UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Tesi di Laurea in Ingegneria Meccatronica

" Tecniche di localizzazione indoor per droni basate su fiducial markers "

Relatore:
Dr. Stefano Michieletto

Correlatrice:
Dr. Giulia Michieletto

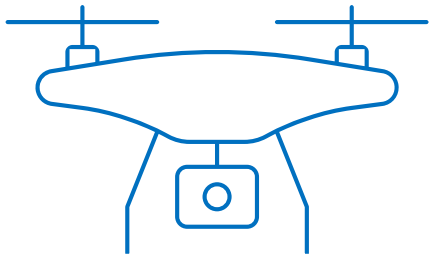
Laureandi:
Kumar Saurav
Matricola 1226950

Boghian Bogdan Dragos
Matricola 1224085

Anno accademico: 2021/2022

UNIVERSITÀ DEGLI STUDI DI PADOVA

Panoramica



I droni stanno rapidamente diventando la tecnologia di punta nel mondo della robotica mobile

In ambienti Indoor il GPS non risulta accurato

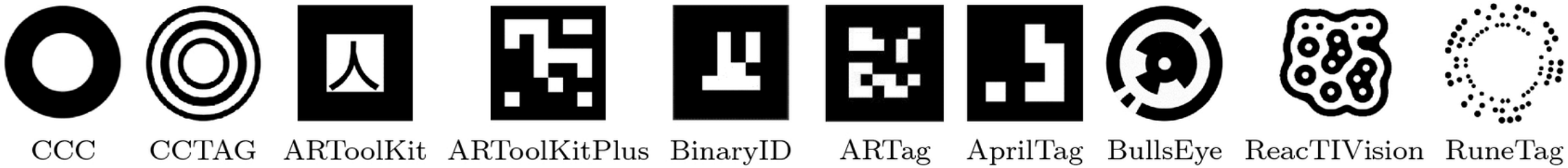
Sviluppo di metodi di localizzazione alternativi

Impiego dei Fiducial Markers

Cosa sono i Fiducial Markers?

- Un *Fiducial Marker* è un oggetto visibile posizionato come punto di riferimento per telecamere di rilevamento della posizione, smartphone o display
- Aiuta a tradurre i riferimenti spaziali tra il mondo concreto e quello aumentato, fungendo da ancoraggio di scala, posizione e orientamento, consentendo la sovrapposizione virtuale in tempo reale
- Permettono applicazioni nel campo della realtà aumentata, robotica e droni
- Soluzione low-cost, robusta, semplice e rapida implementabile con una semplice stampante bianco/nero

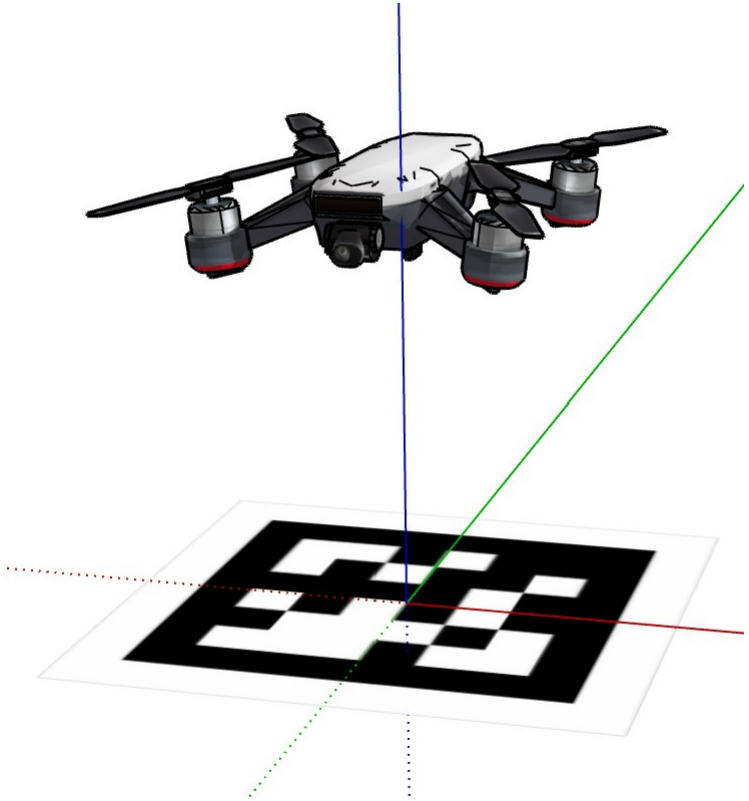
Esempi di Fiducial Markers :



AprilTag

AprilTag fa parte della famiglia dei Fiducial Markers, esso offre ottime prestazioni nel rilevamento rapido e a bassa latenza della stima della posizione 6D (posizione 3D e orientamento 3D).

