viene descritto il test di multiposizionamento, ovvero il positioning contemporaneo di due tag.

Il setup sperimentale è così definito: sono necessari 7 dispositivi, di cui 4 ancore, un dispositivo collegato alla seriale e due tag, alimentati mediante power bank, sono fissati all'estremità opposte di un asta rigida (1197 [mm]). Quest'ultima viene poi fatta muovere all'interno della stanza. La finalità di questa prova è quella di verificare quanto la distanza (la quale dovrebbe essere costante) fra i due tag venga influenzata dal movimento. In questa prova le ancore sono state disposte in configurazione tetraedrica e le loro coordinate sono le medesime dell'esperimento 5.

Indicato con

$$\vec{d} = \vec{p}_1 - \vec{p}_2$$

il vettore della distanza relativa tra i due tag, si riportano gli istogrammi relativi alle tre componenti del vettore  $\vec{d}$  e del suo modulo. Di seguito se ne riporta il valore medio e la varianza relativa ad un'acquisizione di 1000 campioni, il plot della traiettoria seguita e l'istogramma relativo al modulo della distanza fra i due tag.

1. Valor medio: 1139 mm
2. Varianza: 188 mm

E' da notare però che nel corso del moto l'asta non ha semplicemente traslato parallelamente a se stesso ma ha anche ruotato, quindi i valori delle componenti del vettore non si addenseranno attorno ad un valore medio.

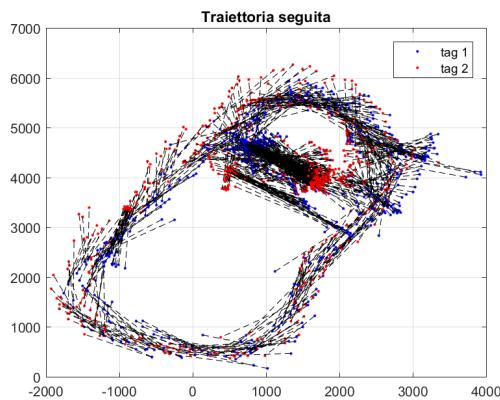


Figura 44: Traiettoria seguita dall'asta

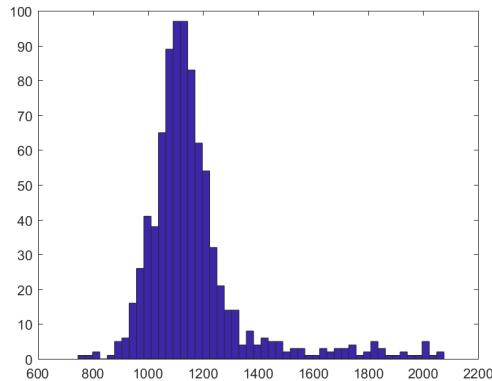


Figura 45: Istogramma della distanza fra i due tag

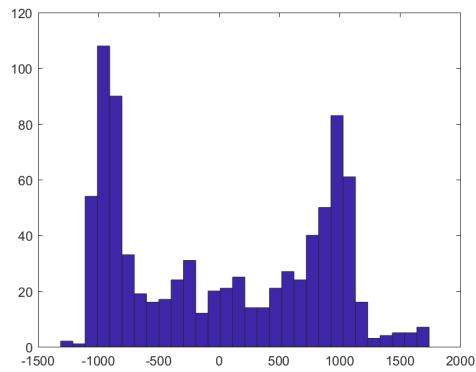


Figura 46: Istogramma distanza relativa x

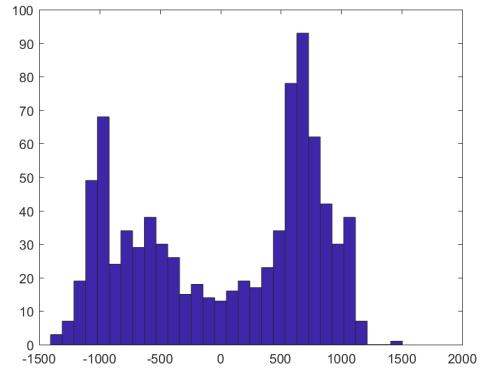


Figura 47: Istogramma distanza relativa y

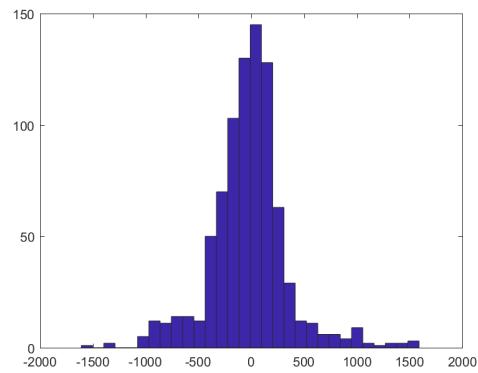


Figura 48: Istogramma distanza relativa z

## 11 Esperimento 8: massima distanza di comunicazione do-Positioning

### 11.1 Descrizione dell'esperimento

In questo esperimento siamo interessati ad analizzare il comportamento del tag al variare della distanza tra questo e l'insieme delle ancore. In particolare vogliamo analizzare la percentuale di fallimenti nel positioning, l'intensità del segnale e la varianza della posizione.

Abbiamo effettuato un ciclo di 1000 acquisizioni alle distanze di 5, 10, 15, 20, 25, 30, 35 e 40 m. Non siamo potuti andare a distanze superiori per via dei limiti fisici imposti dall'edificio. Le distanze sono state misurate dal centro della configurazione delle ancore, le quali sono state poste nella classica configurazione tetraedrica. Le distanze a cui abbiamo effettuato il positioning sono state misurate con una rotella metrica posta a terra in linea retta.

### 11.2 Caratterizzazione di un fallimento nel positioning

In caso di fallimento del Positioning il sistema Pozyx può comportarsi in due modi:

- nel caso in cui una delle ancore della lista interna del dispositivo (cioè le ancore che il dispositivo sa di dover interrogare per effettuare il positioning) sia spenta, NON viene segnalato un POZYX\_FAILURE. Il valore di return della funzione doPositioning può infatti essere POZYX\_SUCCESS o POZYX\_FAILURE a seconda che il procedimento di Positioning abbia avuto successo o meno. Nel caso in cui un'ancora sia spenta però, NON verrà segnalato un POZYX\_FAILURE. E' comunque possibile determinare un fallimento del positioning in quanto le coordinate del tag risulteranno essere poste di default a (0,0,0), la distanza dall'ancora spenta è posta a 0, l'RSS (Received signal strength) tra il tag e l'ancora è posto a 0 e infine la timestamp è posta al valore di default pari a 1705985.
- nel caso in cui tutte le ancore siano correttamente accese ma si fallisca a ricevere la distanza da una delle ancore (signal loss), il sistema Pozyx notificherà l'evento tramite un POZYX\_FAILURE. Se si vanno a leggere i dati relativi alle distanze delle ancore, risulterà evidente che i dati relativi all'ancora con cui è fallita la comunicazione sono identici ai precedenti. Ciò significa che quando il doPositioning chiama internamente la doRanging, quando una delle doRanging fallisce, viene segnalato un POZYX\_FAILURE della procedura di positioning.
- un'altra causa di fallimento potrebbe essere dovuto ai parametri UWB che possono essere impostati differentemente sui vari dispositivi. I dispositivi con parametri UWB differenti da quelli del dispositivo che effettua il positioning appariranno come se fossero spenti, come nel caso 1. Conseguentemente le coordinate del tag ottenute tramite il doPositioning risulteranno essere (0,0,0).

### 11.3 Plot grafici e tabelle dei risultati

| Distanza [m] | x [mm]       | y [mm]       | z [mm]      |
|--------------|--------------|--------------|-------------|
| 5            | 254±576      | 6648 ±428    | -911 ±824   |
| 10           | -546 ±1523   | 1126 ±965    | -650 ±1759  |
| 15           | -1852 ±2115  | 1592 ±1494   | -288 ±3515  |
| 20           | -1181 ±4586  | 1951 ±3642   | -3168 ±3603 |
| 25           | -6173 ±8393  | 21046 ±6769  | -2814 ±3269 |
| 30           | -7572 ±10504 | 25081 ±12121 | -1949 ±3906 |
| 35           | -2934 ±6726  | 31365 ±10785 | -1955 ±4171 |
| 40           | -1632 ±4923  | 33144 ±10365 | -2405 ±3935 |

Tabella 17: Posizione media e varianza del tag in mm in funzione della distanza dal centro della configurazione delle ancore.

| Distanza [m] | Percentuale di fallimenti del positioning |
|--------------|-------------------------------------------|
| 5            | 7.8 %                                     |
| 10           | 5.4 %                                     |
| 15           | 5.2 %                                     |
| 20           | 12 %                                      |
| 25           | 21.4 %                                    |
| 30           | 30.4 %                                    |
| 35           | 73.6 %                                    |
| 40           | 34.2 %                                    |

Tabella 18: Percentuale di fallimenti nel positioning al variare della distanza tra il tag e il centro della configurazione delle ancore.

| Distanza [m] | $RSS_{A0}[dBm]$ | $RSS_{A1}[dBm]$ | $RSS_{A2}[dBm]$ | $RSS_{A3}[dBm]$ |
|--------------|-----------------|-----------------|-----------------|-----------------|
| 5            | -89             | -83             | -85             | -84             |
| 10           | -90             | -89             | -90             | -86             |
| 15           | -94             | -89             | -92             | -92             |
| 20           | -96             | -93             | -97             | -95             |
| 25           | -98             | -94             | -97             | -95             |
| 30           | -99             | -92             | -99             | -97             |
| 35           | -98             | -98             | -99             | -94             |
| 40           | -98             | -98             | -98             | -97             |

Tabella 19: RSS (Received signal strength) in funzione della distanza tra il tag e il centro della configurazione delle ancore.

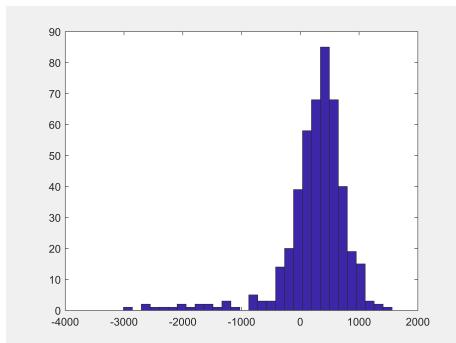


Figura 49: Istogramma coordinata x del tag: distanza 5 m

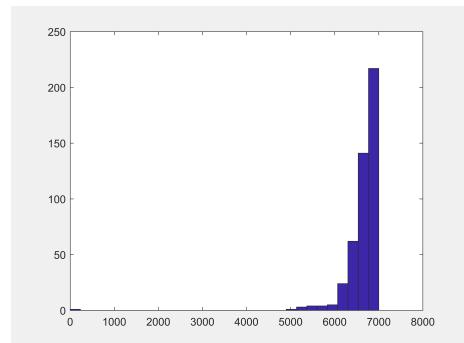


Figura 50: Istogramma coordinata y del tag: distanza 5 m

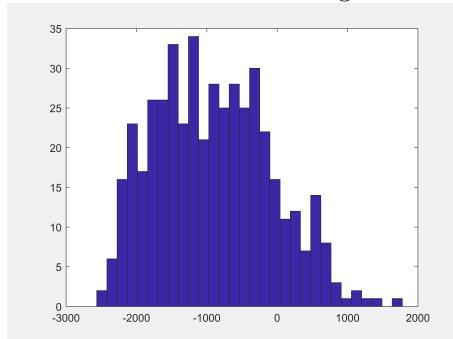


Figura 51: Istogramma coordinata z del tag: distanza 5 m

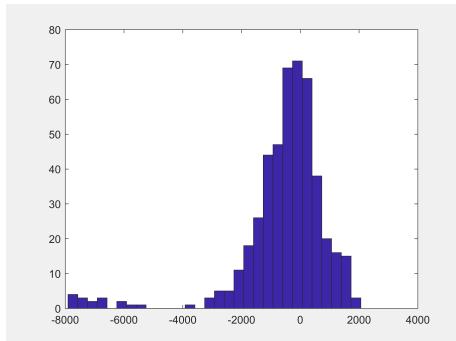


Figura 52: Istogramma coordinata x del tag: distanza 10 m

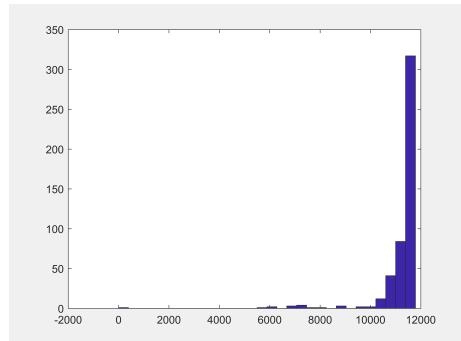


Figura 53: Istogramma coordinata y del tag: distanza 10 m

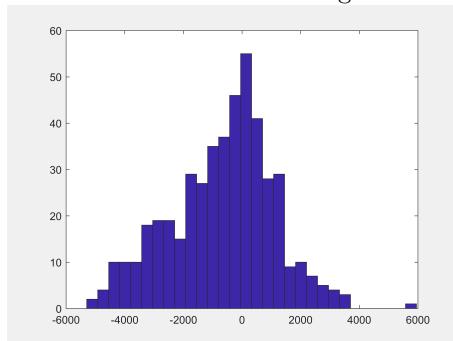


Figura 54: Istogramma coordinata z del tag: distanza 10 m

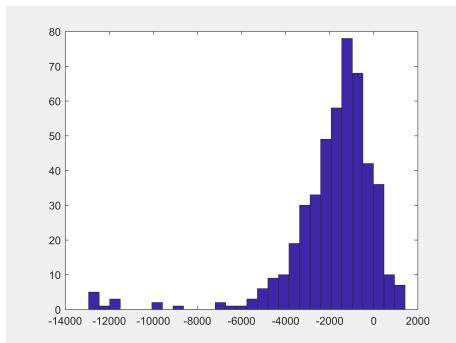


Figura 55: Istogramma coordinata x del tag: distanza 15 m

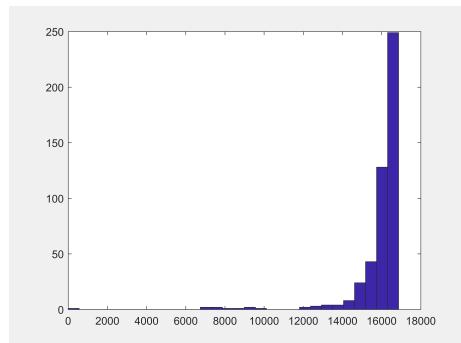


Figura 56: Istogramma coordinata y del tag: distanza 15 m

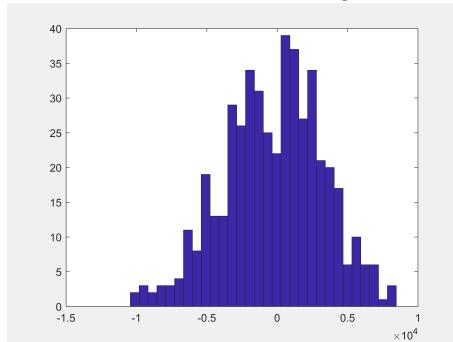


Figura 57: Istogramma coordinata z del tag: distanza 15 m

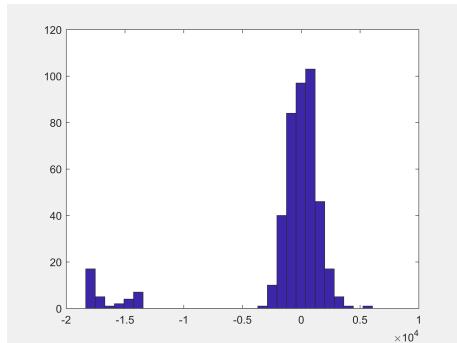


Figura 58: Istogramma coordinata x del tag: distanza 20 m

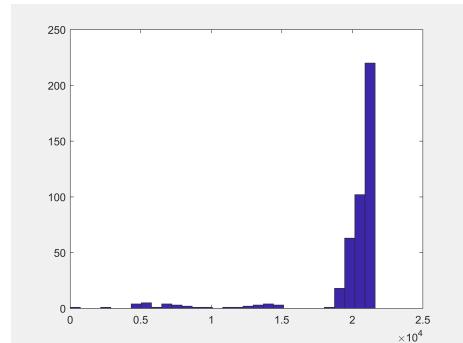


Figura 59: Istogramma coordinata y del tag: distanza 20 m

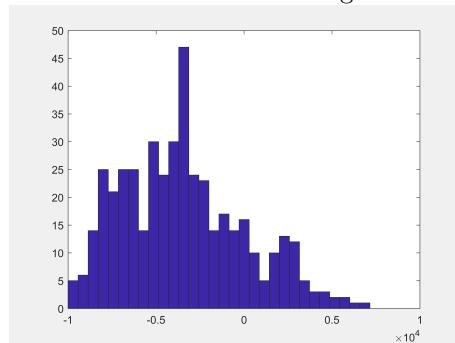


Figura 60: Istogramma coordinata z del tag: distanza 20 m

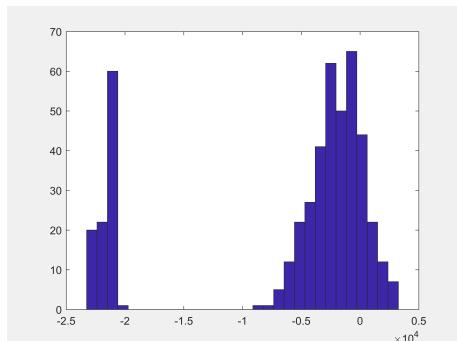


Figura 61: Istogramma coordinata x del tag: distanza 25 m

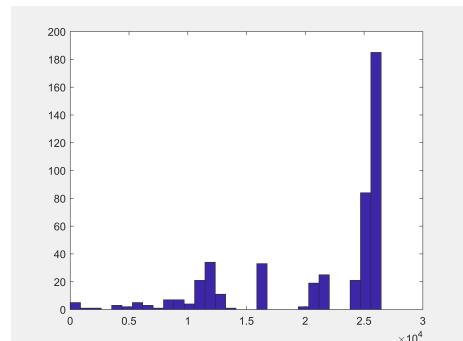


Figura 62: Istogramma coordinata y del tag: distanza 25 m

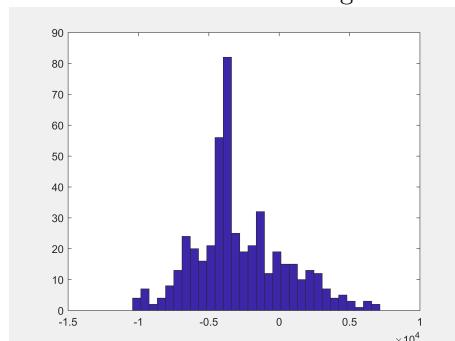


Figura 63: Istogramma coordinata z del tag: distanza 25 m

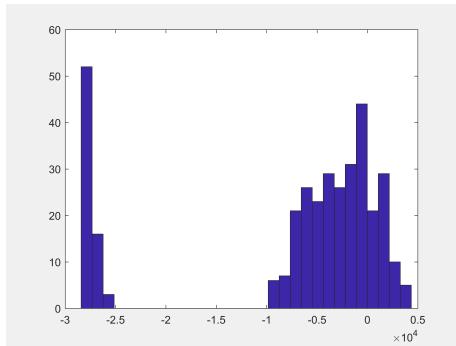


Figura 64: Istogramma coordinata x del tag: distanza 30 m

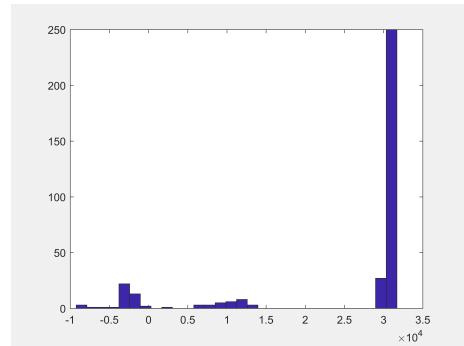


Figura 65: Istogramma coordinata y del tag: distanza 30 m

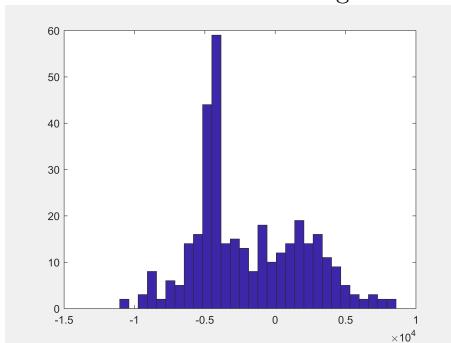


Figura 66: Istogramma coordinata z del tag: distanza 30 m

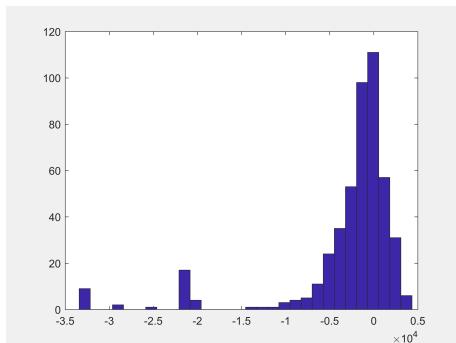


Figura 67: Istogramma coordinata x del tag: distanza 35 m

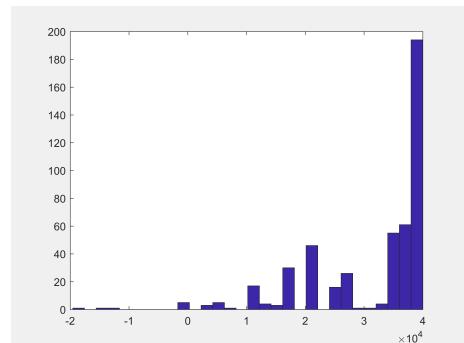


Figura 68: Istogramma coordinata y del tag: distanza 35 m

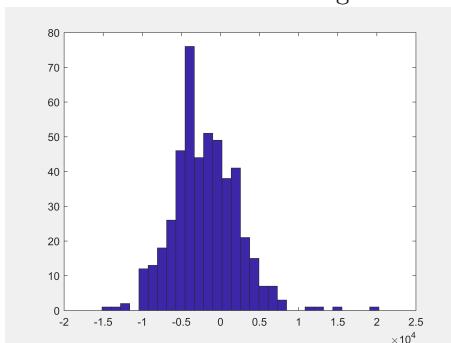


Figura 69: Istogramma coordinata z del tag: distanza 35 m

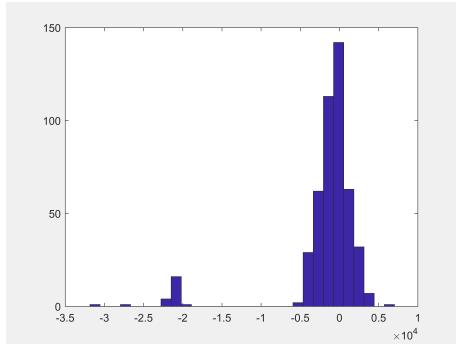


Figura 70: Istogramma coordinata x del tag: distanza 40 m

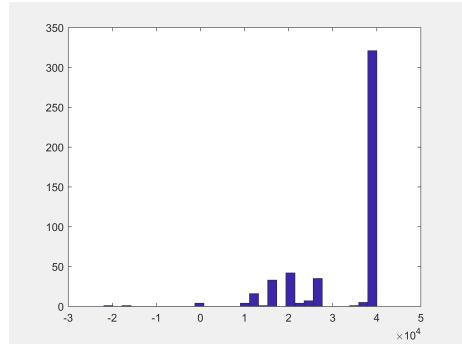


Figura 71: Istogramma coordinata y del tag: distanza 40 m

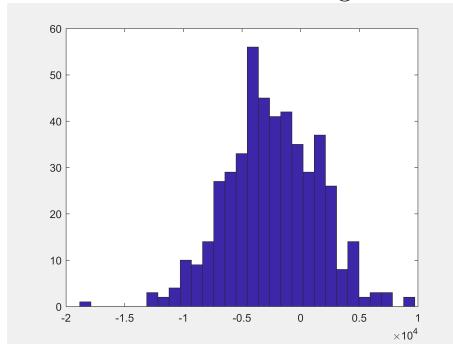


Figura 72: Istogramma coordinata z del tag: distanza 40 m

## 12 Esperimento 9: test outdoor

In questa sezione viene descritto un test finalizzato a valutare a quale distanza massima viene effettuato il positioning. Pertanto verranno riportati, in funzione della distanza dall'ancora 0 (origine del sistema di riferimento), la percentuale di fallimenti del positioning e gli indici statistici relativi alle coordinate del tag. Si riporta inoltre la distanza calcolata con il teorema di Pitagora applicato ai valori medi su  $x$  e  $y$ . Tale distanza deve essere confrontata con quella ottenuta con la rotella metrica (prima colonna della tabella 12.1). In questo esperimento ci siamo situati in un grande spazio aperto, effettuato una calibrazione automatica delle ancore con i parametri di default UWB. Le ancore sono state disposte a una distanza di circa 20 m l'una dall'altra secondo la configurazione che è possibile vedere in figura 117