Il software di rilevamento AprilTag nota la dimensione del Tag ne calcola la posizione 3D precisa, l'orientamento e l'ID del Tag rispetto alla fotocamera.



Il rilevamento dei Tag avviene in due fasi:

i. Ricerca del candidato
ii. Identificazione interna dei Markers

Famiglia TagStandard 41h12

La famiglia, degli AprilTags, più appropriata per il nostro scopo è la *TagStandard 41h12* dove:

- "41" indica la quantità di bit di dati
- "12" indica la distanza minima di Hamming

FAMIGLIA	n_{max}° AprilTag
TagStandard36h11	587
TagStandard41h12	2115
TagStandard52h13	48714
TagCircle21h7	38
TagCircle49h12	65698
TagCustom48h12	42211

Questi tipi di Tag offrono varie caratteristiche:

- buon comportamento alle variazioni di luce
- basso tasso di falsi positivi
- ottima velocità di decodifica
- adeguato numero di Markers per generare una mappa 5m x 7m ($\simeq$1700 AprilTags)

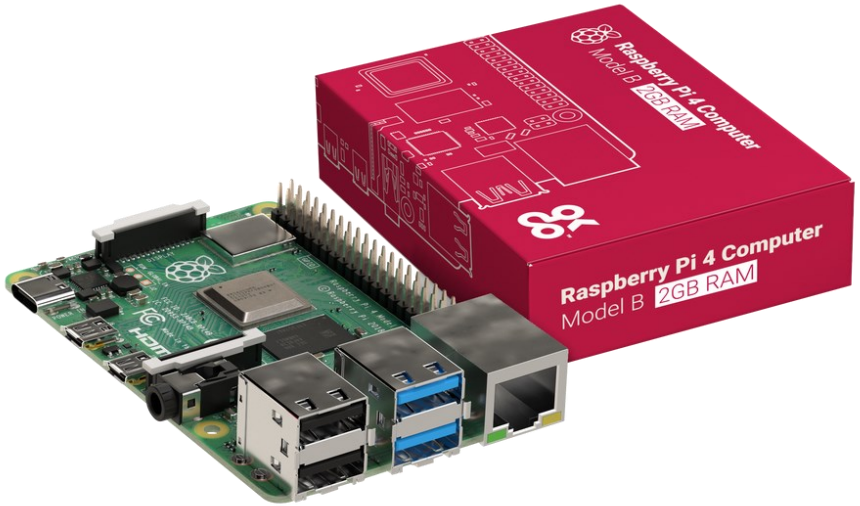
UNIVERSITÀ DEGLI STUDI DI PADOVA

Obiettivo e hardware

L'obiettivo principale della nostra ricerca è stato quello di sostituire la scheda originaria *Rasberry PI 4B* utilizzando un'altra scheda alternativa in cerca di risultati migliori in confronto a quelli ottenuti fin'ora.

L'hardware è composto da :

- Scheda *Nvidia Jetson Nano*
- Telecamera *Intel Real Sense Depth Camera D435*
- Alimentazione via PowerBank o via PC



Scheda Rasberry PI 4B

Nvidia Jetson Nano

La *Nvidia Jetson Nano* è un piccolo ma potente computer che permette di eseguire più reti neurali in parallelo per applicazioni come classificazione di immagini, rilevamento di oggetti, segmentazione ed elaborazione del linguaggio.

Questa scheda, a differenza della *Raspberry* (scheda attualmente utilizzata per questo progetto) è dotata di una scheda grafica (Nvidia Maxwell GPU).

Quest'ultima lavorando assieme alla CPU dovrebbe garantire minor tempo per l'elaborazione dei dati offrendo, perciò, una miglior precisione di localizzazione.

Scheda Nvidia Jetson Nano

Specifiche tecniche scheda

Jetson Nano monta una CPU Quad-core ARM A57 a 64 bit, una GPU NVIDIA Maxwell da 128-core e una memoria LPDDR4 da 4 GB, in grado di sopportare fino a 472 GFLOPS, (Giga FLoating point Operations Per Second) quindi più di 472 miliardi di operazioni a virgola mobile al secondo.

Queste caratteristiche la rendono un'ottimo candidato per ottenere prestazioni migliori in confronto alla versione attuale del progetto.

GPU	Architettura NVIDIA Maxwell™ con 128 core NVIDIA CUDA®
CPU	Processore Quad-core ARM® Cortex®-A57 MPCore
Memoria	LPDDR4 4 GB 64-bit
Spazio di archiviazione	Unità flash eMMC 5.1 da 16 GB
Codifica video	4K a 30 (H.264/H.265)
Decodifica video	4K a 60 (H.264/H.265)

Intel Real Sens Depth Camera

La telecamera di profondità D435 offre una profondità di qualità per diverse applicazioni grazie a una tecnologia di visione stereo assistita da un proiettore a infrarossi.

Il suo ampio campo visivo è perfetto per applicazioni come la robotica o la realtà aumentata e virtuale, dove vedere quanta più scena possibile è di vitale importanza.

Un grande vantaggio è la possibilità di utilizzare questa camera con i più comuni linguaggi di programmazione.

La camera genera due flussi di dati:

- l'immagine DEPTH
- l'immagine RGB

Specifiche tecniche Camera

Operating Range	0.3 m - 10 m
Left/Right Imagers Type	Wide
Depth FOV HD (16:9) (degrees)	H:87 / V:58
Depth FOV VGA (4:3) (degrees)	H:75 / V:62
IR Projector	Wide
IR Projector FOV	H:90 / V:63
Color Sensor	OV2740
Color Camera FOV	H:69 /V:42
IMU	NA
Stereo Depth resolution	up to 1280x720
Stereo Depth Frame Rate (FPS)	up to 90
RGB resolution	up to 1920x1080
RGB Frame Rate (FPS)	up to 60

Il suo ampio campo visivo è perfetto per applicazioni come la robotica o la realtà aumentata e virtuale, dove vedere quanta più scena possibile è di vitale importanza.

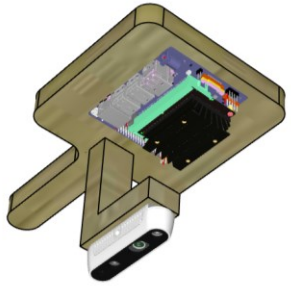
Tale campo di visione è di 42° in verticale e 69° in orizzontale, questo implica che al variare dell'altezza aumenta la dimensione della finestra di acquisizione.

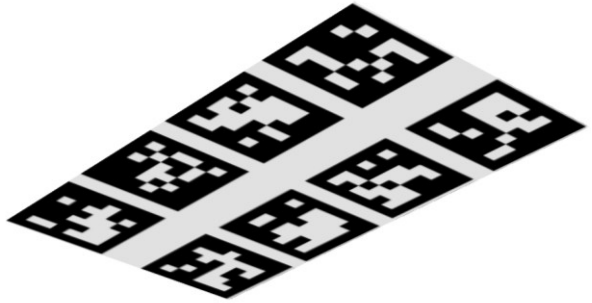
Primo prototipo

Il primo prototipo è stato una specie di paletta in legno pressato su cui avevamo intenzione di fissare la scheda e la telecamera.

I movimento nello spazio erano manuali, ovvero spostandosi semplicemente con la paletta in mano ottenendo, però, una precisione insufficiente.

Fin da subito abbiamo capito la necessità di realizzare un sistema molto più preciso.