## 12.1 Plot grafici e tabelle dei risultati

| Distanza [m]    | x [mm]           | y [mm]          | z [mm]             | %fallimenti | $\mu_x^2 + \mu_y^2$ [m] |
|-----------------|------------------|-----------------|--------------------|-------------|-------------------------|
| 5               | 3218 $\pm$ 57    | 3548 $\pm$ 199  | 187 $\pm$ 59       | 1.0         | 4.79                    |
| 10              | 6394 $\pm$ 32    | 7636 $\pm$ 32   | 136 $\pm$ 68       | 1.0         | 9.96                    |
| 15              | 9377 $\pm$ 100   | 11691 $\pm$ 80  | 27 $\pm$ 73        | 1.2         | 14.98                   |
| 20              | 12490 $\pm$ 44   | 14983 $\pm$ 53  | -111 $\pm$ 79      | 0.2         | 19.50                   |
| 25              | 15422 $\pm$ 53   | 18753 $\pm$ 40  | -476 $\pm$ 240     | 0.6         | 24.28                   |
| 30              | 18596 $\pm$ 101  | 22518 $\pm$ 75  | -1635 $\pm$ 588    | 22.6        | 29.20                   |
| 35              | 21856 $\pm$ 109  | 26340 $\pm$ 114 | -1891 $\pm$ 1133   | 0.8         | 34.22                   |
| 40              | 24799 $\pm$ 174  | 29891 $\pm$ 210 | -3797 $\pm$ 1602   | 2.0         | 38.83                   |
| 45              | 28065 $\pm$ 401  | 33766 $\pm$ 305 | -4894 $\pm$ 2272   | 0.6         | 43.90                   |
| 50 <sub>1</sub> | 30633 $\pm$ 912  | 36813 $\pm$ 525 | -9069 $\pm$ 3771   | 0.6         | 47.89                   |
| 50 <sub>2</sub> | 30419 $\pm$ 736  | 36866 $\pm$ 434 | -9923 $\pm$ 3342   | 18.8        | 47.79                   |
| 55              | 33248 $\pm$ 1122 | 39964 $\pm$ 646 | -13454 $\pm$ 4295  | 1.2         | 51.98                   |
| 60 <sub>1</sub> | 37243 $\pm$ 1042 | 44358 $\pm$ 379 | -13735 $\pm$ 2788  | 0.6         | 57.92                   |
| 60 <sub>2</sub> | 37440 $\pm$ 542  | 44520 $\pm$ 288 | -13855 $\pm$ 1345  | 1.4         | 58.17                   |
| 65              | 40165 $\pm$ 809  | 47636 $\pm$ 423 | -18342 $\pm$ 1837  | 1.4         | 62.30                   |
| 70              | 43925 $\pm$ 814  | 51127 $\pm$ 602 | -20261 $\pm$ 14138 | 44.8        | 67.40                   |

Tabella 20: Posizione media e varianza del tag in mm in funzione della distanza dal centro della configurazione delle ancore.

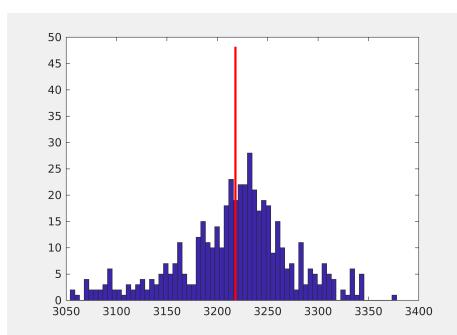


Figura 73: Istogramma coordinata x del tag: distanza 5 m

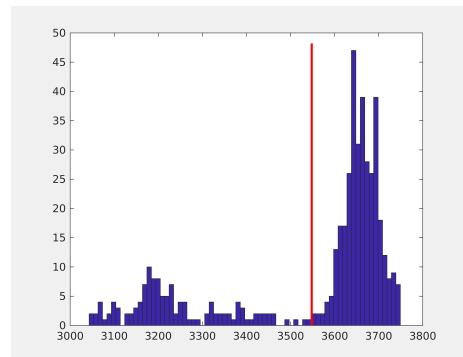


Figura 74: Istogramma coordinata y del tag: distanza 5 m

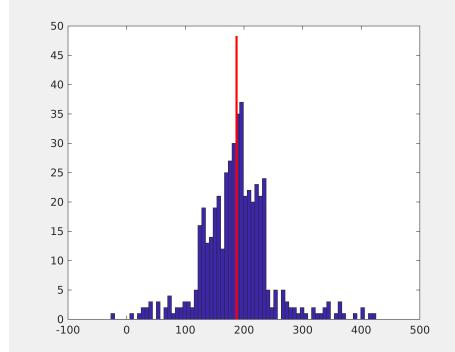


Figura 75: Istogramma coordinata z del tag: distanza 5 m

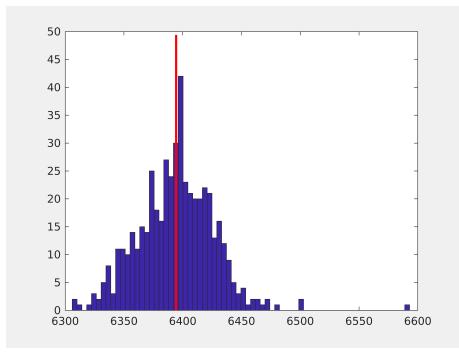


Figura 76: Istogramma coordinata x del tag; distanza 10 m

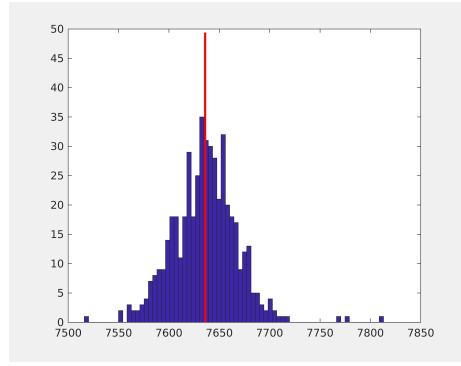


Figura 77: Istogramma coordinata y del tag; distanza 10 m

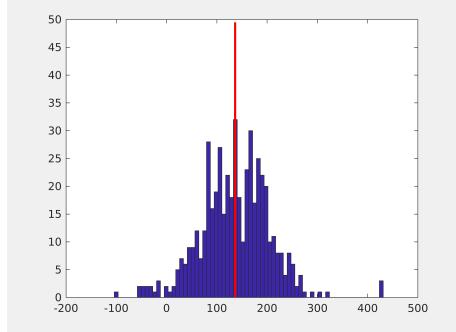


Figura 78: Istogramma coordinata z del tag; distanza 10 m

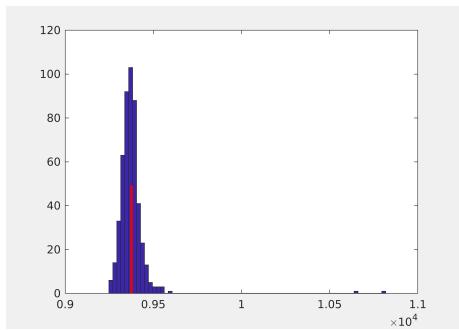


Figura 79: Istogramma coordinata x del tag; distanza 15 m

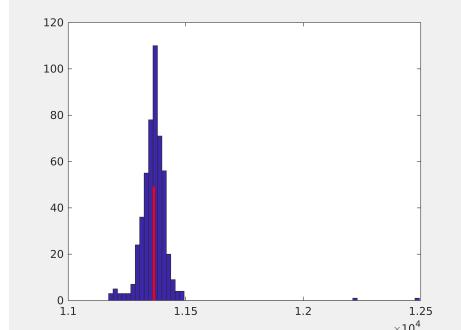


Figura 80: Istogramma coordinata y del tag; distanza 15 m

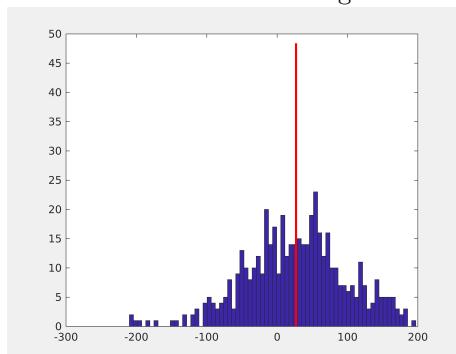


Figura 81: Istogramma coordinata z del tag; distanza 15 m

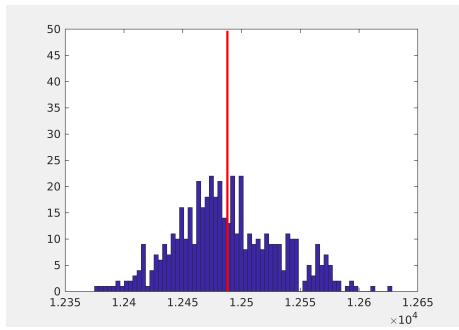


Figura 82: Istogramma coordinata x del tag: distanza 20 m

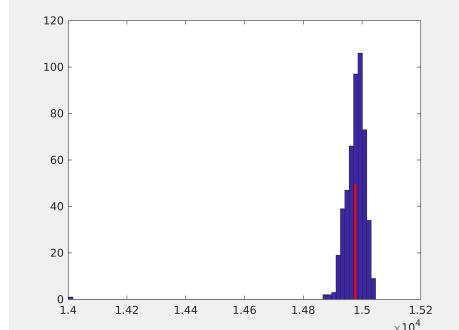


Figura 83: Istogramma coordinata y del tag: distanza 20 m

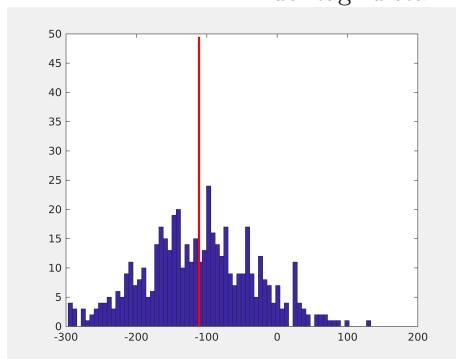


Figura 84: Istogramma coordinata z del tag: distanza 20 m

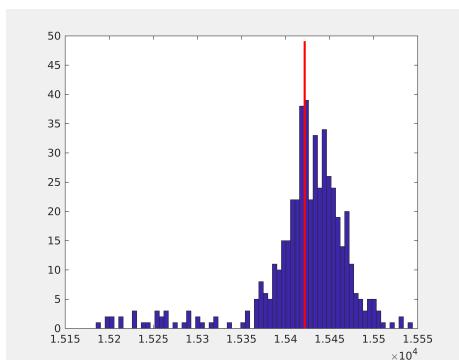


Figura 85: Istogramma coordinata x del tag: distanza 25 m

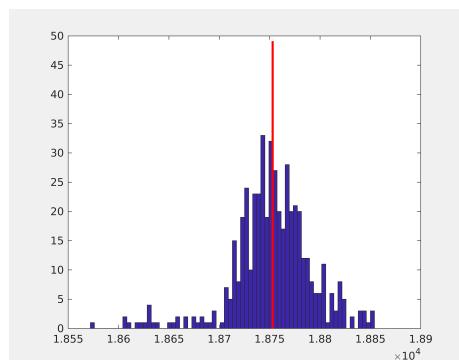


Figura 86: Istogramma coordinata y del tag: distanza 25 m

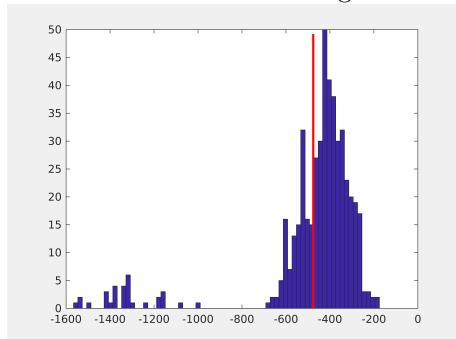


Figura 87: Istogramma coordinata z del tag: distanza 25 m

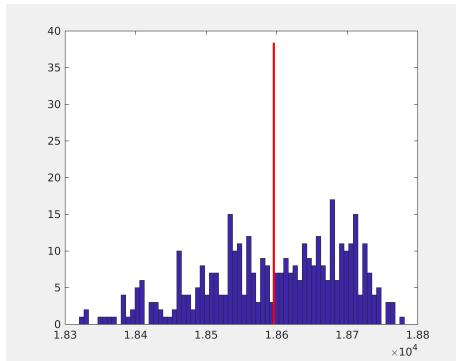


Figura 88: Istogramma coordinata x del tag; distanza 30 m

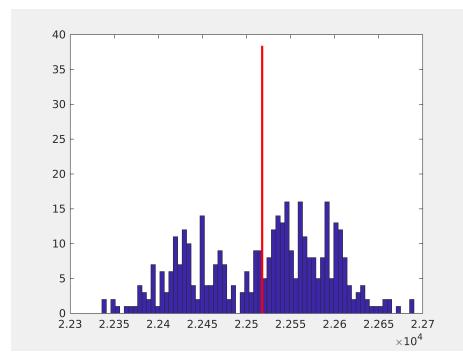


Figura 89: Istogramma coordinata y del tag; distanza 30 m

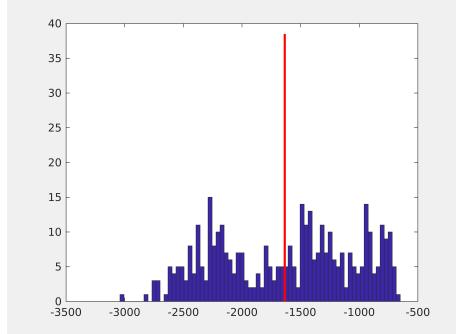


Figura 90: Istogramma coordinata z del tag; distanza 30 m

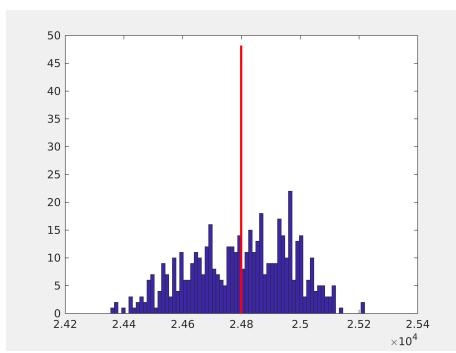


Figura 91: Istogramma coordinata x del tag; distanza 40 m

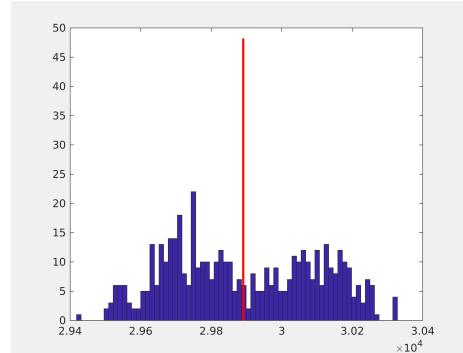


Figura 92: Istogramma coordinata y del tag; distanza 40 m

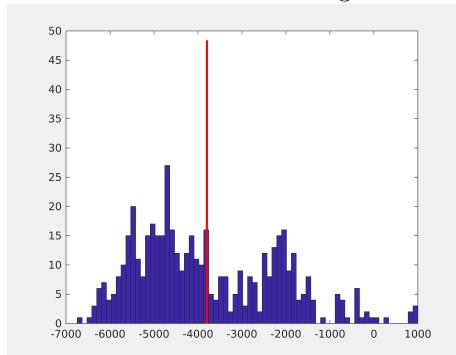


Figura 93: Istogramma coordinata z del tag; distanza 40 m

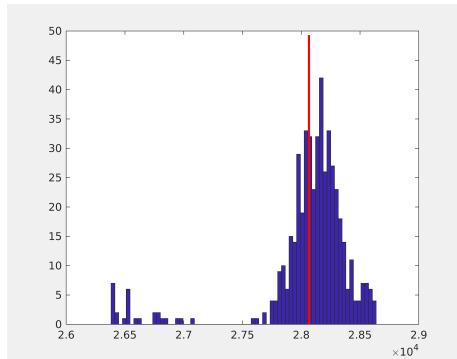


Figura 94: Istogramma coordinata x del tag; distanza 45 m

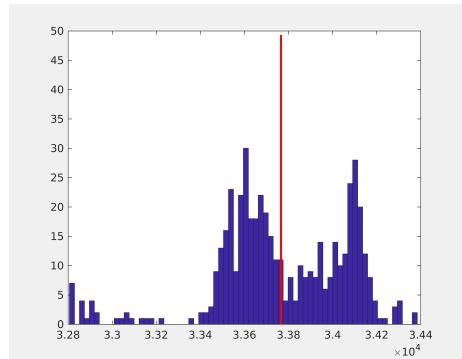


Figura 95: Istogramma coordinata y del tag; distanza 45 m

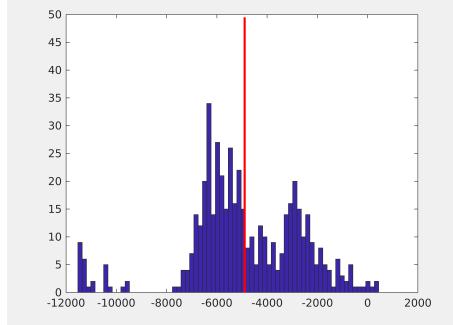


Figura 96: Istogramma coordinata z del tag; distanza 45 m

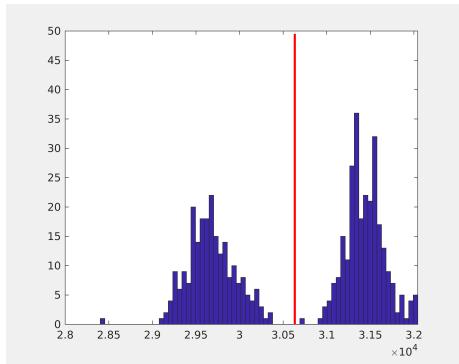


Figura 97: Istogramma coordinata x del tag: distanza 50 m, orientazione 0 gradi

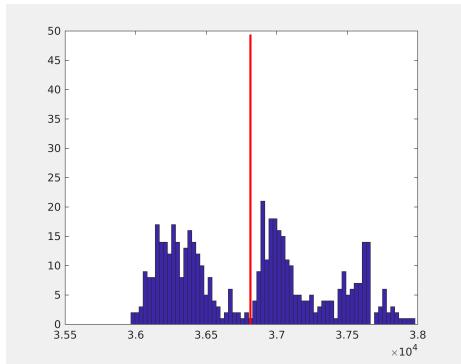


Figura 98: Istogramma coordinata y del tag: distanza 50 m, orientazione 0 gradi

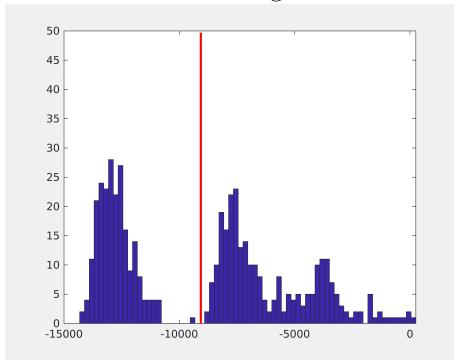


Figura 99: Istogramma coordinata z del tag: distanza 50 m, orientazione 0 gradi

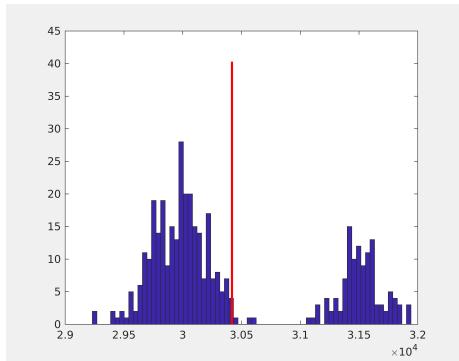


Figura 100: Istogramma coordinata x del tag; distanza 50 m, orientazione 90 gradi