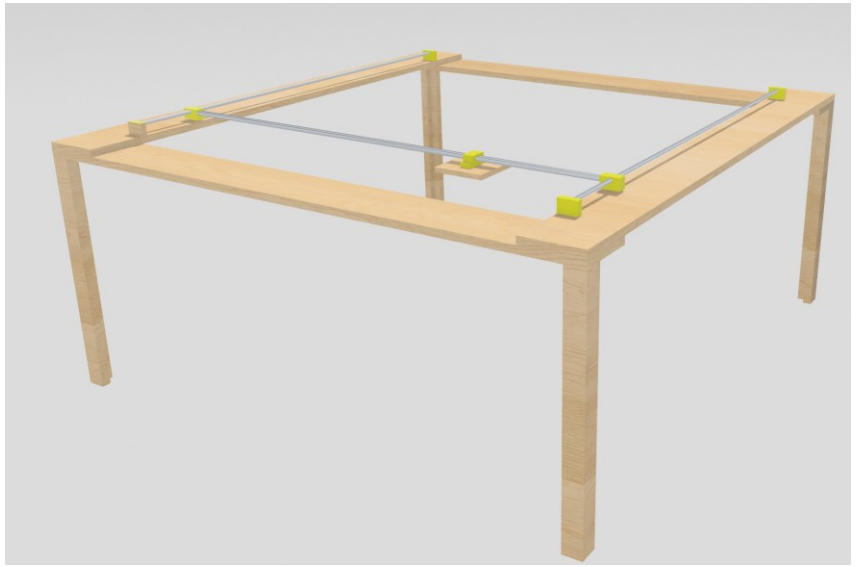


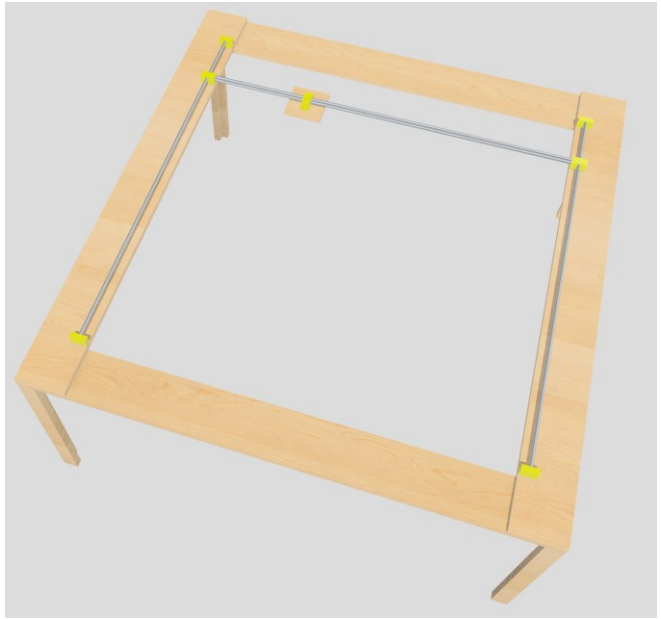


Prototipo finale

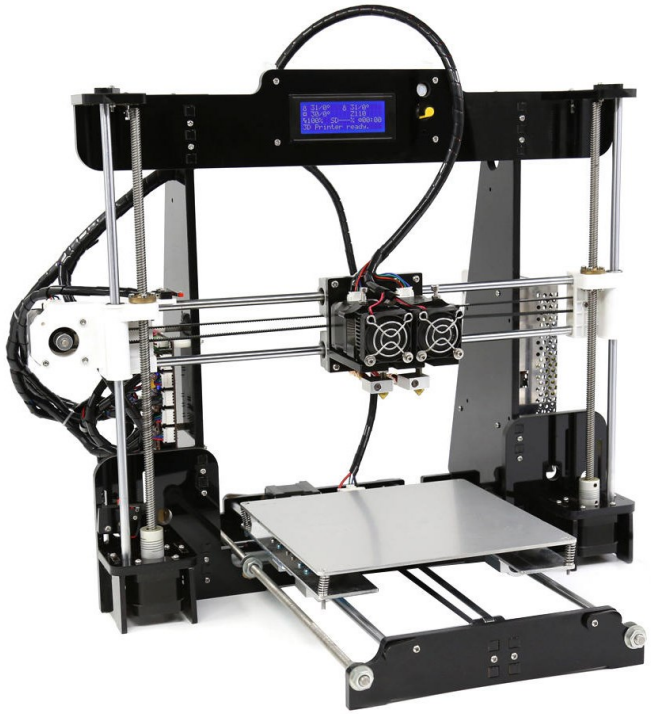
Per adottare un approccio più ingegneristico siamo passati a un sistema di movimento su binari, il quale permette movimenti sui tre assi lineari e grazie a un snodo sferico anche sulle tre rotazioni intorno agli assi con un ordine di precisione migliore rispetto al primo prototipo.

In questo modo siamo riusciti ad ottenere movimenti con una precisione di circa 2 mm.