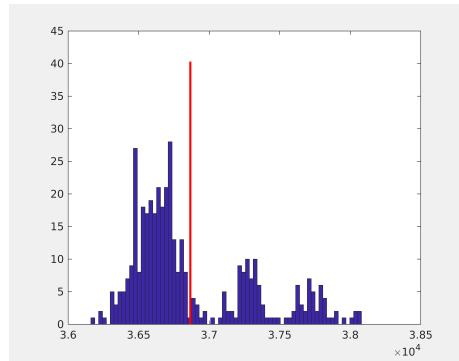


Figura 101: Istogramma coordinata y del tag; distanza 50 m, orientazione 90 gradi

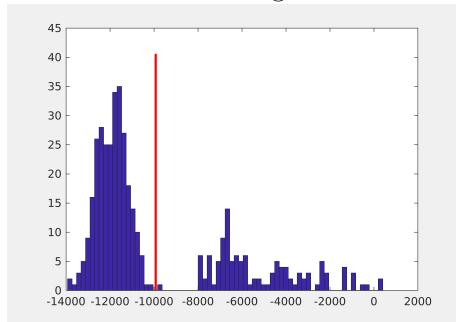


Figura 102: Istogramma coordinata z del tag; distanza 50 m, orientazione 90 gradi

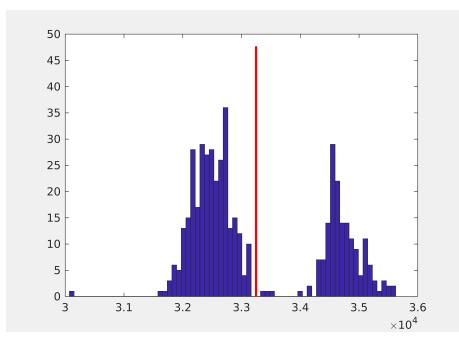


Figura 103: Istogramma coordinata x del tag; distanza 55 m

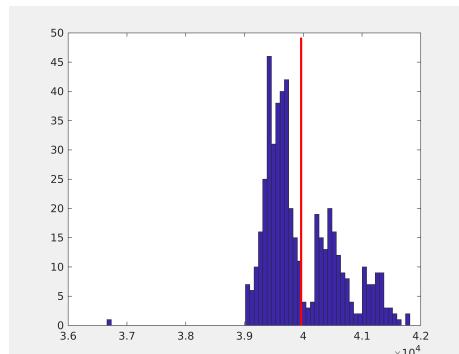


Figura 104: Istogramma coordinata y del tag; distanza 55 m

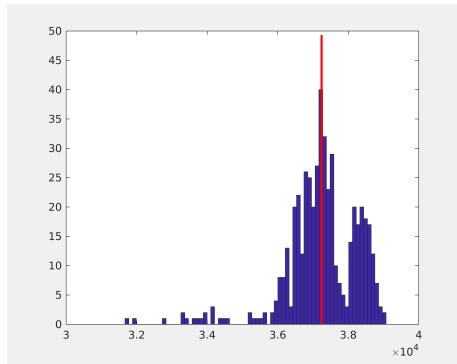


Figura 105: Istogramma coordinata x del tag; distanza 60 m, orientazione 0 gradi

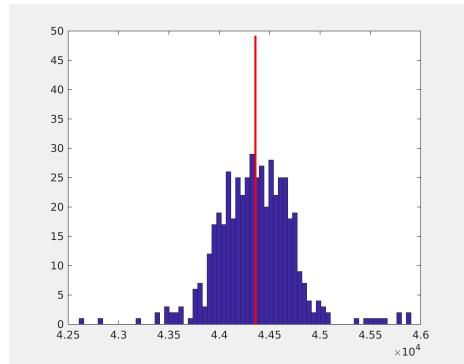


Figura 106: Istogramma coordinata y del tag; distanza 60 m, orientazione 0 gradi

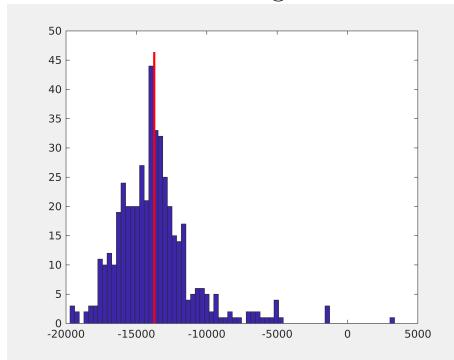


Figura 107: Istogramma coordinata z del tag; distanza 60 m, orientazione 0 gradi

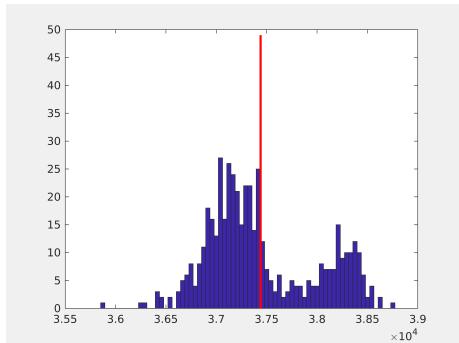


Figura 108: Istogramma coordinata x del tag: distanza 60 m, orientazione 90 gradi

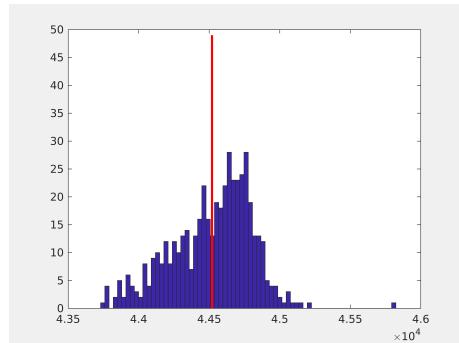


Figura 109: Istogramma coordinata y del tag: distanza 60 m, orientazione 90 gradi

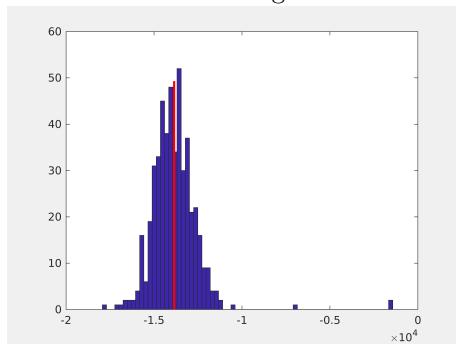


Figura 110: Istogramma coordinata z del tag: distanza 60 m, orientazione 90 gradi

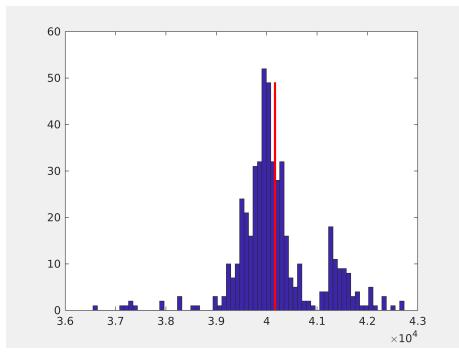


Figura 111: Istogramma coordinata x del tag: distanza 65 m

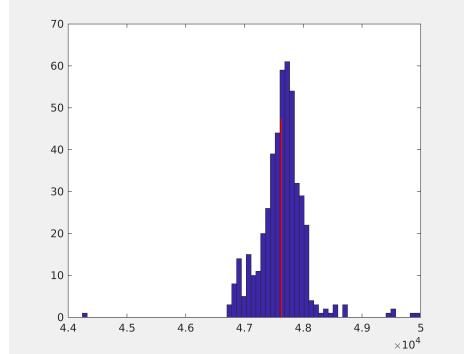


Figura 112: Istogramma coordinata y del tag: distanza 65 m

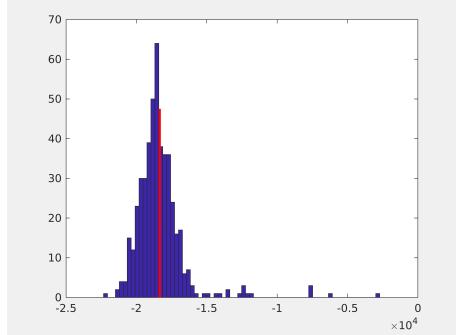


Figura 113: Istogramma coordinata z del tag: distanza 65 m

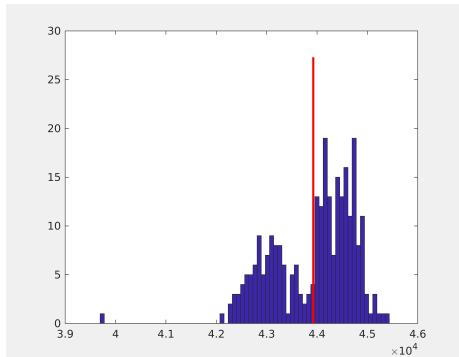


Figura 114: Istogramma coordinata x del tag: distanza 70 m

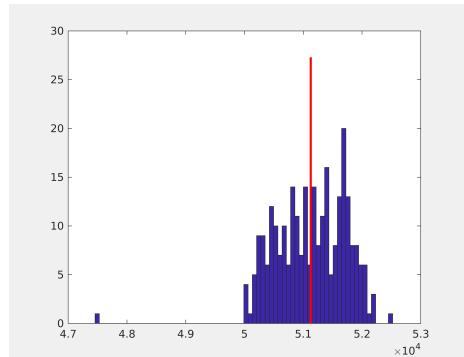


Figura 115: Istogramma coordinata y del tag: distanza 70 m

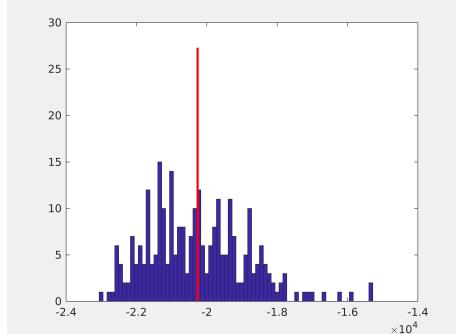


Figura 116: Istogramma coordinata z del tag: distanza 70 m

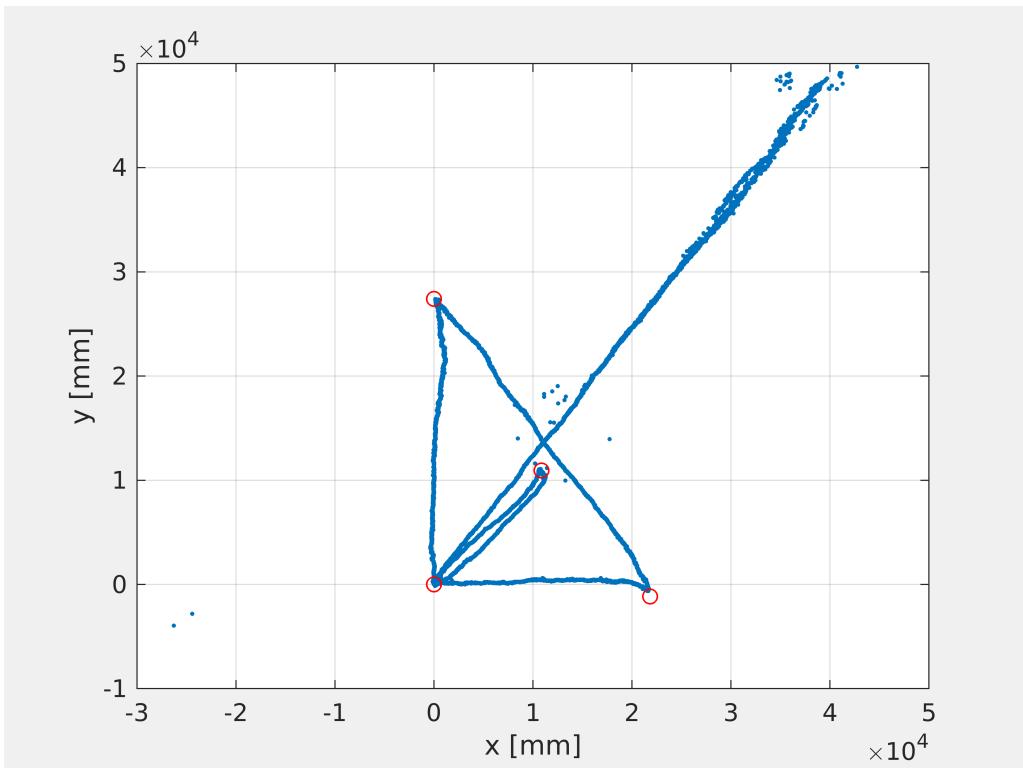


Figura 117: Traiettoria seguita in blu, ancore in rosso. Vista in pianta

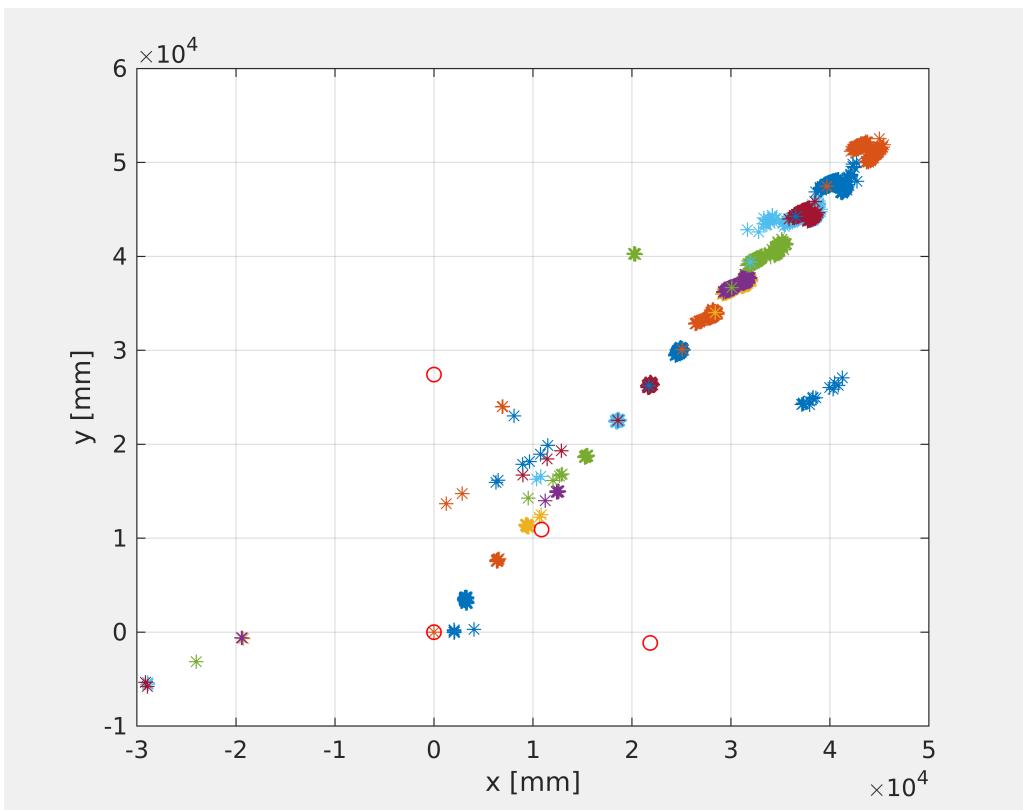


Figura 118: Positioning a varie distanze

## A Script di base

Nella cartella "Script di base" abbiamo inserito alcuni script utili per configurare la rete Pozyx. Per maggiori dettagli sul funzionamento degli script si rimanda all'opportuno READ\_ME.txt.

### A.1 autocalibration.py

Questo script effettua un'autocalibrazione delle ancore. Per farlo funzionare occorre per prima cosa avere quattro ancore correttamente alimentate ed un dispositivo Pozyx connesso alla porta USB del PC, il quale servirà da comunicazione seriale tra la rete Pozyx e l'utente. Il dispositivo seriale può, a discrezione dell'utente, essere un'ancora o un tag. È possibile, se si desidera, effettuare una calibrazione manuale delle ancore. In tal caso si dovrà utilizzare un metro laser per misurare la distanza relativa tra le coppie di ancore della rete ed inserire manualmente i valori misurati nelle opportune variabili r01, r02, r03, r12, r13 ed r23, che rappresentano le distanze tra le rispettive antenne.

Per quanto riguarda invece la calibrazione automatica, lo script prevede una fase di acquisizione dei dati necessari, successivamente viene eseguito l'algoritmo ransac ed infine viene utilizzato l'algoritmo algebrico per determinare le coordinate effettive delle ancore. L'algoritmo ransac ripulisce le misurazioni delle distanze relative tra le antenne da eventuali outliers e fornisce quindi una stima della distanza tra ciascuna coppia di ancore basata sui dati senza outliers.

Nel corso della procedura di autocalibrazione vengono stampati sul terminale i seguenti dati:

- Coordinate dei dispositivi all'accensione del sistema, prima che sia effettuata la nuova calibrazione;
- Risultato del settaggio dei parametri UWB della rete;
- Il numero di misurazioni (procedura di autoranging) effettuate con successo per ciascuna coppia di ancore. Tali misurazioni vengono immagazzinate in un array. Se si indica con  $v_k$  il k-esimo elemento dell'array e con  $r_{ij}^k$  la k-esima misurazione della distanza tra le ancore i e j, effettuata dall'ancora i, si ha che:

$$v_k = \frac{r_{ij}^k + r_{ji}^k}{2}$$

- Il valor medio delle distanze tra le ancore;
- Il risultato dell'algoritmo ransac per la distanza tra le ancore;
- Un'immagine che sintetizza i risultati dell'algoritmo ransac dove vengono visualizzati i dati grezzi delle distanze, il valor medio di tali dati, il valore della distanza ottenuto con l'algoritmo ransac e gli inliers determinati da tale algoritmo. Tale immagine viene salvata nella medesima cartella con il nome `autocalibration_results.png`. Qualora l'algoritmo non andasse a buon fine, è comunque possibile visualizzare i dati raw ottenuti nell'immagine `faw_distances.png`.

In fondo allo script (dopo `if __name__ == "__main__":`) è possibile andare a settare i vari parametri della rete:

- `auto_cal` = mettere a True se si vuole fare autocalibrazione, False se si vuole effettuare una calibrazione manuale.
- `serial_id` = id del dispositivo connesso alla porta seriale. Esso è stampato sulla schedina elettronica del dispositivo stesso ed è un numero esadecimale (es. 0x6e7a)
- `anchor_ids` = id delle ancore che vogliono essere utilizzate. Da inserire in una lista ordinata.
- Vari Parametri dell'Ultra Wide Band, meglio indicati nello script stesso.
- I parametri dell'algoritmo RANSAC:
  - `n`: rappresenta la percentuale di dati che viene scelta randomicamente come passo base dell'algoritmo RANSAC ad ogni iterazione (scegliere ad esempio il 10% del dataset iniziale)

- k: rappresenta il numero di iterazioni dell'algoritmo. Si è infatti scelto di implementare un RANSAC con numero finito di iterazioni, impostato a priori (si consiglia un valore fra  $10^4$  e  $10^5$ ).
- t: è un valore di soglia che permette di discriminare se un dato al di fuori dell'insieme iniziale scelto è un inlier o un outlier. È bene scegliere t dello stesso ordine di grandezza della varianza che ci si aspetta dall'insieme di dati. In questo modo si selezioneranno come dati buoni solo quelli vicini al valor medio.
- d: rappresenta la percentuale minima di inlier, sul numero di dati totali, che deve essere determinata dall'algoritmo ransac. Ci si può aspettare che se il numero di inliers è sufficientemente alto, l'algoritmo RANSAC abbia prodotto un risultato soddisfacente (fissarlo ad esempio al 30% del dataset iniziale ).

Una volta terminata la calibrazione, sia questa stata manuale o automatica, ciascuna ancora avrà salvato nella propria lista dei dispositivi le proprie coordinate, ossia conoscerà le proprie coordinate. Questi dati sono salvati permanentemente nella memoria flash.

In una successiva fase di positioning è quindi buona norma che il dispositivo che si deve localizzare interroghi le ancore per conoscere la loro posizione e crei quindi una propria lista interna dei dispositivi della rete.

## A.2 Descrizione procedura di autocalibrazione

La procedura di autocalibrazione è così articolata:

1. viene fornita sul terminale il risultato della comunicazione UWB dei dispositivi, se i dispositivi hanno tutti gli stessi parametri, si può procedere con la calibrazione, altrimenti questa dovrà essere interrotta.
2. se i parametri UWB dei dispositivi sono differenti e non si conoscesse i parametri UWB dei dispositivi stessi, si consiglia di riavviare i dispositivi. Se i parametri UWB non tornano a quelli di default, significa che i precedenti utilizzatori hanno permanentemente modificato i parametri dei dispositivi. Contattare le persone in questione oppure reinstallare il firmware dei dispositivi oppure tentare di modificare i parametri finché il tag non comunica con questi dispositivi (si tratta comunque di 96 prove, quindi è consigliabile reinstallare il firmware se possibile).
3. se i parametri uwb sono stati settati e la variabile saveUWBSettings è impostata a True, i parametri UWB vengono impostati permanentemente a quelli impostati. Se saveUWBSettings è impostata a False, non saranno salvati permanentemente i parametri UWB.
4. se la variabile autoCal è settata a False, inserire manualmente all'interno dello script la distanza tra le ancore, misurata ad esempio con un metro laser, altrimenti ha inizio la procedura di autocalibrazione.
5. vengono effettuate un numero di acquisizioni preimpostate nello script delle distanze tra le possibili coppie di antenne.
6. i dati relativi a questa acquisizione vengono inseriti in un grafico nell'immagine raw\_distances.png
7. ha inizio una procedura di algoritmo ransac per la rimozione degli outliers, se questa ha successo verrà mostrata un'immagine autocalibration\_results.png che riassumerà i risultati dell'algoritmo.
8. ) viene chiesto se si desidera ripetere l'algoritmo ransac e se si vogliono modificare i parametri di tale algoritmo.
9. le distanze calcolate con l'algoritmo ransac vengono fornite in input all'algoritmo algebrico per la determinazione delle coordinate delle ancore.
10. i risultati dell'autocalibrazione delle ancore vengono mostrati sulla shell e si chiede se si desidera salvare le coordinate delle ancore nella loro lista interna dei dispositivi. In questo modo è possibile realizzare futuri script di positioning che non richiedano di inserire manualmente

le coordinate delle ancore. Basterà far sì che in una prima fase di setup il tag interroghi le ancore chiedendo loro le coordinate e salvare i risultati di questa procedura nella lista interna dei dispositivi del tag. E' comunque necessario inserire manualmente gli id delle ancore nell'ordine della convenzione.

### A.3 Autocalibrazione senza RANSAC

```

1  # Questo script python permette di effettuare un'acquisizione automatica delle
2  # posizioni dei tag,
3  # utilizzando l'algoritmo di posizionamento della pozyx.
4  # La posizione dei tag determinata dall'algoritmo viene comunicata sulla linea UWB
5  # al dispositivo
6  # ché alla porta USB del PC. Sullo schermo vengono stampate le
7  # posizioni dei due tag.
8  # E' possibile calibrare le ancore sia manualmente che automaticamente. I parametri
9  # da modificare
10 # si trovano nella funzione "main" in fondo al programma. Il programma pensato
11 # per una
12 # configurazione minima che prevede l'utilizzo di 4 ancore per la calibrazione.
13
14 # Su linux:
15 # da terminale digitare >> sudo python3 test_UWB.py
16
17 import time
18 from rangesToPos import (rangesToPos, average)
19 from pypyzyx import *
20
21 MAX_DISTANCE = 100000
22
23 class CalibrationPozyx:
24     def __init__(self, pozyx, anchor_ids, serial_id, auto_cal, R_mis,
25                  ranging_protocol,
26                  N_acq, myUWBSettings=None, dimension = POZYX_3D, height=1000, algorithm=
27                  POZYX_POS_ALG_UWB_ONLY,
28                  update_interval = 100):
29         self.pozyx = pozyx
30         self.anchor_ids = anchor_ids
31         self.auto_cal = auto_cal
32         self.R_mis = R_mis
33         self.ranging_protocol = ranging_protocol
34         self.dimension = dimension
35         self.height = height
36         self.algorithm = algorithm
37         self.N_acq = N_acq
38         self.serial_id = serial_id
39
40         self.anchors = []
41         self.distances = []
42
43         self.UWBSettings = myUWBSettings
44         self.UWBChannel = myUWBSettings.channel
45         self.defaultUWBSettings = UWBSettings(5,0,2,8,11.5)
46
47     def setup(self):
48         # Set UWB Parameters
49         for remote in anchor_ids:
50             if remote != self.serial_id:
51                 self.pozyx.setRangingProtocol(self.ranging_protocol, remote)
52                 self.pozyx.setUWBSettings(self.UWBSettings, remote)
53                 self.pozyx.setUWBChannel(self.UWBChannel, remote)
54
55         self.pozyx.setRangingProtocol(self.ranging_protocol)
56         self.pozyx.setUWBSettings(self.UWBSettings)
57         self.pozyx.setUWBChannel(self.UWBChannel)
58
59         print("Coordinate iniziali dei dispositivi della rete")
60         coordinates = Coordinates()

```

```

58     for anchor in self.anchor_ids:
59         if anchor == self.serial_id:
60             status = self.pozyx.getDeviceCoordinates(anchor, coordinates, None)
61         else :
62             status = self.pozyx.getDeviceCoordinates(anchor, coordinates,
63                                         anchor)
64
65         if status == POZYX_FAILURE:
66             print ("Device " + str(hex(anchor)) + "list is empty")
67             print(hex(anchor), coordinates.x, coordinates.y, coordinates.z)
68
69     ### Print UWB configuration results
70     for remote in anchor_ids:
71         self.printUWBSettings(remote)
72         self.printUWBSettings(None)
73
74     # Perform autocalibration if needed
75     if not self.auto_cal:
76         self.anchors = rangesToPos(self.R_mis, self.anchor_ids)
77     else:
78         print("Performing autocalibration")
79         self.getDistances()
80         self.anchors = rangesToPos(self.R_mis, self.anchor_ids)
81
82     self.printConfigurationResults()
83     self.addAnchors()                                #Salve le ancore nella lista interna
84                                     #dei dispositivi
85                                     #dei due tag
86     print("Ancore aggiunte alla lista interna dei dispositivi")
87
88 def getDistances(self):
89     device_range1 = DeviceRange()
90     device_range2 = DeviceRange()
91     R = [[],[],[],[],[],[]]
92
93     coppie = [ [anchor_ids[0], anchor_ids[1]],
94               [anchor_ids[0], anchor_ids[2]],
95               [anchor_ids[0], anchor_ids[3]],
96               [anchor_ids[1], anchor_ids[2]],
97               [anchor_ids[1], anchor_ids[3]],
98               [anchor_ids[2], anchor_ids[3]] ]
99
100    for k in list(range(self.N_acq)):
101        i = 0
102        for coppia in coppie:
103            if coppia[0] == self.serial_id:
104                status1 = POZYX_SUCCESS
105                status2 = self.pozyx.doRanging(coppia[1], device_range2, None)
106                device_range1 = device_range2
107            elif coppia[1] == self.serial_id:
108                status1 = self.pozyx.doRanging(coppia[0], device_range1, None)
109                status2 = POZYX_SUCCESS
110                device_range2 = device_range1
111            else:
112                status1 = self.pozyx.doRanging(coppia[0], device_range1, coppia
113                                            [1])
114                status2 = self.pozyx.doRanging(coppia[1], device_range2, coppia
115                                            [0])
116
117                if ((status1 == POZYX_SUCCESS) and (device_range1.distance <=
118                                              MAX_DISTANCE)):
119                    if ((status2 == POZYX_SUCCESS) and (device_range2.distance <=
120                                              MAX_DISTANCE)):
121                        dist = (device_range1.distance + device_range2.distance)/2
122                        R[i].append(dist)
123                        i = i+1
124
125                for i in list(range(len(R))):
126                    R[i] = average(R[i])
127                self.R_mis = R

```

```

125
126
127     def printConfigurationResults(self):
128         print("Configuration finished: ")
129         i=0
130         for anchor in self.anchors:
131             print("ANCHOR", i, ", 0x%0.4x, %s" % (anchor.network_id, str(anchor.pos
132                                         ))))
133             i += 1
134         print()
135         print()
136
137
138     def printPosition(self, position, remote_id=None):
139         """Stampa a video il risultato dell'algoritmo di calibrazione del tag"""
140         new_time = int(round(time.time() * 1000))
141         if self.use_remote_tag:
142             if remote_id is not None:
143                 print("POS ID {}, x(mm): {} y(mm): {} z(mm): {}".format(
144                     "0x%0.4x" % remote_id, pos=position), " Delta_t [ms]: ", new_time
145                     - self.old_time[remote_id])
146             else:
147                 print("Serial pozyx: x(mm): {} y(mm): {} z(mm): {}".format(
148                     "0x%0.4x", pos=position), " Delta_t [ms]: ", new_time - self.
149                     old_time[remote_id])
150         self.old_time[remote_id] = new_time
151
152
153     def printUWBSettings(self, remote_id):
154         aux = UWBSettings()
155         status = self.pozyx.getUWBSettings(aux, remote_id)
156         if status == POZYX_SUCCESS:
157             if remote_id == None:
158                 print("Serial ", aux.channel, aux.bitrate, aux.prf, hex(aux.plen),
159                     aux.gain_db)
160             else:
161                 print(hex(remote_id), "Channel: ", aux.channel, "Bitrate: ", aux.
162                     bitrate,
163                     "Prf: ", aux.prf, "Plen: ", hex(aux.plen), "Gain: ", aux.
164                     gain_db)
165         else:
166             if remote_id == self.serial_id:
167                 return
168             print(hex(remote_id), "Failed to receive UWB Settings")
169
170
171     def addAnchors(self):
172         for anchor in self.anchors:
173             aux_id = anchor.network_id
174             if anchor.network_id == self.serial_id:
175                 aux_id = None
176             status = self.pozyx.clearDevices(aux_id)
177             status &= self.pozyx.addDevice(anchor, aux_id)
178             status &= self.pozyx.saveNetwork(aux_id)
179             if status == POZYX_FAILURE or status == POZYX_TIMEOUT:
180                 raise Exception("Device " + str(hex(aux_id)) + ": Failed to add
181                                 anchors to tag's device list")
182             quit()
183
184
185     if __name__ == "__main__":
186         # Riconosce se ci sono dispositivi Pozyx connessi alla porta USB del PC
187         serial_port = get_first_pozyx_serial_port()
188         print(serial_port)
189         if serial_port is None:
190             raise Exception("No Pozyx connected. Check your USB cable or your driver")
191             quit()

```

```

189 pozzyx = PozyxSerial(serial_port)
190
191
192 ##### - PARAMETRI - #####
193 ##### - PARAMETRI - #####
194 ##### - PARAMETRI - #####
195 ##### - PARAMETRI - #####
196 # Calibrazione automatica-manuale
197 auto_cal = False # False = manuale, True = automatica
198 serial_id = 0x675e
199 anchor_ids = [0x6e44, 0x6e7a, 0x6e6c, 0x6939] # !!! Seguire la convenzione di
200 # rangesToPos.py
201 r01 = 5448 # Inserire i valori se si vuole fare una calibrazione manuale
202 r02 = 5802
203 r03 = 3600
204 r12 = 4210
205 r13 = 3160
206 r23 = 2682
207 N_acq = 20
208 R_mis = [r01, r02, r03, r12, r13, r23]
209
210 # Ranging protocol: POZYX_RANGE_PROTOCOL_FAST or POZYX_RANGE_PROTOCOL_PRECISION
211 ranging_protocol = POZYX_RANGE_PROTOCOL_PRECISION
212
213 # Impostazioni Ultra Wide Band
214 # Canale: 1,2,3,5 o 7
215 channel = 5 # Default value = 5
216
217 # Bitrate, possibili valori:
218 # PozyxConstants.UWB_BITRATE_110_KBPS Default value
219 # PozyxConstants.UWB_BITRATE_850_KBPS
220 # PozyxConstants.UWB_BITRATE_6810_KBPS
221 bitrate = PozyxConstants.UWB_BITRATE_6810_KBPS
222
223 # Pulse repeat frequency, possibili valori:
224 # PozyxConstants.UWB_PRF_64_MHZ Default value
225 # PozyxConstants.UWB_PRF_16_MHZ
226 prf = PozyxConstants.UWB_PRF_64_MHZ
227
228 # Preamble length of the UWB packets, possibili valori:
229 # PozyxConstants.UWB_PLEN_64
230 # PozyxConstants.UWB_PLEN_128
231 # PozyxConstants.UWB_PLEN_256
232 # PozyxConstants.UWB_PLEN_512
233 # PozyxConstants.UWB_PLEN_1024 Default value
234 # PozyxConstants.UWB_PLEN_1536
235 # PozyxConstants.UWB_PLEN_2048
236 # PozyxConstants.UWB_PLEN_4096
237 plen = PozyxConstants.UWB_PLEN_64
238
239 # UWB Gain, possibili valori:
240 # float tra 0 e 67.1 dB
241 gain = 11.5 # Default value 11.5 dB
242
243 myUWBSettings = UWBSignals(channel, bitrate, prf, plen, gain)
244 ##### - Fine settaggio parametri - #####
245
246
247
248 # Inizializzo pozzyx seriale
249 serial_tag = CalibrationPozyx(pozzyx, anchor_ids, serial_id, auto_cal, R_mis,
250 ranging_protocol, N_acq, myUWBSettings)
251
252 # Effettuo il setup della rete
253 serial_tag.setup()
254
255 # Genero il file di testo Parametri_UWB.txt
256 out_file = open("Parametri_UWB.txt", "w")
257 out_file.write(str(channel)+'\n')
258 out_file.write(str(bitrate)+'\n')

```

```

259     out_file.write(str(prf)+'\n')
260     out_file.write(str(plen)+'\n')
261     out_file.write(str(gain)+'\n')
262     out_file.close()

```

## A.4 Algoritmo RANSAC

```

1 import numpy
2 import scipy # use numpy if scipy unavailable
3 import scipy.linalg # use numpy if scipy unavailable
4
5 ## Copyright (c) 2004-2007, Andrew D. Straw. All rights reserved.
6
7 ## Redistribution and use in source and binary forms, with or without
8 ## modification, are permitted provided that the following conditions are
9 ## met:
10
11 ## * Redistributions of source code must retain the above copyright
12 ##   notice, this list of conditions and the following disclaimer.
13
14 ## * Redistributions in binary form must reproduce the above
15 ##   copyright notice, this list of conditions and the following
16 ##   disclaimer in the documentation and/or other materials provided
17 ##   with the distribution.
18
19 ## * Neither the name of the Andrew D. Straw nor the names of its
20 ##   contributors may be used to endorse or promote products derived
21 ##   from this software without specific prior written permission.
22
23 ## THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
24 ## "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
25 ## LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
26 ## A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
27 ## OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
28 ## SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
29 ## LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
30 ## DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
31 ## THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
32 ## (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
33 ## OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
34
35 def ransac(data,model,n,k,t,d,debug=False,return_all=False):
36     """fit model parameters to data using the RANSAC algorithm
37
38     This implementation written from pseudocode found at
39     http://en.wikipedia.org/w/index.php?title=RANSAC&oldid=116358182
40
41     """
42     Given:
43         data - a set of observed data points
44         model - a model that can be fitted to data points. It is class with at
45             least two methods:
46             1) self.fit(self, data) method that determines the parameters of
47                 the model given a
48                 data set.
49             2) self.get_error(self, data, model) that determines an error
50                 associated to each
51                 data point with the current parameters of the model, indicated
52                 in 'model'.
53
54     n - the minimum number of data values required to fit the model
55     k - the maximum number of iterations allowed in the algorithm
56     t - a threshold value for determining when a data point fits a model
57     d - the number of close data values required to assert that a model fits
58         well to data
59
60     Return:
61         bestfit - model parameters which best fit the data (or nil if no good model
62             is found)
63         iterations = 0
64         bestfit = nil
65         besterr = something really large

```

```

59     while iterations < k {
60         maybeinliers = n randomly selected values from data
61         maybeModel = model parameters fitted to maybeinliers
62         alsoInliers = empty set
63         for every point in data not in maybeinliers {
64             if point fits maybeModel with an error smaller than t
65                 add point to alsoInliers
66         }
67         if the number of elements in alsoInliers is > d {
68             % this implies that we may have found a good model
69             % now test how good it is
70             betterModel = model parameters fitted to all points in maybeinliers and
71                 alsoInliers
72             thiserr = a measure of how well model fits these points
73             if thiserr < besterr {
74                 bestfit = betterModel
75                 besterr = thiserr
76             }
77             increment iterations
78         }
79     return bestfit
80 }
81 """
82 errors = numpy.array([]) # Container for the errors determined at
83 # each successful
84 # iteration. An iteration is successfull
85 # when at least d
86 # inliers are found and when the error
87 # found at the current
88 # step is better than the previous one. An
89 # inlier is found
90 # when its error is below the threshold
91
92 iteration_array = numpy.array([]) # Container for the number of iterations
93 # that were
94 # successful.
95
96 iterations = 0
97 bestfit = None # Container for the best model parameters
98 found
99 besterr = numpy.inf # Error associated to the best model found
100 best_inlier_idxs = None # Indeces of the data that were included to
101 # compute the
102 # best error parameters
103
104 while iterations < k:
105     # Separate the data into two sets
106
107     # Select n random samples for algorithm initial condition and N-n samples
108     # for testing
109     maybe_idxs, test_idxs = random_partition(n,data.shape[0])
110     maybeinliers = data[maybe_idxs,:] # These are the n samples with
111     # their values
112     test_points = data[test_idxs] # these are the testing points with
113     # only their indeces
114
115     maybeModel = model.fit(maybeinliers) # Find model parameters of the
116     # random set
117     test_err = model.get_error( test_points, maybeModel) # Find error
118     # associated to each testing point
119     also_idxs = test_idxs[test_err < t] # select indices of rows with
120     # accepted points
121     alsoInliers = data[also_idxs,:] # Include the data of the above
122     # indices as well
123
124     if debug:
125         print ('test_err.min()',test_err.min())
126         print ('test_err.max()',test_err.max())
127         print ('numpy.mean(test_err)',numpy.mean(test_err))
128         print ('iteration %d:len(alsoInliers) = %d'%(
129             iterations,len(alsoInliers)))
130
131     if len(alsoInliers) > d:
132         betterdata = numpy.concatenate( (maybeinliers, alsoInliers) )

```

```

117     bettermodel = model.fit(betterdata)      # Find model parameters of the
118         new data set
119
120         # the new data set is composed by the
121         # initial random
122         # points and the testing points that
123         # passed the test
124     better_errs = model.get_error( betterdata, bettermodel)      # Find
125         error associated to
126             # each element of new model
127     thiserr = numpy.mean( better_errs ) # Find error associated to
128         new_model
129     if thiserr < besterr:           # Update values if the model is better
130         than the previous
131     bestfit = bettermodel
132     besterr = thiserr
133     errors = numpy.append(errors, thiserr)
134     iteration_array = numpy.append(iteration_array, iterations)
135     best_inlier_idxs = numpy.concatenate( (maybe_idxs, also_idxs) )
136     iterations+=1
137     if bestfit is None:
138         raise ValueError("did not meet fit acceptance criteria")
139     if return_all:
140         return bestfit, {'inliers':best_inlier_idxs}, errors, iteration_array
141     else:
142         return bestfit
143
144
145
146
147
148 # Partition a list of indeces in two separate groups randomly
149 def random_partition(n,n_data):
150     """return n random rows of data (and also the other len(data)-n rows)"""
151     all_idxs = numpy.arange( n_data )
152     numpy.random.shuffle(all_idxs)
153     idxs1 = all_idxs[:n]
154     idxs2 = all_idxs[n:]
155     return idxs1, idxs2
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182 def test():
183     # generate perfect input data

```

```

184
185     n_samples = 500
186     n_inputs = 1
187     n_outputs = 1
188     A_exact = 20*numpy.random.random((n_samples,n_inputs) )
189
190     #Linear coefficient of the line
191     perfect_fit = 60*numpy.random.normal(size=(n_inputs,n_outputs) ) # the model
192
193     #Exact values on the y-axis
194     B_exact = scipy.dot(A_exact,perfect_fit)
195
196
197     assert B_exact.shape == (n_samples,n_outputs)
198
199     # add a little gaussian noise (linear least squares alone should handle this
200     # well)
201     A_noisy = A_exact + numpy.random.normal(size=A_exact.shape )
202     B_noisy = B_exact + numpy.random.normal(size=B_exact.shape )
203     if 1:
204         # add some outliers
205         n_outliers = 100
206         all_idxs = numpy.arange( A_noisy.shape[0] )
207         numpy.random.shuffle(all_idxs)
208         outlier_idxs = all_idxs[:n_outliers]
209         non_outlier_idxs = all_idxs[n_outliers:]
210         A_noisy[outlier_idxs] = 20*numpy.random.random((n_outliers,n_inputs) )
211         B_noisy[outlier_idxs] = 50*numpy.random.normal(size=(n_outliers,n_outputs)
212             )
213
214     # setup model
215
216     all_data = numpy.hstack( (A_noisy,B_noisy) )
217     input_columns = range(n_inputs) # the first columns of the array
218     output_columns = [n_inputs+i for i in range(n_outputs)] # the last columns of
219     # the array
220     debug = False
221     model = LinearLeastSquaresModel(input_columns,output_columns,debug=debug)
222
223     linear_fit,resids,rank,s = scipy.linalg.lstsq(all_data[:,input_columns],
224                                                 all_data[:,output_columns])
225
226     # run RANSAC algorithm
227     ransac_fit, ransac_data, errors, iterations= ransac(all_data,model,
228                                                       50, 10000, 7e3, 300, # misc. parameters
229                                                       debug=debug,return_all=True)
230     print(errors)
231     print(iterations)
232     if 1:
233         import pylab
234
235         sort_idxs = numpy.argsort(A_exact[:,0])
236         A_col0_sorted = A_exact[sort_idxs] # maintain as rank-2 array
237
238         if 1:
239             pylab.plot( A_noisy[:,0], B_noisy[:,0], 'k.', label='data' )
240             pylab.plot( A_noisy[ransac_data['inliers'],0], B_noisy[ransac_data['
241                 inliers'],0], 'bx', label='RANSAC data' )
242         else:
243             pylab.plot( A_noisy[non_outlier_idxs,0], B_noisy[non_outlier_idxs,0], '
244                 k.', label='noisy data' )
245             pylab.plot( A_noisy[outlier_idxs,0], B_noisy[outlier_idxs,0], 'r.',
246                         label='outlier data' )
247             pylab.plot( A_col0_sorted[:,0],
248                         numpy.dot(A_col0_sorted,ransac_fit)[:,0],
249                         label='RANSAC fit' )
250             pylab.plot( A_col0_sorted[:,0],
251                         numpy.dot(A_col0_sorted,perfect_fit)[:,0],
252                         label='exact system' )
253             pylab.plot( A_col0_sorted[:,0],
254                         numpy.dot(A_col0_sorted,linear_fit)[:,0],
255                         label='linear fit' )
256
257         pylab.legend()

```

```

251     pylab.show()
252
253 if __name__=='__main__':
254     test()

```

## A.5 Autocalibrazione con RANSAC

```

1  # Questo script python permette di effettuare un'acquisizione automatica delle
2  # posizioni dei tag,
3  # utilizzando l'algoritmo di posizionamento della pozyx.
4  # La posizione dei tag determinata dall'algoritmo viene comunicata sulla linea UWB
5  # al dispositivo
6  # che si connette alla porta USB del PC. Sullo schermo vengono stampate le
7  # posizioni dei due tag.
8  # E' possibile calibrare le ancore sia manualmente che automaticamente. I parametri
9  # da modificare
10 # si trovano nella funzione "main" in fondo al programma. Il programma pensato
11 # per una
12 # configurazione minima che prevede l'utilizzo di 4 ancore per la calibrazione.
13
14
15
16
17
18
19
20 #####
21 ##### CalibrationPozyx class #####
22 #####
23 class CalibrationPozyx:
24     def __init__(self, pozyx, anchor_ids, serial_id, auto_cal, R_mis,
25                  ranging_protocol,
26                  N_acq, n_percentage, k, t, d_percentage, plot_ransac_error = False,
27                  myUWBSettings=None):
28         # Argument variables
29         self.pozyx = pozyx                         # PozyxSerial class' instance
30         self.anchor_ids = anchor_ids                # list of anchor ids [A0, A1, A2, A3]
31         self.auto_cal = auto_cal                   # True if you want to perform
32             autocalibration
33         self.R_mis = R_mis                         # Container for measured distances. It's
34             a list of lists
35
36         self.ranging_protocol = ranging_protocol      # Pozyx ranging protocol used
37         self.N_acq = N_acq                          # number of acquisitions t be
38             performed
39         self.serial_id = serial_id                  # Id of serial pozyx device
40         self.UWBSettings = myUWBSettings
41
42         # Ransac parameters: explained in main function
43         self.n_percentage = n_percentage
44         self.k = k
45         self.t = t
46         self.d_percentage = d_percentage
47         self.plot_ransac_error = plot_ransac_error
48
49
50         # Internal class variables
51         self.R = self.init_empty_list(6)           # Each element of this list is a vector
52             containing all
53
54             # measurements of the distance between
55             # all possible
56             # anchor pairs. Each measurement is the
57             # average between
58             # the measurement from anchor i to j and
59             # from anchor j

```

```

51                                     # to i.
52 self.R_avg = []                      # Container for average values of the
53                                     anchor distances
54 self.anchors = []                     # Container for anchor ids and
55                                     coordinates. It stores
56                                     # the data obtained with rangesToPos
57                                     algorithm.
58 self.coppie = [[anchor_ids[0], anchor_ids[1]],           # List of all possible
59                                     anchor pairs
60                                     [anchor_ids[0], anchor_ids[2]],
61                                     [anchor_ids[0], anchor_ids[3]],
62                                     [anchor_ids[1], anchor_ids[2]],
63                                     [anchor_ids[1], anchor_ids[3]],
64                                     [anchor_ids[2], anchor_ids[3]]]
65
66
67
68 #####
69 #####           SETUP METHOD           #####
70 ##########
71 #####
72 def setup(self):
73     # Set the UWB parameters for all the anchors, the remote tags and the
74     # serial tag
75     for remote in anchor_ids:
76         if remote != self.serial_id:
77             self.pozyx.setRangingProtocol(self.ranging_protocol, remote)
78             self.pozyx.setUWBSettings(self.UWBSettings, remote)
79             self.pozyx.setUWBChannel(self.UWBChannel, remote)
80
81             self.pozyx.setRangingProtocol(self.ranging_protocol)
82             self.pozyx.setUWBSettings(self.UWBSettings)
83             self.pozyx.setUWBChannel(self.UWBChannel)
84
85             #Print Device coordinates when system is turned on
86             print("Device coordinates at starting:")
87             coordinates = Coordinates()
88             i=0
89             for anchor in self.anchor_ids:
90                 if anchor == self.serial_id:
91                     status = self.pozyx.getDeviceCoordinates(anchor, coordinates, None)
92                 else:
93                     status = self.pozyx.getDeviceCoordinates(anchor, coordinates,
94                                                 anchor)
95
96                 if status == POZYX_FAILURE:
97                     print ("Couldn't connect to device ", str(hex(anchor)))
98                     print("ANCHOR ", i, str(hex(anchor)), ", X:", coordinates.x, ", Y:",
99                           coordinates.y,
100                           ", Z:", coordinates.z)
101                i += 1
102            print()
103
104            ### Print UWB configuration results
105            print("Current UWB configuration:")
106            for remote in anchor_ids:
107                self.printUWBSettings(remote)
108            self.printUWBSettings(None)
109            print()
110
111            # Perform autocalibration if needed, then plot results
112            if not self.auto_cal:
113                # Compute algebraic algorithm to determine the anchors' coordinates
114                self.anchors = rangesToPos(self.R_mis, self.anchor_ids)
115            else:
116                print("Performing autocalibration")

```

```

114     self.getDistances()      # get the distances between the anchors
115         automatically
116
117     ransac_again = 'y'
118     while ransac_again == 'y':
119         self.R_mis = []
120         self.ransac()
121         try:
122             print("\nAverage distances between the anchors, NO ransac: \n ["
123                 r01,r02,r03,r12,r13,r23] =", self.R_avg)
124             self.anchors = rangesToPos(self.R_mis, self.anchor_ids)
125             self.printConfigurationResults()      #Print anchor coordinates
126                 in a readable way
127         except:
128             print("Configuration failed\n")
129             plt.show()
130
131     #User interface for running RANSAC algorithm
132         #####
133     ransac_again = input("Perform algorithm to obtain distances again?
134         y/n\t")
135     if ransac_again == 'y':
136         change_ransac_param = input("Do you want to change ransac
137             parameters? y/n\t")
138         if change_ransac_param == 'y':
139             print("Current values n:", self.n_percentage, " k",
140                 self.
141                     k, " t", self.t, " d", self.d_percentage)
142             self.n_percentage = float(input("Percentage of data
143                 initially chosen n:\t"))
144             self.k = int(input("Number of iterations k:\t"))
145             self.t = float(input("Tolerance t:\t"))
146             self.d_percentage = float(input("Percentage of desired
147                 inliers d:\t"))
148
149             plt.close('all')
150
151
152     # User interface to save results #####
153     save_results = input("Save current results? y/n\t")
154     if save_results == 'y':
155
156         attempts = 'y'
157         success = False
158         while attempts =='y':
159             try:
160                 print("Saving...")
161                 self.addAnchors()          #Add anchors' to tag internal device
162                     list
163                     success = True
164                     break
165             except:
166                 attempts = input("Failed to add anchors to anchors's device
167                     list, try saving again? y/n\t")
168             if success == False:
169                 raise Exception("Quitting...")
170                 quit()
171             print("Success: each anchor has been added to its internal device list
172                     and saved in its flash memory")
173
174         else:
175             print("Quitting...")
176             quit()
177
178     ##### End of SETUP METHOD
179         #####
180
181
182     ##### getDistances METHOD #####
183
184     # This function starts a ranging procedure between the anchors of the system in
185         order to get the

```

```

173     # distances between all possible couples of anchors. The distances are obtained
174     # with the
175     # doRanging function of the pozyx library. The data are stored in the self.R
176     # variable.
177
178     def getDistances(self):
179         device_range1 = DeviceRange()      # Container for distance between anchor i
180         and anchor j
181         device_range2 = DeviceRange()      # Container for distance between anchor j
182         and anchor i
183
184         # Possible anchor's pairs
185
186         for k in list(range(self.N_acq)):
187             i = 0
188
189             if k%20==0:
190                 print("Acquisition ", k+1, "of ", N_acq)
191             for coppia in self.coppie:
192                 # call doRanging with None argument when the considered anchor's id
193                 # is serial_id
194                 if coppia[0] == self.serial_id:
195                     status1 = POZYX_SUCCESS
196                     status2 = self.pozyx.doRanging(coppia[1], device_range2, None)
197                     device_range1 = device_range2
198                 elif coppia[1] == self.serial_id:
199                     status1 = self.pozyx.doRanging(coppia[0], device_range1, None)
200                     status2 = POZYX_SUCCESS
201                     device_range2 = device_range1
202                 else:
203                     status1 = self.pozyx.doRanging(coppia[0], device_range1, coppia
204                         [1])
205                     status2 = self.pozyx.doRanging(coppia[1], device_range2, coppia
206                         [0])
207
208             # Take average result
209             MAX_DISTANCE = 1e6
210             if ((status1 == POZYX_SUCCESS) and (device_range1.distance <=
211                 MAX_DISTANCE)):
212                 if ((status2 == POZYX_SUCCESS) and (device_range2.distance <=
213                     MAX_DISTANCE)):
214                     dist = (device_range1.distance + device_range2.distance)/2
215                     self.R[i].append(dist)
216             i = i+1
217
218
219     # Plot raw data and save it in raw_distances.png
220     data_fig, axarr = plt.subplots(2,3,figsize=(16,9))
221     plt.suptitle('Measured distances between the anchors, raw data only')
222     for i in list(range(len(self.R))):
223         n_samples = len(self.R[i])
224         if n_samples < (self.N_acq)*0.20:
225             print("Could not measure the distances between the anchors properly
226                  ")
227             print("only ",n_samples, "of ", self.N_acq, "acquisitions were
228                  completed")
229             print("check if the anchors can communicate properly")
230             quit()
231         row = int(i/3)
232         col = int(i%3)
233         curr_ax = axarr[row,col]
234         curr_ax.plot(np.arange(n_samples), np.array(self.R[i]), 'k.', label='
235             data')
236         curr_title = self.create_subplot_title(row,col,self.coppie[i])
237         curr_ax.set_title(curr_title, fontsize=10)
238         curr_ax.set_ylabel('distance [mm]')
239         data_fig.savefig('raw_distances.png')
240         plt.close(data_fig)
241     ##### End of getDistances METHOD
242     #####
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332

```

```

233
234 ##### RANSAC METHOD #####
235 ##### This method performs a fitting of the data acquired by getDistances with the
236 ##### ransac algorithm.
237 # It is possible to plot the errors at the end of each ransac algorithm call.
238 # The distances obtained and a figure with the ransac results is shown. It is
239 # possible to
240 # perform multiple ransac algorithms and change the ransac parameters
241 # and than accept or reject the results.
242
243 def ransac(self):
244     # Conatainer for arrays of distances between anchor pairs. These lists
245     # should be as long as
246     # the number of anchor pairs
247     R_fitted = self.init_empty_list(6) # Container for fitted anchor's
248     # distances
249
250     R_avg = self.init_empty_list(6)      # Container for average results (no
251     # ransac algorithm).                      # Each element of this list is the
252                                         # average of the
253                                         # the elements in the vectors of list R
254                                         .
255
256     r = self.init_empty_list(6)          # Auxialiry variable, R is a list of
257     # lists, r is a list                  # of numpy.array. Conversion is needed
258                                         # because of
259                                         # ransac.py
260
261     indexes = self.init_empty_list(6)    # list of lists. Each list contains the
262     # indexes that are                  # the output of the RANSAC algorithm.
263                                         # This indexes
264                                         # represent the points chosen for the
265                                         # fit.
266
267     # Compute R_avg
268     for i in list(range(6)):
269         R_avg[i] = average(self.R[i])
270     self.R_avg = R_avg
271
272     # Perform ransac algorithm
273     print()
274     ransac_success = True
275     for i in list(range(len(self.R))):
276         r[i] = np.array(self.R[i])[:,np.newaxis]           # Converts R in a
277         # format suitable for ransac.py
278         N = np.shape(r[i])[0]                            # get the length of the i-th
279         # vector of distances
280         print("Successful acquisitions between anchors ",str(hex(self.coppie[i][0])),
281               str(hex(self.coppie[i][1])),": ", N)
282         regressor = np.ones([N,1])                      # Creates the matrix used for
283         # linear regression
284
285         if N < 100:
286             print("Warning: only", N, "distances between anchors", str(hex(self.
287                 .coppie[i][0])),
288                 str(hex(self.coppie[i][1])), "were acquired, this number is
289                 considered small for autocalibration")
290
291         # Performs a ransac algorithm with a model based on linear regression
292         # in the form of
293         # A*x=b. A is the regressor, its number of rows is equal to the number
294         # of measurements.
295         # Its number of columns is equal to the number of parameters that will
296         # be estimated.
297         # b is also containing measurements. In this case we set A = numpy.ones
298         # ([N,1]) as our

```

```

283     # parameter is a simple constant. b contains the measurments of the
284     # distances (R[i]).  

285     n_inputs = 1           # Number of columns of the regressor (usually small
286     # number)  

287     n_outputs = 1          # Number of columns of the output  

288     all_data = np.hstack( (regressor, r[i]) )  

289     input_columns = range(n_inputs)           # indexes of the first
290     # columns of the array  

291     output_columns = [n_inputs+i for i in range(n_outputs)] # the last
292     # columns of the array  

293     debug = False  

294     model = LinearLeastSquaresModel(input_columns, output_columns, debug=
295     debug)  

296  

297     n = int(self.n_percentage*N)
298     d = int(self.d_percentage*N)  

299  

300     try:  

301         R_fitted[i], indexes[i], errors_arr, iteration_arr = ransac(
302             all_data, model, n, self.k, self.t, d,
303             debug=debug, return_all=True)
304         print("Errors in successful iteration of RANSAC: ", errors_arr)
305         print("The above error were found at iteration: ", iteration_arr)
306         print()
307         print()
308         if self.plot_ransac_error:
309             figure, ax = plt.subplots()
310             ax.plot(iteration_arr, errors_arr, 'kx-', label='data')
311             row = int(i/3)
312             col = int(i%3)
313             title = self.create_subplot_title(row, col, self.coppie[i])
314             ax.set_xlabel('iteration', color='red')
315             ax.set_ylabel('variance [mm^2]', color='red')
316             ax.set_title(title)
317             plt.show()
318             plt.close(figure)
319     except:
320         ransac_success = False  

321  

322     #Plot results
323     if ransac_success:
324         fig, axarr = plt.subplots(2,3,figsize=(16,9))
325         plt.suptitle('Measured distances between the anchors')
326         for i in list(range(len(self.R))):
327             n_samples = len(self.R[i])
328             n_inliers = indexes[i]['inliers'].shape[0]
329             row = int(i/3)
330             col = int(i%3)
331             curr_ax = axarr[row,col]
332             line1 = curr_ax.plot(np.arange(n_samples), np.array(self.R[i]), 'k'
333             , label='data')
334             line4 = curr_ax.plot(np.arange(n_samples)[indexes[i]['inliers']],
335             [i][indexes[i]['inliers'],0], 'bx',label='RANSAC data')
336             line2 = curr_ax.plot(np.arange(n_samples), np.ones(n_samples)*
337             R_fitted[i][0][0],
338             'r-', label='ransac_fit')
339             line3 = curr_ax.plot(np.arange(n_samples), np.ones(n_samples)*R_avg
340             [i], 'g-',
341             label='simple average')
342             curr_title = self.create_subplot_title(row,col,self.coppie[i])
343             curr_ax.set_title(curr_title, fontsize=10)
344             curr_ax.set_ylabel('distance [mm]')
345             if row==0 and col==0:
346                 curr_ax.legend( ncol = 4, shadow=True, bbox_to_anchor=[0, 1.2],
347                 loc='upper left')
348             fig.savefig('autocalibration_results.png')
349             self.R_mis = self.convert_back_to_list(R_fitted)           # Data that
350             # will be given to the
351             # algebraic alg
352             # to
353             # compute
354             # anchor
355             # coordinates
356  

357     else:
358

```

```

341     print("Failed to obtain anchor distances with current RANSAC parameters
342         ")
343 ###### End of RANSAC METHOD #####
344 #####
345 #####
346 #####
347 ##### auxiliary METHODS #####
348 ##########
349 ##########
350 def create_subplot_title(self, row, col, coppia):
351     if row==0:
352         curr_title = 'r'+str(row)+str(col+1)+': '+str(hex(coppia[0]))+', '+
353             str(hex(coppia[1]))
354     if row==1:
355         if col!=2:
356             curr_title = 'r'+str(row)+str(col+2)+': '+str(hex(coppia[0]))+', '+
357                 str(hex(coppia[1]))
358         if col==2:
359             curr_title = 'r'+str(row+1)+str(col+1)+': '+str(hex(coppia[0]))+', '+
360                 str(hex(coppia[1]))
361     return curr_title
362
363 def convert_back_to_list(self, R_fitted):
364     R_aux = [[],[],[],[],[],[]]
365     for i in list(range(len(R_fitted))):
366         R_aux[i] = R_fitted[i].tolist()
367         R_aux[i] = R_aux[i][0][0]
368     return R_aux
369
370 # initializes an empty list of n empty elements: [[], [], ... , []]
371 def init_empty_list(self, n):
372     list_obj = []
373     for i in list(range(n)):
374         list_obj.append([])
375     return list_obj
376
377 # Add the anchors with their coordinates to the device internal list. Each
378 # anchor stores its own
379 # coordinates. This storage will last even if the anchor is turned off.
380 def addAnchors(self):
381     for anchor in self.anchors:
382         aux_id = anchor.network_id
383         if anchor.network_id == self.serial_id:
384             aux_id = None
385         status = self.pozyx.clearDevices(aux_id)
386         status &= self.pozyx.addDevice(anchor, aux_id)
387         status &= self.pozyx.saveNetwork(aux_id)
388         if status == POZYX_FAILURE or status == POZYX_TIMEOUT:
389             raise Exception("Device " + str(hex(aux_id))+": Failed to add
390                             anchors to tag's device list")
391             quit()
392
393 ##### Printing METHODS #####
394 #####
395 #####
396 # Print the tag position in a readable way on the screen
397 def printPosition(self, position, remote_id=None):
398     """Stampa a video il risultato dell'algoritmo di calibrazione del tag"""
399     new_time = int(round(time.time() * 1000))
400     if self.use_remote_tag:
401         if remote_id is not None:
402             print("POS ID {}, x(mm): {} pos.x} y(mm): {} pos.y} z(mm): {} pos.z}".
403                 format(

```

```

403         "0x%0.4x" % remote_id, pos=position), "Delta_t [ms]: ", new_time
404         - self.old_time[remote_id])
405     else:
406         print("Serial pozyx: x(mm): {pos.x} y(mm): {pos.y} z(mm): {pos.z}".
407             format(
408                 "0x%0.4x", pos=position), "Delta_t [ms]: ", new_time - self.
409                 old_time[remote_id])
410     self.old_time[remote_id] = new_time
411
412
413 def printUWBSettings(self, remote_id):
414     bitrate_dict = {PozyxConstants.UWB_BITRATE_110_KBPS : "Bitrate: 110 kb/s, "
415                     ,
416                     PozyxConstants.UWB_BITRATE_850_KBPS : "Bitrate: 850 kb/s, ",
417                     PozyxConstants.UWB_BITRATE_6810_KBPS : "Bitrate: 6810 kb/s, "}
418     prf_dict = {PozyxConstants.UWB_PRF_64_MHZ : "PRF: 64 MHz, ",
419                     PozyxConstants.UWB_PRF_16_MHZ : "PRF: 16 MHz, "}
420     plen_dict = {PozyxConstants.UWB_PLEN_64 : "Preamble length: 64 symbols, ",
421                     PozyxConstants.UWB_PLEN_128 : "Preamble length: 128 symbols, ",
422                     PozyxConstants.UWB_PLEN_256 : "Preamble length: 256 symbols, ",
423                     PozyxConstants.UWB_PLEN_512 : "Preamble length: 512 symbols, ",
424                     PozyxConstants.UWB_PLEN_1024 : "Preamble length: 1024 symbols,
425                     ",
426                     PozyxConstants.UWB_PLEN_1536 : "Preamble length: 1536 symbols,
427                     ",
428                     PozyxConstants.UWB_PLEN_2048 : "Preamble length: 2048 symbols,
429                     ",
430                     PozyxConstants.UWB_PLEN_4096 : "Preamble length: 4096 symbols,
431                     "}
432     protocol_dict = {POZYX_RANGE_PROTOCOL_PRECISION : "Ranging protocol:
433                         PRECISION",
434                         POZYX_RANGE_PROTOCOL_FAST : "Ranging protocol: FAST"}
435     aux = UWBSettings()
436     status = self.pozyx.getUWBSettings(aux, remote_id)
437
438     if status == POZYX_SUCCESS:
439         if remote_id == None:
440             print("Serial ", bitrate_dict[aux.bitrate],
441                  prf_dict[aux.prf], plen_dict[aux.plen], "Gain: ", aux.
442                      gain_db)
443         else:
444             print(hex(remote_id), "Channel: ", aux.channel, bitrate_dict[aux.
445                             bitrate],
446                             prf_dict[aux.prf], plen_dict[aux.plen], "Gain: ", aux.
447                                 gain_db)
448     else:
449         if remote_id == self.serial_id:
450             return
451         print(hex(remote_id), "Failed to receive UWB Settings")
452
453 def printProtocol(self):
454     protocol_dict = {POZYX_RANGE_PROTOCOL_PRECISION : "Ranging protocol:
455                         PRECISION",
456                         POZYX_RANGE_PROTOCOL_FAST : "Ranging protocol: FAST"}
457     print(protocol_dict[self.ranging_protocol])
458
459 # Print anchor coordinates in a readable way
460 def printConfigurationResults(self):
461     print("Configuration finished: ")
462     i=0
463     for anchor in self.anchors:
464         print("ANCHOR", i, ", 0x%0.4x, %s" % (anchor.network_id, str(anchor.pos
465                                         ))))
466         i += 1
467     print()
468     print()
469     print("Please check data figure and close it when you are ready\n\n")
470 #
471 ##### End of class CalibrationPozyx #####

```

```

460
461
462
463
464
465
466
467 ##### MAIN #####
468 ##### Riconosce se ci sono dispositivi Pozyx connessi alla porta USB del PC
469 if __name__ == "__main__":
470     # Riconosce se ci sono dispositivi Pozyx connessi alla porta USB del PC
471     serial_port = get_first_pozyx_serial_port()
472     print("Pozyx device found in port: ", serial_port, '\n')
473     if serial_port is None:
474         raise Exception("No Pozyx connected. Check your USB cable or your driver")
475         quit()
476     pozyx = PozyxSerial(serial_port)
477
478
479 ##### -PARAMETRI - #####
480 ##### Calibrazione automatica-manuale
481 auto_cal = True                                # False = manuale, True =
482             automatica
483 serial_id = 0x675d
484 anchor_ids = [0x6939, 0x6e7a, 0x6e44, 0x6e6c]    # !!! Seguire la convenzione di
485             rangesToPos.py
486
487 r01 = 5448  # Inserire i valori se si vuole fare una calibrazione manuale
488 r02 = 5802
489 r03 = 3600
490 r12 = 4210
491 r13 = 3160
492 r23 = 2682
493
494
495 # Parameters of the ransac algorithm to eliminate outliers
496 N_acq = 100                                     # Percentage of elements extracted at each
497 n_percentage = 0.1                               # iteration of ransac
498 k = 1e4                                         # Maximum number of iterations of ransac
499             algorithm
500 t = 1e5                                         # Tolerance. Values with variance greater than
501             t will not be
502 d_percentage = 0.2                             # added to the initial extracted data
503             # minimum percentage of points to consider
504             ransac iteration good
505 plot_ransac_error = False                      # True se si vogliono mostrare gli errori
506 R_mis = [r01, r02, r03, r12, r13, r23]
507
508 # Ranging protocol: POZYX_RANGE_PROTOCOL_FAST or POZYX_RANGE_PROTOCOL_PRECISION
509 ranging_protocol = POZYX_RANGE_PROTOCOL_PRECISION
510
511 # Impostazioni Ultra Wide Band
512 # Canale: 1,2,3,5 o 7
513 channel = 5                                     # Default value = 5
514
515 # Bitrate, possibili valori:
516     # PozyxConstants.UWB_BITRATE_110_KBPS      Default value
517     # PozyxConstants.UWB_BITRATE_850_KBPS
518     # PozyxConstants.UWB_BITRATE_6810_KBPS
519 bitrate = PozyxConstants.UWB_BITRATE_110_KBPS
520
521 # Pulse repeat frequency, possibili valori:
522     # PozyxConstants.UWB_PRF_64_MHZ      Default value
523     # PozyxConstants.UWB_PRF_16_MHZ
524 prf = PozyxConstants.UWB_PRF_64_MHZ
525
526 # Preamble length of the UWB packets, possibili valori:
527     # PozyxConstants.UWB_PLEN_64
528     # PozyxConstants.UWB_PLEN_128

```

```

527     # PozyxConstants.UWB_PLEN_256
528     # PozyxConstants.UWB_PLEN_512
529     # PozyxConstants.UWB_PLEN_1024      Default value
530     # PozyxConstants.UWB_PLEN_1536
531     # PozyxConstants.UWB_PLEN_2048
532     # PozyxConstants.UWB_PLEN_4096
533 plen = PozyxConstants.UWB_PLEN_1024
534
535     # UWB Gain, possibili valori:
536     # float tra 0 e 67.1 dB
537 gain = 11.5                         # Default value 11.5 dB
538
539 myUWBSettings = UWBSettings(channel, bitrate, prf, plen, gain)
540 ##### -Fine settaggio parametri-
541
542 #####
543 ##### Start autocalibration #####
544 ##### Initialize serial connection with the class CalibrationPozyx
545
546 serial_tag = CalibrationPozyx(pozyx, anchor_ids, serial_id, auto_cal, R_mis,
547                                 ranging_protocol,
548                                 N_acq, n_percentage, k, t, d_percentage,
549                                 plot_ransac_error, myUWBSettings)
550
551     # Performs a network setup: gets anchor distances and performs autocalibration,
552     # save anchor
553     # coordinates.
554     serial_tag.setup()
555
556     # Generate text file Parametri_UWB.txt that will be used by doPositioning.py
557     out_file = open("Parametri_UWB.txt", "w")
558     out_file.write(str(channel)+"\n")
559     out_file.write(str(bitrate)+"\n")
560     out_file.write(str(prf)+"\n")
561     out_file.write(str(plen)+"\n")
562     out_file.write(str(gain)+"\n")
563     out_file.close()

```

## A.6 Positioning

```

1  # Questo script python permette di effettuare un'acquisizione automatica delle
2  # posizioni dei tag,
3  # utilizzando l'algoritmo di posizionamento della pozyx.
4  # La posizione dei tag determinata dall'algoritmo viene comunicata sulla linea UWB
5  # al dispositivo
6  # che si connette alla porta USB del PC. Sullo schermo vengono stampate le
7  # posizioni dei due tag.
8  # E' possibile calibrare le ancore sia manualmente che automaticamente. I parametri
9  # da modificare
10 # si trovano nella funzione "main" in fondo al programma. Il programma pensato
11 # per una
12 # configurazione minima che prevede l'utilizzo di 4 ancore per la calibrazione.
13
14 # Su linux:
15 # da terminale digitare >> sudo python3 test_UWB.py
16
17 import time
18 from rangesToPos import (rangesToPos, average)
19 from pypyzyx import *
20
21 MAX_DISTANCE = 100000
22
23 class SerialTag:
24     def __init__(self, pozyx, anchor_ids, ranging_protocol, myUWBSettings=None,
25                  dimension = POZYX_3D, height=1000, algorithm=POZYX_POS_ALG_UWB_ONLY):
26         self.pozyx = pozyx
27         self.anchor_ids = anchor_ids
28         self.ranging_protocol = ranging_protocol
29         self.dimension = dimension
30         self.height = height

```

```

25     self.algorithm = algorithm
26
27     self.anchors = []
28     self.distances = []
29     self.old_time = {None:0}
30
31     self.UWBSettings = myUWBSettings
32     self.UWBChannel = myUWBSettings.channel
33     self.defaultUWBSettings = UWBSettings(5,0,2,8,11.5)
34
35     self.pos_error = 0          # Conta il numero di errori nel positioning del
36     # tag
37
38
39 def setup(self):
40     self.pozyx.setRangingProtocol(self.ranging_protocol)
41     self.pozyx.setUWBSettings(self.UWBSettings)
42     self.pozyx.setUWBChannel(self.UWBChannel)
43
44     print("Coordinate iniziali dei dispositivi della rete")
45     coordinates = Coordinates()
46     for anchor in self.anchor_ids:
47         status = self.pozyx.getDeviceCoordinates(anchor, coordinates, anchor)
48         if status == POZYX_FAILURE:
49             print ("Device " + str(hex(anchor)) + " list is empty")
50             print(hex(anchor), coordinates.x, coordinates.y, coordinates.z)
51
52     ### Print UWB configuration results
53     for remote in anchor_ids:
54         self.printUWBSettings(remote)
55     self.printUWBSettings(None)
56
57
58     self.addAnchors()           #Salve le ancore nella lista interna
59     # dei dispositivi
60     # dei due tag
61     self.printConfigurationResults()
62     print("Ancore aggiunte alla lista interna dei Tag")
63
64
65 def loop(self):
66     position = Coordinates()
67     status = self.pozyx.doPositioning(position, self.dimension, self.height,
68                                         self.algorithm)
69     if status == POZYX_SUCCESS and position.x != 0 and position.y !=0 and
70         position.z !=0:
71         self.printPosition(position)
72     else:
73         self.pos_error = self.pos_error + 1
74         print("Failed positioning")
75
76
77 def printConfigurationResults(self):
78     print("Anchors identified: ")
79     i=0
80     for anchor in self.anchors:
81         print("ANCHOR", i, ", 0x%0.4x, %s" % (anchor.network_id, str(anchor.pos
82                         ))))
83         i += 1
84     print()
85     print()
86
86 def printPosition(self, position, remote_id=None):
87     """Stampa a video il risultato dell'algoritmo di calibrazione del tag"""
88     new_time = int(round(time.time() * 1000))
89     print("Serial pozyx: x(mm): {pos.x} y(mm): {pos.y} z(mm): {pos.z}".format(
90         pos=position), " Delta_t [ms]: ", new_time - self.old_time
91         [remote_id])
92     self.old_time[remote_id] = new_time

```

```

91
92
93
94     def addAnchors(self):
95         status = self.pozyx.clearDevices()
96         coordinates = Coordinates()
97
98         for anchor_id in anchor_ids:
99             status &= self.pozyx.getDeviceCoordinates(anchor_id, coordinates,
100                                         anchor_id)
101            self.anchors.append(DeviceCoordinates(anchor_id, 1, coordinates))
102            if status != POZYX_SUCCESS:
103                raise Exception(str(hex(anchor_id)) + ": failed to get anchor's
104                               coordinates from anchor's internal list")
105                quit()
106
107            for anchor in self.anchors:
108                status &= self.pozyx.addDevice(anchor)
109            if status == POZYX_FAILURE or status == POZYX_TIMEOUT:
110                raise Exception("Failed to add anchors to tag's device list")
111                quit()
112
113
114     def printUWBSettings(self, remote_id):
115         aux = UWBSettings()
116         status = self.pozyx.getUWBSettings(aux, remote_id)
117         if status == POZYX_SUCCESS:
118             if remote_id == None:
119                 print("Serial ", aux.channel, aux.bitrate, aux.prf, hex(aux.plen),
120                      aux.gain_db)
121             else:
122                 print(hex(remote_id), "Channel: ", aux.channel, "Bitrate: ", aux.
123                       bitrate,
124                         "Prf: ", aux.prf, "Plen: ", hex(aux.plen), "Gain: ", aux.
125                           gain_db)
126             else:
127                 print(hex(remote_id), "Failed to receive UWB Settings")
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
# Riconosce se ci sono dispositivi Pozyx connessi alla porta USB del PC
serial_port = get_first_pozyx_serial_port()
print(serial_port)
if serial_port is None:
    raise Exception("No Pozyx connected. Check your USB cable or your driver")
quit()
pozyx = PozyxSerial(serial_port)

#####
# -PARAMETRI -
#####
# ID delle ancore utilizzate
anchor_ids = [0x6e44, 0x6e7a, 0x6e6c, 0x6939]      # !!! Seguire la convenzione di
                                                       rangesToPos.py

# Ranging protocol: POZYX_RANGE_PROTOCOL_FAST or POZYX_RANGE_PROTOCOL_PRECISION
ranging_protocol = POZYX_RANGE_PROTOCOL_PRECISION

input_file = open("Parametri_UWB.txt")
channel = int(input_file.readline())
bitrate = int(input_file.readline())
prf = int(input_file.readline())
plen = int(input_file.readline())
gain = float(input_file.readline())

myUWBSettings = UWBSettings(channel, bitrate, prf, plen, gain)
# Inizializzo pozyx seriale
serial_tag = SerialTag(pozyx, anchor_ids, ranging_protocol, myUWBSettings)

```

```
158
159     # Effettuo il setup della rete
160     serial_tag.setup()
161
162
163     # Loop Function
164     try:
165         while True:
166             serial_tag.loop()
167     except KeyboardInterrupt:
168         print("interrupted!")
```

## A.7 Calibrazione e positioning

```

1 import math
2 from pypyzyx import (DeviceCoordinates, Coordinates)
3
4 def rangesToPos ( R_mis , anchor_ids ):
5     print( "\nDistances between the anchors, fitted with ransac algorithm: \n [r01,
6         r02,r03,r12,r13,r23] = " , R_mis )
7     print()
8     for r in R_mis:
9         if r<0 :
10             raise Exception("Negative distance inserted '\n' ")
11         return
12     r01 = R_mis[0]
13     r02 = R_mis[1]
14     r03 = R_mis[2]
15     r12 = R_mis[3]
16     r13 = R_mis[4]
17     r23 = R_mis[5]
18
19     C = (r01**2+r02**2-r12**2)/(2*r01*r02)
20
21     if abs(C) <= 1:
22         y1 = r01;
23         y2 = (r01**2 - r12**2 + r02**2)/ (2*r01)
24         aux1 = r02**2 - y2**2
25
26         if aux1 < 0:
27             raise Exception("Too much noise in the measurements, incompatible
28                             distances")
29         return
30
31     x2 = math.sqrt(r02**2 - y2**2)
32     y3 = (r03**2 - r13**2 + y1**2)/(2*y1)
33     x3 = (x2**2 - 2*y3*y2 + y2**2 - r23**2 + r03**2)/(2*x2)
34
35     aux2 = r03**2 - x3**2 - y3**2
36     if aux2 < 0:
37         raise Exception("Too much noise in the measurements, incompatible
38                             distances")
39     return
40
41     z3 = math.sqrt(r03**2 - x3**2 - y3**2)
42
43     y1 = int(y1)
44     x2 = int(x2)
45     y2 = int(y2)
46     x3 = int(x3)
47     y3 = int(y3)
48     z3 = int(z3)
49
50     anchors = [DeviceCoordinates(anchor_ids[0], 1, Coordinates(0, 0, 0)),
51                 DeviceCoordinates(anchor_ids[1], 1, Coordinates(0, y1, 0)),
52                 DeviceCoordinates(anchor_ids[2], 1, Coordinates(x2, y2, 0)),
53                 DeviceCoordinates(anchor_ids[3], 1, Coordinates(x3, y3, z3))]
54     return anchors
55
56
57 def average(x):
58     av = 0
59     if len(x) == 0:
60         raise Exception("Trying to average an empty list")
61     for i in list(range(len(x))):
62         av = av + x[i]
63     av = av / len(x)
64     return av

```

## B Codice esperimento 1

```
1 #!/usr/bin/env python
2
3 from pypyozx import (PozyxSerial, POZYX_RANGE_PROTOCOL_FAST,
4                     POZYX_RANGE_PROTOCOL_PRECISION,
5                     SingleRegister, DeviceRange, POZYX_SUCCESS, POZYX_FAILURE,
6                     get_first_pozyx_serial_port)
7
8
9
10 class ReadyToRange(object):
11     """Continuously performs ranging between the Pozyx and a destination and sets
12     their LEDs"""
13
14     def __init__(self, pozyx, destination_id, range_step_mm=1000, protocol=
15                 POZYX_RANGE_PROTOCOL_FAST, remote_id=None):
16         self.pozyx = pozyx
17         self.destination_id = destination_id
18         self.range_step_mm = range_step_mm
19         self.remote_id = remote_id
20         self.protocol = protocol
21
22     def setup(self):
23         """Sets up both the ranging and destination Pozyx's LED configuration"""
24         print("-----POZYX RANGING V1.1 -----")
25         print("NOTES: ")
26         print(" - Change the parameters: ")
27         print("\tdestination_id(target device)")
28         print("\trange_step(mm)")
29         print()
30         print("- Approach target device to see range and")
31         print("led control")
32         print()
33         self.pozyx.printDeviceInfo(self.remote_id)
34         print()
35         print("START Ranging: ")
36
37         # make sure the local/remote pozyx system has no control over the LEDs.
38         led_config = 0x0
39         self.pozyx.setLedConfig(led_config, self.remote_id)
40         # do the same for the destination.
41         self.pozyx.setLedConfig(led_config, self.destination_id)
42         # set the ranging protocol
43         self.pozyx.setRangingProtocol(self.protocol, self.remote_id)
44
45     def loop(self, output_file=None):
46         """Performs ranging and sets the LEDs accordingly"""
47         device_range = DeviceRange()
48         status = self.pozyx.doRanging(
49             self.destination_id, device_range, self.remote_id)
50         if status == POZYX_SUCCESS:
51             print(device_range)
52             if self.ledControl(device_range.distance) == POZYX_FAILURE:
53                 print("ERROR: setting (remote) leds")
54                 out_file.write(str(device_range.distance)+'\n')      # Scrive sul file di
55                                         output
56         else:
57             error_code = SingleRegister()
58             status = self.pozyx.getErrorCode(error_code)
59             if status == POZYX_SUCCESS:
60                 print("ERROR Ranging, local %s" %
61                       self.pozyx.getErrorMessage(error_code))
62             else:
63                 print("ERROR Ranging, couldn't retrieve local error")
64
65     def ledControl(self, distance):
66         """Sets LEDs according to the distance between two devices"""
67         status = POZYX_SUCCESS
```

```

67     ids = [self.remote_id, self.destination_id]
68     # set the leds of both local/remote and destination pozyx device
69     for id in ids:
70         status &= self.pozyx.setLed(4, (distance < range_step_mm), id)
71         status &= self.pozyx.setLed(3, (distance < 2 * range_step_mm), id)
72         status &= self.pozyx.setLed(2, (distance < 3 * range_step_mm), id)
73         status &= self.pozyx.setLed(1, (distance < 4 * range_step_mm), id)
74     return status
75
76
77 if __name__ == "__main__":
78     # hardcoded way to assign a serial port of the Pozyx
79     serial_port = 'COM12',
80
81     # the easier way
82     serial_port = get_first_pozyx_serial_port()
83     if serial_port is None:
84         print("No Pozyx connected. Check your USB cable or your driver!")
85         quit()
86
87     remote_id = 0x6e44          # the network ID of the remote device
88     remote = False             # whether to use the given remote device for ranging
89     if not remote:
90         remote_id = None
91
92     destination_id = 0x6e7a    # network ID of the ranging destination
93     range_step_mm = 1000       # distance that separates the amount of LEDs lighting
94     up.
95
96     ranging_protocol = POZYX_RANGE_PROTOCOL_PRECISION      # the ranging protocol
97
98     pozyx = PozyxSerial(serial_port)
99     r = ReadyToRange(pozyx, destination_id, range_step_mm,
100                      ranging_protocol, remote_id)
101    r.setup()
102
103    # Check the opening of the file so as not to overwrite the previous data
104    i=1
105    print('distanza_due_antenne_'+str(i)+'.txt')
106    while os.path.isfile('./data/distanza_due_antenne_'+str(i)+'.txt'):
107        print(i)
108        i = i + 1
109
110    # First raw of file .txt
111    name = "./data/distanza_due_antenne_"+str(i)+".txt"
112    out_file = open(name, "w")
113
114    # Data acquisition
115    N = 1000      #number of samples
116    for i in list(range(N)):
117        r.loop(out_file)
118    out_file.close()

```

## C Codice esperimento 2

```
1 #!/usr/bin/env python
2 """
3 The Pozyx ready to range tutorial (c) Pozyx Labs
4 Please read the tutorial that accompanies this sketch: https://www.pozyx.io/
5 Documentation/Tutorials/ready_to_range/Python
6
7 This demo requires two Pozyx devices. It demonstrates the ranging capabilities and
8 the functionality to
9 to remotely control a Pozyx device. Move around with the other Pozyx device.
10
11 This demo measures the range between the two devices. The closer the devices are to
12 each other, the more LEDs will
13 light up on both devices.
14 """
15
16 from pypozyx import (PozyxSerial, POZYX_RANGE_PROTOCOL_FAST,
17     POZYX_RANGE_PROTOCOL_PRECISION,
18     SingleRegister, DeviceRange, POZYX_SUCCESS, POZYX_FAILURE,
19     get_first_pozyx_serial_port, PozyxConstants, Coordinates
20 )
21
22 import os.path
23
24
25 class ReadyToRange(object):
26     """Continuously performs ranging between the Pozyx and a destination and sets
27     their LEDs"""
28
29     def __init__(self, pozyx, destination_id, range_step_mm=1000, protocol=
30         POZYX_RANGE_PROTOCOL_FAST, remote_id=None):
31         self.pozyx = pozyx
32         self.destination_id = destination_id
33         self.range_step_mm = range_step_mm
34         self.remote_id = remote_id
35         self.protocol = protocol
36
37     def setup(self):
38         """Sets up both the ranging and destination Pozyx's LED configuration"""
39         print("-----POZYX RANGING V1.1 -----")
40         print("NOTES: ")
41         print(" - Change the parameters: ")
42         print("\tdestination_id(target device)")
43         print("\trange_step(mm)")
44         print()
45         print(" - Approach target device to see range and")
46         print("led control")
47         print()
48         self.pozyx.printDeviceInfo(self.remote_id)
49         print()
50         print("-----POZYX RANGING V1.1 -----")
51         print()
52         print("START Ranging: ")
53
54         # make sure the local/remote pozyx system has no control over the LEDs.
55         led_config = 0x0
56         self.pozyx.setLedConfig(led_config, self.remote_id)
57         # do the same for the destination.
58         self.pozyx.setLedConfig(led_config, self.destination_id)
59         # set the ranging protocol
60         self.pozyx.setRangingProtocol(self.protocol, self.remote_id)
61
62     def ledControl(self, distance):
63         """Sets LEDs according to the distance between two devices"""
64         status = POZYX_SUCCESS
65         ids = [self.remote_id, self.destination_id]
66         # set the leds of both local/remote and destination pozyx device
67         for id in ids:
68             status &= self.pozyx.setLed(4, (distance < range_step_mm), id)
69             status &= self.pozyx.setLed(3, (distance < 2 * range_step_mm), id)
70             status &= self.pozyx.setLed(2, (distance < 3 * range_step_mm), id)
71             status &= self.pozyx.setLed(1, (distance < 4 * range_step_mm), id)
72
73         return status
74
```

```

64
65 if __name__ == "__main__":
66     # hardcoded way to assign a serial port of the Pozyx
67     serial_port = 'COM12'
68
69     # the easier way
70     serial_port = get_first_pozyx_serial_port()
71     if serial_port is None:
72         print("No Pozyx connected. Check your USB cable or your driver!")
73         quit()
74
75     anchor_id1 = 0x6e7a
76     anchor_id2 = 0x6e44
77     anchor_id3 = 0x6934
78     anchor_id4 = 0x6e1d
79
80     anchor_id1_str = '0x6e7a'
81     anchor_id2_str = '0x6e44',
82     anchor_id3_str = '0x6934',
83     anchor_id4_str = '0x6e1d'
84
85     anchors = [anchor_id1, anchor_id2, anchor_id3, anchor_id4]
86
87     dimension = PozyxConstants.DIMENSION_2_5D
88
89     num_measurements = 50
90
91     remote_id = 0x6e7e
92     remote = False      # whether to use the given remote device for ranging
93     if not remote:
94         remote_id = None
95
96     # range_step_mm = 1000
97
98     # ranging_protocol = POZYX_RANGE_PROTOCOL_PRECISION    # the ranging protocol
99
100    pozyx = PozyxSerial(serial_port)
101    pozyx.doAnchorCalibration(dimension, num_measurements, anchors)
102    coordinates = Coordinates()
103    pozyx.getDeviceCoordinates(anchor_id1, coordinates)
104    print(anchor_id1_str+'\t'+str(coordinates))
105    pozyx.getDeviceCoordinates(anchor_id2, coordinates)
106    print(anchor_id2_str+'\t'+str(coordinates))
107    pozyx.getDeviceCoordinates(anchor_id3, coordinates)
108    print(anchor_id3_str+'\t'+str(coordinates))
109    pozyx.getDeviceCoordinates(anchor_id4, coordinates)
110    print(anchor_id4_str+'\t'+str(coordinates))
111
112    # Controlla l'apertura del file corretto in modo da non sovrascrivere i dati
113    # precedenti
114    i=1
115    print('auto_calibrazione_2D_geometria_'+str(i)+'.txt')
116    while os.path.isfile('./dati/auto_calibrazione_2D_geometria_'+str(i)+'.txt'):
117        print(i)
118        i = i + 1
119
120    # Prima riga del file
121    name = "./dati/auto_calibrazione_2D_geometria_"+str(i)+".txt"
122    out_file = open(name, "w")
123
124    # Acquisizioni
125    N = 10
126    for i in list(range(N)):
127        print(i)
128        pozyx.doAnchorCalibration(dimension, num_measurements, anchors)
129        pozyx.getDeviceCoordinates(anchor_id1, coordinates)
130        out_file.write(anchor_id1_str+'\t'+str(coordinates)+'\n')
131        pozyx.getDeviceCoordinates(anchor_id2, coordinates)
132        out_file.write(anchor_id2_str+'\t'+str(coordinates)+'\n')
133        pozyx.getDeviceCoordinates(anchor_id3, coordinates)
134        out_file.write(anchor_id3_str+'\t'+str(coordinates)+'\n')
135        pozyx.getDeviceCoordinates(anchor_id4, coordinates)
136        out_file.write(anchor_id4_str+'\t'+str(coordinates)+'
```

136 | out\_file.close()

## D Codice esperimento 4

```
1 #!/usr/bin/env python
2
3 from pypyozx import (PozyxSerial, POZYX_RANGE_PROTOCOL_FAST,
4     POZYX_RANGE_PROTOCOL_PRECISION,
5     SingleRegister, DeviceRange, POZYX_SUCCESS, POZYX_FAILURE,
6     get_first_pozyx_serial_port)
7
8 import os.path
9
10 class ReadyToRange(object):
11     """Continuously performs ranging between the Pozyx and a destination and sets
12     their LEDs"""
13
14     def __init__(self, pozyx, anchor_ids, protocol=POZYX_RANGE_PROTOCOL_FAST,
15         remote_id=None):
16         self.pozyx = pozyx
17         self.anchor_ids = anchor_ids
18         self.protocol = protocol
19         self.remote_id = remote_id
20
21     def setup(self):
22         # set the ranging protocol
23         for remote in anchor_ids:
24             self.pozyx.setRangingProtocol(self.protocol, remote)
25             self.pozyx.setRangingProtocol(self.protocol, self.remote_id)
26
27     def loop(self, output_file=None):
28         """Performs ranging and sets the LEDs accordingly"""
29         device_range = DeviceRange()
30
31         for remote in anchor_ids:
32             for anchor in anchor_ids:
33                 if remote != anchor:
34                     status = self.pozyx.doRanging(anchor, device_range, remote)
35
36                     if status == POZYX_SUCCESS:
37                         if device_range.distance <= 100000:
38                             out_file.write(str(device_range.distance)+'\t\t')
39                         else:
40                             out_file.write('Err\t\t')
41
42                     else:
43                         error_code = SingleRegister()
44                         status = self.pozyx.getErrorCode(error_code)
45                         if status == POZYX_SUCCESS:
46                             print("ERROR Ranging, local %s" %
47                                 self.pozyx.getErrorMessage(error_code))
48                         else:
49                             print("ERROR Ranging, couldn't retrieve local error")
50                         out_file.write('Err\t\t')
51
52         out_file.write('\n')
53
54 if __name__ == "__main__":
55
56     #Connessione alla seriale del pozyx
57     serial_port = get_first_pozyx_serial_port()
58     if serial_port is None:
59         print("No Pozyx connected. Check your USB cable or your driver!")
60         quit()
61
62     anchor0 = 0x6939
63     anchor1 = 0x6e7a
64     anchor2 = 0x6e44
65     anchor3 = 0x6e6c
66     anchor_ids = [anchor0, anchor1, anchor2, anchor3]
```

```

64     remote_id = 0x6760           # remote device network ID
65     remote = False              # whether to use a remote device
66     if not remote:
67         remote_id = None
68
69     ranging_protocol = POZYX_RANGE_PROTOCOL_PRECISION    # the ranging protocol
70
71     pozyx = PozyxSerial(serial_port)
72     r = ReadyToRange(pozyx, anchor_ids, ranging_protocol, remote_id)
73     r.setup()
74
75     #Controlla l'apertura del file corretto in modo da non sovrascrivere i dati
76     #precedenti
77     i=0
78     print('distanza_coppie_antenne_'+str(i)+'.txt')
79     while os.path.isfile('./dati/distanza_coppie_antenne_'+str(i)+'.txt'):
80         print(i)
81         i = i + 1
82
83     #Prima riga del file
84     name = "./dati/distanza_coppie_antenne_"+str(i)+".txt"
85     out_file = open(name, "w")
86     for i in list(range(4)):
87         for j in list (range(4)):
88             if (i!=j):
89                 out_file.write('r'+str(i)+str(j)+'\t\t')
90     out_file.write('\n')
91
92     #Acquisizioni
93     N = 20                      #Number of acquisitions
94     for i in list(range(N)):
95         print(i)
96         r.loop(out_file)
97     out_file.close()

```

## E Codice esperimento 5

```

1  #!/usr/bin/env python
2
3  from pypozyx import (PozyxSerial, POZYX_RANGE_PROTOCOL_FAST,
4                        POZYX_RANGE_PROTOCOL_PRECISION,
5                        SingleRegister, DeviceRange, POZYX_SUCCESS, POZYX_FAILURE,
6                        get_first_pozyx_serial_port)
7
8
9  class ReadyToRange(object):
10    """Continuously performs ranging between the Pozyx and a destination and sets
11       their LEDs"""
12
13    def __init__(self, pozyx, anchor_ids, protocol=POZYX_RANGE_PROTOCOL_FAST,
14                 remote_id=None):
15        self.pozyx = pozyx
16        self.anchor_ids = anchor_ids
17        self.protocol = protocol
18        self.remote_id = remote_id
19
20    def setup(self):
21        # set the ranging protocol
22        for remote in anchor_ids:
23            self.pozyx.setRangingProtocol(self.protocol, remote)
24            self.pozyx.setRangingProtocol(self.protocol, self.remote_id)
25
26    def loop(self, output_file=None):
27        """Performs ranging and sets the LEDs accordingly"""
28        device_range = DeviceRange()
29
29        for remote in anchor_ids:
            for anchor in anchor_ids:

```

```

30     if remote != anchor:
31         status = self.pozyx.doRanging(anchor, device_range, remote)
32
33         if status == POZYX_SUCCESS:
34             if device_range.distance <= 100000:
35                 out_file.write(str(device_range.distance)+'\t\t')
36             else:
37                 out_file.write('Err\t\t')
38         else:
39             error_code = SingleRegister()
40             status = self.pozyx.getErrorCode(error_code)
41             if status == POZYX_SUCCESS:
42                 print("ERROR Ranging, local %s" %
43                     self.pozyx.getErrorMessage(error_code))
44             else:
45                 print("ERROR Ranging, couldn't retrieve local error")
46             out_file.write('Err\t\t')
47         out_file.write('\n')
48
49
50 if __name__ == "__main__":
51
52     #Connessione alla seriale del pozyx
53     serial_port = get_first_pozyx_serial_port()
54     if serial_port is None:
55         print("No Pozyx connected. Check your USB cable or your driver!")
56         quit()
57
58     anchor0 = 0x6939
59     anchor1 = 0x6e7a
60     anchor2 = 0x6e44
61     anchor3 = 0x6e6c
62     anchor_ids = [anchor0, anchor1, anchor2, anchor3]
63
64     remote_id = 0x6760          # remote device network ID
65     remote = False              # whether to use a remote device
66     if not remote:
67         remote_id = None
68
69     ranging_protocol = POZYX_RANGE_PROTOCOL_PRECISION    # the ranging protocol
70
71     pozyx = PozyxSerial(serial_port)
72     r = ReadyToRange(pozyx, anchor_ids, ranging_protocol, remote_id)
73     r.setup()
74
75     #Controlla l'apertura del file corretto in modo da non sovrascrivere i dati
76     #precedenti
77     i=1
78     print('distanza_coppie_antenne_'+str(i)+'.txt')
79     while os.path.isfile('./dati/distanza_coppie_antenne_'+str(i)+'.txt'):
80         print(i)
81         i = i + 1
82
83     #Prima riga del file
84     name = "./dati/distanza_coppie_antenne_"+str(i)+".txt"
85     out_file = open(name, "w")
86     for i in list(range(4)):
87         for j in list(range(4)):
88             if (i!=j):
89                 out_file.write('r'+str(i)+str(j)+'\t\t')
90     out_file.write('\n')
91
92     #Acquisizioni
93     N = 20                      #Number of acquisitions
94     for i in list(range(N)):
95         print(i)
96         r.loop(out_file)
97     out_file.close()

```

```

1 #!/usr/bin/env python
2 from time import sleep
3

```

```

4  from pypozyx import (POZYX_POS_ALG_UWB_ONLY, POZYX_3D, Coordinates, POZYX_SUCCESS,
5      POZYX_ANCHOR_SEL_AUTO,
6          DeviceCoordinates, PozyxSerial, get_first_pozyx_serial_port,
7              SingleRegister, DeviceList, DeviceRange)
8  from pythonosc.udp_client import SimpleUDPCClient
9  import os.path
10
11 class ReadyToLocalize(object):
12     """Continuously calls the Pozyx positioning function and prints its position.
13         """
14
15     def __init__(self, pozyx, osc_udp_client, anchors, anchor_ids, algorithm=
16         POZYX_POS_ALG_UWB_ONLY, dimension=POZYX_3D, height=1000, remote_id=None):
17         self.pozyx = pozyx
18         self.osc_udp_client = osc_udp_client
19         self.anchor_ids = anchor_ids
20
21         self.anchors = anchors
22         self.algorithm = algorithm
23         self.dimension = dimension
24         self.height = height
25         self.remote_id = remote_id
26
27     def setup(self):
28         """Sets up the Pozyx for positioning by calibrating its anchor list."""
29         print("-----POZYX POSITIONING V1.1 -----")
30         print("NOTES: ")
31         print("- No parameters required.")
32         print()
33         print("- System will auto start configuration")
34         print()
35         print("- System will auto start positioning")
36         print()
37         self.pozyx.printDeviceInfo(self.remote_id)
38         print()
39         print("-----POZYX POSITIONING V1.1 -----")
40         print()
41         self.pozyx.clearDevices(self.remote_id)
42
43     def loop(self, out_file=None):
44         """Performs positioning and displays(exports) the results."""
45         position = Coordinates()
46         range = DeviceRange()
47         status = self.pozyx.doPositioning(
48             position, self.dimension, self.height, self.algorithm, remote_id=self.
49                 remote_id)
50         if status == POZYX_SUCCESS:
51             out_file.write(str(position.x)+'\t'+str(position.y)+'\t'+str(position.z)
52                             +'\t')
53
54             count = 0
55             for anchor in self.anchor_ids:
56                 self.pozyx.getDeviceRangeInfo(anchor, range, remote_id)
57                 out_file.write(str(range.distance)+'\t'+str(range.RSS)+'\t'+str(
58                     range.timestamp)+'\t')
59                 count = count + 1
60                 out_file.write('\n')
61             else:
62                 self.printPublishErrorCode("positioning")
63
64     def printPublishPosition(self, position):
65         """Prints the Pozyx's position and possibly sends it as a OSC packet"""
66         network_id = self.remote_id
67         if network_id is None:
68             network_id = 0
69         print("POS ID {}, x(mm): {} y(mm): {} z(mm): {}".format(
70             "0x%0.4x" % network_id, pos=position))
71         if self.osc_udp_client is not None:

```

```

70         self.osc_udp_client.send_message(
71             "/position", [network_id, int(position.x), int(position.y), int(
72                 position.z)])
73
74     def printPublishErrorCode(self, operation):
75         """Prints the Pozyx's error and possibly sends it as a OSC packet"""
76         error_code = SingleRegister()
77         network_id = self.remote_id
78         if network_id is None:
79             self.pozyx.getErrorCode(error_code)
80             print("LOCAL ERROR %s, %s" % (operation, self.pozyx.getErrorMessage(
81                 error_code)))
82             if self.osc_udp_client is not None:
83                 self.osc_udp_client.send_message("/error", [operation, 0,
84                     error_code[0]])
85             return
86         status = self.pozyx.getErrorCode(error_code, self.remote_id)
87         if status == POZYX_SUCCESS:
88             print("ERROR %s on ID %s, %s" %
89                 (operation, "0x%0.4x" % network_id, self.pozyx.getErrorMessage(
90                     error_code)))
91             if self.osc_udp_client is not None:
92                 self.osc_udp_client.send_message(
93                     "/error", [operation, network_id, error_code[0]])
94             else:
95                 self.pozyx.getErrorCode(error_code)
96                 print("ERROR %s, couldn't retrieve remote error code, LOCAL ERROR %s" %
97                     (operation, self.pozyx.getErrorMessage(error_code)))
98             if self.osc_udp_client is not None:
99                 self.osc_udp_client.send_message("/error", [operation, 0, -1])
100            # should only happen when not being able to communicate with a remote
101            # Pozyx.
102
103     def setAnchorsManual(self):
104         """Adds the manually measured anchors to the Pozyx's device list one for
105         one."""
106         status = self.pozyx.clearDevices(self.remote_id)
107         for anchor in self.anchors:
108             status &= self.pozyx.addDevice(anchor, self.remote_id)
109         if len(self.anchors) > 4:
110             status &= self.pozyx.setSelectionOfAnchors(POZYX_ANCHOR_SEL_AUTO, len(
111                 self.anchors))
112         return status
113
114     def printPublishConfigurationResult(self):
115         """Prints and potentially publishes the anchor configuration result in a
116         human-readable way."""
117         list_size = SingleRegister()
118
119         self.pozyx.getDeviceListSize(list_size, self.remote_id)
120         print("List size: {}".format(list_size[0]))
121         if list_size[0] != len(self.anchors):
122             self.printPublishErrorCode("configuration")
123             return
124         device_list = DeviceList(list_size=list_size[0])
125         self.pozyx.getDeviceIds(device_list, self.remote_id)
126         print("Calibration result:")
127         print("Anchors found: {}".format(list_size[0]))
128         print("Anchor IDs: ", device_list)
129
130         for i in range(list_size[0]):
131             anchor_coordinates = Coordinates()
132             self.pozyx.getDeviceCoordinates(device_list[i], anchor_coordinates,
133                 self.remote_id)
134             print("ANCHOR, 0x%0.4x, %s" % (device_list[i], str(anchor_coordinates)))
135             if self.osc_udp_client is not None:
136                 self.osc_udp_client.send_message(
137                     "/anchor", [device_list[i], int(anchor_coordinates.x), int(
138                         anchor_coordinates.y), int(anchor_coordinates.z)])
139             sleep(0.025)
140
141     def printPublishAnchorConfiguration(self):

```

```

132     """Prints and potentially publishes the anchor configuration"""
133     for anchor in self.anchors:
134         print("ANCHOR,0x%0.4x,%s" % (anchor.network_id, str(anchor.coordinates)
135             ))
136         if self.osc_udp_client is not None:
137             self.osc_udp_client.send_message(
138                 "/anchor", [anchor.network_id, int(anchor.coordinates.x), int(
139                     anchor.coordinates.y), int(anchor.coordinates.z)])
140         sleep(0.025)
141
142 if __name__ == "__main__":
143     # shortcut to not have to find out the port yourself
144
145     serial_port = get_first_pozyx_serial_port()
146     print(serial_port)
147     if serial_port is None:
148         print("No Pozyx connected. Check your USB cable or your driver!")
149         quit()
150
151     remote_id = 0x6760          # remote device network ID
152     remote = False              # whether to use a remote device
153     if not remote:
154         remote_id = None
155
156     use_processing = False      # enable to send position data through OSC
157     ip = "127.0.0.1"           # IP for the OSC UDP
158     network_port = 8888         # network port for the OSC UDP
159     osc_udp_client = None
160     if use_processing:
161         osc_udp_client = SimpleUDPClient(ip, network_port)
162     # necessary data for calibration, change the IDs and coordinates yourself
163
164     anchor_ids = [0x6939, 0x6e7a, 0x6e44, 0x6e6c]
165     anchors = [DeviceCoordinates(anchor_ids[0], 1, Coordinates(0, 0, 0)),
166                DeviceCoordinates(anchor_ids[1], 1, Coordinates(0,5514, 0)),
167                DeviceCoordinates(anchor_ids[2], 1, Coordinates(3900, 4544, 0)),
168                DeviceCoordinates(anchor_ids[3], 1, Coordinates(1957, 3138, 1113))]
169
170     algorithm = POZYX_POS_ALG_UWB_ONLY # positioning algorithm to use
171     dimension = POZYX_3D            # positioning dimension
172     height = 1000                  # height of device, required in 2.5D
173     # positioning
174
175     pozyx = PozyxSerial(serial_port)
176     r = ReadyToLocalize(pozyx, osc_udp_client, anchors, anchor_ids, algorithm,
177         dimension, height, remote_id)
178     r.setup()
179
180     #Controlla l'apertura del file corretto in modo da non sovrascrivere i dati
181     # precedenti
182     i=1
183     print('PosTag_'+str(i)+'.txt')
184     while os.path.isfile('./dati/PosTag_'+str(i)+'.txt'):
185         print(i)
186         i = i + 1
187
188     #Prima riga del file
189     name = "./dati/PosTag_"+str(i)+".txt"
190     out_file = open(name, "w")
191
192     #Acquisizioni
193     N = 10 #Number of acquisitions
194     out_file.write('X [mm]+'\t+'Y [mm]+'\t+'Z [mm]+'\t+'AO \t RSS \t Time \
195         \t A1 \t RSS \t Time \t A2 \t RSS \t Time \t A3 \t RSS \t Time \t \n')
196     for i in list(range(N)):
197         print(i)
198         r.loop(out_file)
199     out_file.close()

```

## F Codice esperimento 6

```

1 import math
2 from pypyzyx import (DeviceCoordinates, Coordinates)
3
4 def rangesToPos ( R_mis, anchor_ids ):
5     print("Vettore delle distanze tra le ancora: \n [r01,r02,r03,r12,r13,r23] =", R_mis)
6     print()
7     for r in R_mis:
8         if r<0 :
9             raise Exception("Hai inserito una distanza negativa '\n' ")
10        return
11    r01 = R_mis[0]
12    r02 = R_mis[1]
13    r03 = R_mis[2]
14    r12 = R_mis[3]
15    r13 = R_mis[4]
16    r23 = R_mis[5]
17
18    C = (r01**2+r02**2-r12**2)/(2*r01*r02)
19
20    if abs(C) <= 1:
21        y1 = r01;
22        y2 = (r01**2 - r12**2 + r02**2)/ (2*r01)
23        aux1 = r02**2 - y2**2
24
25        if aux1 < 0:
26            raise Exception("Misurazioni tra le ancora affette da troppo rumore")
27            return
28
29        x2 = math.sqrt(r02**2 - y2**2)
30        y3 = (r03**2 - r13**2 + y1**2)/(2*y1)
31        x3 = (x2**2 - 2*y3*y2 + y2**2 - r23**2 + r03**2)/(2*x2)
32
33        aux2 = r03**2 - x3**2 - y3**2
34        if aux2 < 0:
35            raise Exception("Misurazioni tra le ancora affette da troppo rumore")
36            return
37
38        z3 = math.sqrt(r03**2 - x3**2 - y3**2)
39
40        y1 = int(y1)
41        x2 = int(x2)
42        y2 = int(y2)
43        x3 = int(x3)
44        y3 = int(y3)
45        z3 = int(z3)
46
47        anchors = [DeviceCoordinates(anchor_ids[0], 1, Coordinates(0, 0, 0)),
48                    DeviceCoordinates(anchor_ids[1], 1, Coordinates(0, y1, 0)),
49                    DeviceCoordinates(anchor_ids[2], 1, Coordinates(x2, y2, 0)),
50                    DeviceCoordinates(anchor_ids[3], 1, Coordinates(x3, y3, z3))]
51        return anchors
52    else:
53        raise Exception("Distanze incompatibili, autocalibrazione fallita")
54    return
55
56
57
58 def average(x):
59    av = 0
60    if len(x) == 0:
61        raise Exception("Error in anchor distances, check anchors' IDs")
62    for i in list(range(len(x))):
63        av = av + x[i]
64    av = av / len(x)
65    return av

```

```

1 # Questo script python permette di effettuare un'acquisizione automatica delle
2 # posizioni dei tag,
2 # utilizzando l'algoritmo di posizionamento della pozzyx.

```

```

3 # La posizione dei tag determinata dall'algoritmo viene comunicata sulla linea UWB
4 # al dispositivo
5 # chU+FFFDonnesso alla porta USB del PC. Sullo schermo vengono stampate le
6 # posizioni dei due tag.
7 # E' possibile calibrare le ancora sia manualmente che automaticamente. I parametri
8 # da modificare
9 # si trovano nella funzione "main" in fondo al programma. Il programmaU+FFFDensato
10 # per una
11 # configurazione minima che prevede l'utilizzo di 4 ancora per la calibrazione.
12 # Si consiglia di creare le cartelle desiderate e di inserire il relativo path
13
14 import time
15 from rangesToPos import (rangesToPos, average)
16 from pyozyx import *
17 import os.path
18
19 MAX_DISTANCE = 100000
20
21 class SerialTag:
22     def __init__(self, pozyx, anchor_ids, remote_tag_ids, auto_cal, R_mis,
23                  ranging_protocol,
24                  N_acq, use_remote_tag, myUWBSettings=None, dimension = POZYX_3D, height=1000,
25                  algorithm=POZYX_POS_ALG_UWB_ONLY,
26                  update_interval = 100):
27         self.pozyx = pozyx
28         self.anchor_ids = anchor_ids
29         self.remote_tag_ids = remote_tag_ids
30         self.auto_cal = auto_cal
31         self.R_mis = R_mis
32         self.ranging_protocol = ranging_protocol
33         self.dimension = dimension
34         self.height = height
35         self.algorithm = algorithm
36         self.N_acq = N_acq
37
38         self.anchors = []
39         self.distances = []
40         self.old_time = {remote_tag_ids[0]:0, remote_tag_ids[1] :0}
41
42         self.UWBSettings = myUWBSettings
43         self.UWBChannel = myUWBSettings.channel
44         self.use_remote_tag = use_remote_tag
45         self.defaultUWBSettings = UWBSettings(5,0,2,8,11.5)
46
47         self.pos_error = 0          # Conta il numero di errori nel positioning del
48                                     # tag
49
50     def setup(self):
51         # Set UWB Parameters
52         for remote in anchor_ids:
53             self.pozyx.setRangingProtocol(self.ranging_protocol, remote)
54             self.pozyx.setUWBSettings(self.UWBSettings, remote)
55             self.pozyx.setUWBChannel(self.UWBChannel, remote)
56
57         for tag in remote_tag_ids:
58             if tag is not None:
59                 self.pozyx.setRangingProtocol(self.ranging_protocol, tag)
60                 self.pozyx.setUWBSettings(self.UWBSettings, remote)
61                 self.pozyx.setUWBChannel(self.UWBChannel, tag)
62
63         self.pozyx.setRangingProtocol(self.ranging_protocol)
64         self.pozyx.setUWBSettings(self.UWBSettings)
65         self.pozyx.setUWBChannel(self.UWBChannel)
66
67         ### Print UWB configuration results
68         for remote in anchor_ids:
69             self.printUWBSettings(remote)

```

```

69     for tag in remote_tag_ids:
70         self.printUWBSettings(tag)
71     self.printUWBSettings(None)
72
73     # Perform autocalibration if needed
74     if not self.auto_cal:
75         self.anchors = rangesToPos(self.R_mis, self.anchor_ids)
76     else:
77         print("Performing autocalibration")
78         self.getDistances()
79         self.anchors = rangesToPos(self.R_mis, self.anchor_ids)
80
81     self.printConfigurationResults()
82     self.addAnchors()                      #Salve le ancore nella lista interna
83                           dei dispositivi
84                           #dei due tag
85     print("Ancore aggiunte alla lista interna dei Tag")
86
87
88 def loop(self, out_file):
89     if use_remote_tag:
90         for tag in self.remote_tag_ids:
91             position = Coordinates()
92             device_range = DeviceRange()
93
94             if tag is not None:
95                 status = self.pozyx.doPositioning(position, self.dimension,
96                                         self.height, self.algorithm, tag)
97
98                 if status == POZYX_SUCCESS and position.x != 0 and position.y
99                     !=0 and position.z !=0:
100                     self.writePosition(position, tag, out_file)
101
102                     count = 0
103                     for anchor in self.anchor_ids:
104                         self.pozyx.getDeviceInfo(anchor, device_range,
105                                         remote_id)
106                         out_file.write(str(device_range.distance)+'\t')
107                         count = count + 1
108                         out_file.write('\n')
109                     else:
110                         self.pos_error = self.pos_error + 1
111                         print("Failed Positioning")
112
113             else:
114                 position = Coordinates()
115                 device_range = DeviceRange()
116                 status = self.pozyx.doPositioning(position, self.dimension, self.height
117                                         , self.algorithm)
118                 if status == POZYX_SUCCESS and position.x != 0 and position.y !=0 and
119                     position.z !=0:
120                     self.writePosition(position, None, out_file)
121
122                     count = 0
123                     for anchor in self.anchor_ids:
124                         self.pozyx.getDeviceInfo(anchor, device_range)
125                         out_file.write(str(device_range.distance)+'\t')
126                         count = count + 1
127                         out_file.write('\n')
128                     else:
129                         self.pos_error = self.pos_error + 1
130                         print("Failed positioning")
131
132
133     def getDistances(self):
134         device_range1 = DeviceRange()
135         device_range2 = DeviceRange()
136         R = [[],[],[],[],[],[]]
137
138         coppie = [ [anchor_ids[0], anchor_ids[1]],
139                   [anchor_ids[0], anchor_ids[2]],
140                   [anchor_ids[0], anchor_ids[3]],
141

```

```

136             [anchor_ids[1], anchor_ids[2]],
137             [anchor_ids[1], anchor_ids[3]],
138             [anchor_ids[2], anchor_ids[3]] ]
139
140     for k in list(range(self.N_acq)):
141         i = 0
142         for coppia in coppie:
143             status1 = self.pozyx.doRanging(coppia[0], device_range1, coppia[1])
144             status2 = self.pozyx.doRanging(coppia[1], device_range2, coppia[0])
145
146
147             if ((status1 == POZYX_SUCCESS) and (device_range1.distance <=
148                 MAX_DISTANCE)):
149                 if ((status2 == POZYX_SUCCESS) and (device_range2.distance <=
150                     MAX_DISTANCE)):
151                     dist = (device_range1.distance + device_range2.distance)/2
152                     R[i].append(dist)
153                     i = i+1
154
155     for i in list(range(len(R))):
156         R[i] = average(R[i])
157     self.R_mis = R
158
159
160     def printConfigurationResults(self):
161         print("Configuration finished: ")
162         i=0
163         for anchor in self.anchors:
164             print("ANCHOR", i, ", 0x%0.4x, %s" % (anchor.network_id, str(anchor.pos
165                                         ))))
166             i += 1
167         print()
168         print()
169
170     def printPosition(self, position, remote_id=None):
171         """Stampa a video il risultato dell'algoritmo di calibrazione del tag"""
172         new_time = int(round(time.time() * 1000))
173         if self.use_remote_tag:
174             if remote_id is not None:
175                 print("POS ID {}, x(mm): {} y(mm): {} z(mm): {}".
176                         format(
177                             "0x%0.4x" % remote_id, pos=position), " Delta_t [ms]: ", new_time
178                             - self.old_time[remote_id])
179             else:
180                 print("Serial pozyx: x(mm): {} y(mm): {} z(mm): {}".
181                         format(
182                             "0x%0.4x", pos=position), " Delta_t [ms]: ", new_time - self.
183                             old_time[remote_id])
184             self.old_time[remote_id] = new_time
185
186     def writePosition(self, position, remote_id = None, out_file = None):
187         new_time = int(round(time.time() * 1000))
188         out_file.write(str(position.x)+'\t'+str(position.y)+'\t'+str(position.z)+'\t'
189                         + str(new_time - self.old_time[remote_id])+'\t')
190         self.old_time[remote_id] = new_time
191
192     def addAnchors(self):
193         for tag in self.remote_tag_ids:
194             status = self.pozyx.clearDevices(tag)
195             for anchor in self.anchors:
196                 status &= self.pozyx.addDevice(anchor, tag)
197             if status == POZYX_FAILURE or status == POZYX_TIMEOUT:
198                 raise Exception("Failed to add anchors to tag's device list")
199                 quit()
200
201
202     def printUWBSettings(self, remote_id):
203         aux = UWBSettings()
204         status = self.pozyx.getUWBSettings(aux, remote_id)

```

```

201     if status == POZYX_SUCCESS:
202         if remote_id == None:
203             print("Serial ", aux.channel, aux.bitrate, aux.prf, hex(auxplen),
204                  aux.gain_db)
205         else:
206             print(hex(remote_id), "Channel: ", aux.channel, "Bitrate: ", aux.
207                   bitrate,
208                   "Prf: ", aux.prf, "Plen: ", hex(auxplen), "Gain: ", aux.
209                   gain_db)
210     else:
211         print(hex(remote_id), "Failed to receive UWB Settings")
212
213
214 def write_uwb_parameters(myUWBSettings, protocol, out_file = None):
215     bitrate_dict = {PozyxConstants.UWB_BITRATE_110_KBPS : "Bitrate: 110 kb/s",
216                     PozyxConstants.UWB_BITRATE_850_KBPS : "Bitrate: 850 kb/s",
217                     PozyxConstants.UWB_BITRATE_6810_KBPS : "Bitrate: 6810 kb/s"}
218     prf_dict = {PozyxConstants.UWB_PRF_64_MHZ : "PRF: 64 MHz",
219                 PozyxConstants.UWB_PRF_16_MHZ : "PRF: 16 MHz"}
220     plen_dict = {PozyxConstants.UWB_PLEN_64 : "Preamble length: 64 symbols",
221                   PozyxConstants.UWB_PLEN_128 : "Preamble length: 128 symbols",
222                   PozyxConstants.UWB_PLEN_256 : "Preamble length: 256 symbols",
223                   PozyxConstants.UWB_PLEN_512 : "Preamble length: 512 symbols",
224                   PozyxConstants.UWB_PLEN_1024 : "Preamble length: 1024 symbols",
225                   PozyxConstants.UWB_PLEN_1536 : "Preamble length: 1536 symbols",
226                   PozyxConstants.UWB_PLEN_2048 : "Preamble length: 2048 symbols",
227                   PozyxConstants.UWB_PLEN_4096 : "Preamble length: 4096 symbols"}
228     protocol_dict = {POZYX_RANGE_PROTOCOL_PRECISION : "Ranging protocol: PRECISION"
229                      ,
230                      POZYX_RANGE_PROTOCOL_FAST : "Ranging protocol: FAST"}
231     out_file.write("##### UWB SETTINGS #####\n")
232     out_file.write(protocol_dict[protocol] + '\n')
233     out_file.write("Channel: " + str(myUWBSettings.channel) + '\n')
234     out_file.write(bitrate_dict[myUWBSettings.bitrate] + '\n')
235     out_file.write(prf_dict[myUWBSettings.prf] + '\n')
236     out_file.write(plen_dict[myUWBSettings.plen] + '\n')
237     out_file.write("Gain: " + str(myUWBSettings.gain_db) + "\n")
238     out_file.write("#####\n\n")
239
240
241
242
243
244
245
246 if __name__ == "__main__":
247     # Riconosce se ci sono dispositivi Pozyx connessi alla porta USB del PC
248     serial_port = get_first_pozyx_serial_port()
249     print(serial_port)
250     if serial_port is None:
251         raise Exception("No Pozyx connected. Check your USB cable or your driver")
252         quit()
253     pozyx = PozyxSerial(serial_port)
254
255
256 ##### - PARAMETRI - #####
257 ##### - PARAMETRI - #####
258 ##### - PARAMETRI - #####
259 ##### - PARAMETRI - #####
260 ##### - PARAMETRI - #####
261 ##### - PARAMETRI - #####
262 ##### - PARAMETRI - #####
263 ##### - PARAMETRI - #####
264 ##### - PARAMETRI - #####
265 ##### - PARAMETRI - #####
266 ##### - PARAMETRI - #####
267 ##### - PARAMETRI - #####
268 ##### - PARAMETRI - #####

```

```

269     i = i+1
270
271     # Calibrazione automatica-manuale
272     auto_cal = False                                # False = manuale, True =
273     anchor_ids = [0x6e44, 0x6e7a, 0x6e6c, 0x6939]    # !!! Seguire la convenzione di
274     rangesToPos.py
275
276     r01 = 5650  # Inserire i valori se si vuole fare una calibrazione manuale
277     r02 = 6111
278     r03 = 3830
279     r12 = 4150
280     r13 = 3176
281     r23 = 2717
282     N_acq = 30
283     R_mis = [r01, r02, r03, r12, r13, r23]
284
285     # Ranging protocol: POZYX_RANGE_PROTOCOL_FAST or POZYX_RANGE_PROTOCOL_PRECISION
286     ranging_protocol = POZYX_RANGE_PROTOCOL_PRECISION
287
288     # Impostazioni Ultra Wide Band
289     # Canale: 1,2,3,5 o 7
290     channel = 7          # Default value = 5
291
292     # Bitrate, possibili valori:
293     # PozyxConstants.UWB_BITRATE_110_KBPS      Default value
294     # PozyxConstants.UWB_BITRATE_850_KBPS
295     # PozyxConstants.UWB_BITRATE_6810_KBPS
296     bitrate = PozyxConstants.UWB_BITRATE_850_KBPS
297
298     # Pulse repeat frequency, possibili valori:
299     # PozyxConstants.UWB_PRF_64_MHZ      Default value
300     # PozyxConstants.UWB_PRF_16_MHZ
301     prf = PozyxConstants.UWB_PRF_64_MHZ
302
303     # Preamble length of the UWB packets, possibili valori:
304     # PozyxConstants.UWB_PLEN_64
305     # PozyxConstants.UWB_PLEN_128
306     # PozyxConstants.UWB_PLEN_256
307     # PozyxConstants.UWB_PLEN_512
308     # PozyxConstants.UWB_PLEN_1024      Default value
309     # PozyxConstants.UWB_PLEN_1536
310     # PozyxConstants.UWB_PLEN_2048
311     # PozyxConstants.UWB_PLEN_4096
312     plen = PozyxConstants.UWB_PLEN_128
313
314     # UWB Gain, possibili valori:
315     # float tra 0 e 67.1 dB
316     gain = 11.5          # Default value 11.5 dB
317
318     myUWBSettings = UWBSettings(channel, bitrate, prf, plen, gain)
319     ##### -Fine settaggio parametri- #####
320
321     # Inserire "" alla linea 324 e alla linea 379 per testare o alle linee 385 e
322     # 471. Nle primo
323     # caso verra testato un singolo insieme di parametri, nel secondo tutti i
324     # possibili parametri ad
325     # un canale fissato
326
327     # Inizializzo pozyx seriale
328     serial_tag = SerialTag(pozyx, anchor_ids, remote_tag_ids, auto_cal, R_mis,
329                             ranging_protocol, N_acq, use_remote_tag, myUWBSettings)
330
331     serial_tag.setup()
332
333     # Open file
334     # Controlla l'apertura del file corretto in modo da non sovrascrivere i dati
335     # precedenti
336     k = 0
337     if ranging_protocol == POZYX_RANGE_PROTOCOL_PRECISION:
338         protocol_str = "/PROTOCOL_PRECISION/"

```

```

336     elif ranging_protocol == POZYX_RANGE_PROTOCOL_FAST:
337         protocol_str = "/PROTOCOL_FAST/"
338     else:
339         raise Exception("Ranging protocol is not valid")
340         quit()
341     path = "./Canale" + str(channel) + "/PROTOCOL_PRECISION/"
342     while os.path.isfile(path + 'frequenza_'+str(k)+'.txt'):
343         k += 1
344
345     #Prima riga del file
346     name = path + "frequenza_"+str(k)+".txt"
347     print(name)
348     out_file = open(name, "w")
349
350     if out_file is None:
351         raise Exception("Couldn't open file, aborting...")
352         quit()
353
354     # Write anchor's configuration
355     i = 0
356     for anchor in serial_tag.anchors:
357         string = "Anchor" + str(i) + ", 0x%0.4x, %s \n" % (anchor.network_id, str(
358             anchor.pos))
359         out_file.write(string)
360         i += 1
361     out_file.write('\n')
362     out_file.write('\n')
363
364     # Write UWB Settings
365     write_uwb_parameters(myUWBSettings, ranging_protocol, out_file)
366
367
368     # Acquisizioni
369     N = 500          #Numero di acquisizioni
370     out_file.write("Misure in [mm] e [ms]\n\n")
371     out_file.write("X\tY\tZ\tDt\tAO\tA1\tA2\tA3\t\n")
372
373     try:
374         for i in list(range(N)):
375             print(i)
376             serial_tag.loop(out_file)
377
378             fail_percentage = 100 * serial_tag.pos_error / N
379             out_file.write("Percentuale di fallimenti nel positioning: " + str(
380                 fail_percentage)+ "%")
381             out_file.close()
382     except KeyboardInterrupt:
383         print("interrupted!")
384         quit()
385
386
387
388
389     channel = 3
390     bitrates = [PozyxConstants.UWB_BITRATE_110_KBPS, PozyxConstants.
391                 UWB_BITRATE_850_KBPS, PozyxConstants.UWB_BITRATE_6810_KBPS]
392     protocols = [POZYX_RANGE_PROTOCOL_PRECISION, POZYX_RANGE_PROTOCOL_FAST]
393     prfs = [PozyxConstants.UWB_PRF_64_MHZ]
394     plens = [PozyxConstants.UWB_PLEN_64,
395               PozyxConstants.UWB_PLEN_128,
396               PozyxConstants.UWB_PLEN_256,
397               PozyxConstants.UWB_PLEN_512,
398               PozyxConstants.UWB_PLEN_1024,
399               PozyxConstants.UWB_PLEN_1536,
400               PozyxConstants.UWB_PLEN_2048,
401               PozyxConstants.UWB_PLEN_4096]
402
403     total = 48
404     iteration = 0
405
406     for protocol in protocols:

```

```

406     for bitrate in bitrates:
407         for prf in prfs:
408             for plen in plens:
409                 myUWBSettings = UWBSettings(channel, bitrate, prf, plen, gain)
410
411                 # Inizializzo pozyx seriale
412                 serial_tag = SerialTag(pozyx, anchor_ids, remote_tag_ids,
413                                         auto_cal, R_mis, ranging_protocol, N_acq, use_remote_tag,
414                                         myUWBSettings)
415
416                 serial_tag.setup()
417
418                 # Open file
419                 # Controlla l'apertura del file corretto in modo da non
420                 # sovrascrivere i dati precedenti
421                 k = 0
422
423                 if protocol == POZYX_RANGE_PROTOCOL_PRECISION:
424                     protocol_str = "/PROTOCOL_PRECISION/"
425                 elif protocol == POZYX_RANGE_PROTOCOL_FAST:
426                     protocol_str = "/PROTOCOL_FAST/"
427                 else:
428                     raise Exception("Ranging protocol is not valid")
429                     quit()
430
431                 path = "./Canale" + str(channel) + protocol_str
432                 while os.path.isfile(path + 'frequenza_+' + str(k) + '.txt'):
433                     k += 1
434
435                 # Prima riga del file
436                 name = path + "frequenza_+" + str(k) + ".txt"
437                 print(name)
438                 out_file = open(name, "w")
439
440                 if out_file is None:
441                     raise Exception("Couldn't open file, aborting...")
442                     quit()
443
444                 # Write anchor's configuration
445                 i = 0
446                 for anchor in serial_tag.anchors:
447                     string = "Anchor" + str(i) + ", 0x%0.4x, %s \n" % (anchor.
448                                         network_id, str(anchor.pos))
449                     out_file.write(string)
450                     i += 1
451
452                 out_file.write('\n')
453                 out_file.write('\n')
454
455                 # Write UWB Settings
456                 write_uwb_parameters(myUWBSettings, ranging_protocol, out_file)
457
458
459                 # Acquisizioni
460                 N = 500           # Numero di acquisizioni
461                 out_file.write("Misure in [mm] e [ms]\n\n")
462
463                 out_file.write("X\|tY\|t\|tZ\|tDt\|tA0\|t\|tA1\|t\|tA2\|t\|tA3\|t\|n")
464
465                 print()
466                 print("-----")
467                 print("iteration " + str(iteration + 1) + " of 48")
468                 print()
469                 print("-----")
470
471                 try:
472                     for i in list(range(N)):
473                         print(i)
474                         serial_tag.loop(out_file)
475
476                         fail_percentage = 100 * serial_tag.pos_error / N
477                         out_file.write("Percentuale di fallimenti nel positioning:
478                                         " + str(fail_percentage) + "%")
479                         out_file.close()
480                 except KeyboardInterrupt:
481                     print("interrupted!")
482                     quit()

```

```

474     iteration += 1
475     """

```

## G Codice propagazione dell'errore

```

1 function [ Dx2, Dy2, Dx3, Dy3, Dz3] = prop_err( R01, R02, R03, R12, R13, R23, D01,
2     ...
3 %           D02, D03, D12, D13, D23 )
4 % function [ Dx2, Dy2, Dx3, Dy3, Dz3] = prop_err( R01, R02, R03, R12, R13, R23, D01
5 %     , ...
6 %           D02, D03, D12, D13, D23 )
7 % Questa funzione effettua una propagazione degli errori e sulla base
8 % delle incertezze nelle misurazioni al metro laser, fornisce
9 % l'incertezza delle coordinate delle singole ancore.
10 % Dy = sum(abs(dy/dx)*Dx). Dove dy/dx è la derivata di y rispetto alla
11 % variabile x, Dx è l'incertezza su x, e sum indica la sommatoria su tutte
12 % le variabili indipendenti di cui la funzione.
13
14 R = [R01, R02, R03, R12, R13, R23]; % distanze fra le coppie di ancore
15 D = [D01, D02, D03, D12, D13, D23]; % incertezze fra le coppie di ancore
16
17 for i = 1:6
18     if ((R(i)<0)|| (D(i)<0))
19         error('Le distanze e le incertezze devono essere positive.')
20     end
21 end
22
23 % Definizioni delle variabili simboliche
24 syms r01 r02 r03 r12 r13 r23
25 y2 = (r01^2 - r12^2 + r02^2)/(2*r01);
26 x2 = sqrt(r02^2 - y2^2);
27 y3 = (r03^2 - r13^2 + r01^2)/(2*r01);
28 x3 = (x2^2 - 2*y3*x2 + y2^2 - r23^2 + r03^2)/(2*x2);
29 z3 = sqrt(r03^2 - x3^2 - y3^2);
30
31 Dy2_sym = simplify(jacobian(y2,[r01,r02,r03,r12,r13,r23]));
32 Dy2 = abs(subs(Dy2_sym, [r01,r02,r03,r12,r13,r23], R))*D';
33
34 Dx2_sym = simplify(jacobian(x2,[r01,r02,r03,r12,r13,r23]));
35 Dx2 = abs(subs(Dx2_sym, [r01,r02,r03,r12,r13,r23], R))*D';
36
37 Dx3_sym = simplify(jacobian(x3,[r01,r02,r03,r12,r13,r23]));
38 Dx3 = abs(subs(Dx3_sym, [r01,r02,r03,r12,r13,r23], R))*D';
39
40 Dy3_sym = simplify(jacobian(y3,[r01,r02,r03,r12,r13,r23]));
41 Dy3 = abs(subs(Dy3_sym, [r01,r02,r03,r12,r13,r23], R))*D';
42
43 Dz3_sym = simplify(jacobian(z3,[r01,r02,r03,r12,r13,r23]));
44 Dz3 = abs(subs(Dz3_sym, [r01,r02,r03,r12,r13,r23], R))*D';
45
46 Dx3 = double(Dx3);
47 Dy3 = double(Dy3);
48 Dz3 = double(Dz3);
49 Dx2 = double(Dx2);
50 Dy2 = double(Dy2);
51
52
53 end

```