Realizzazione prototipo



Stampante 3D Anet A8

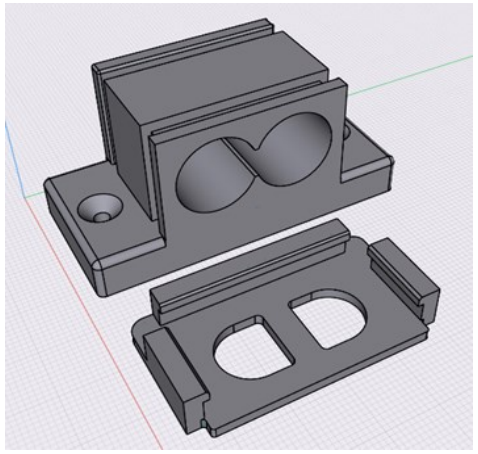
Filamento PLA 1,75 mm

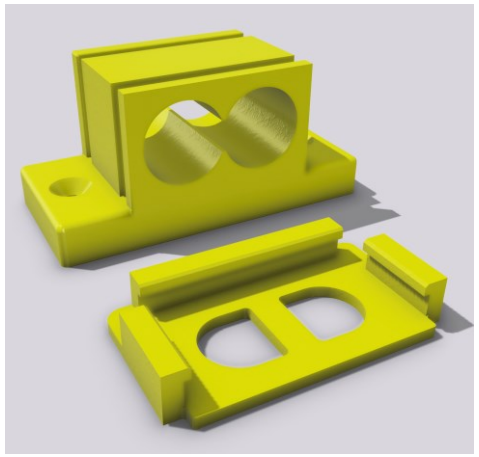
Tondi pieni di alluminio D= 8mm

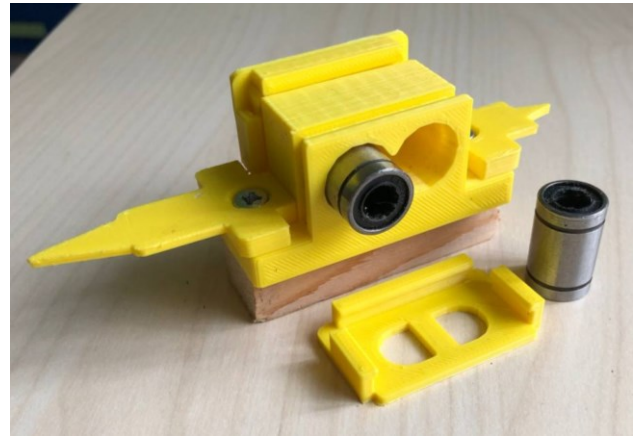
Cuscinetti lineari a sfere d=8 mm

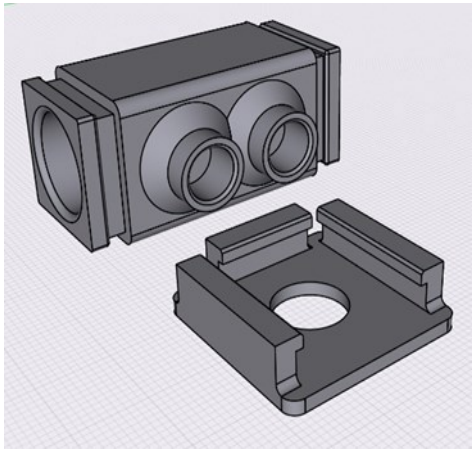
Listelli di legno multistrato

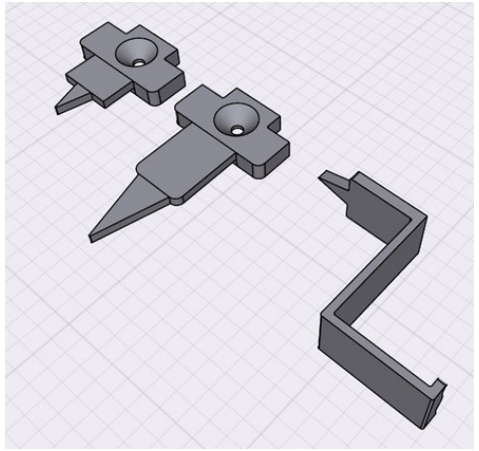
Progettazione e stampa 3D

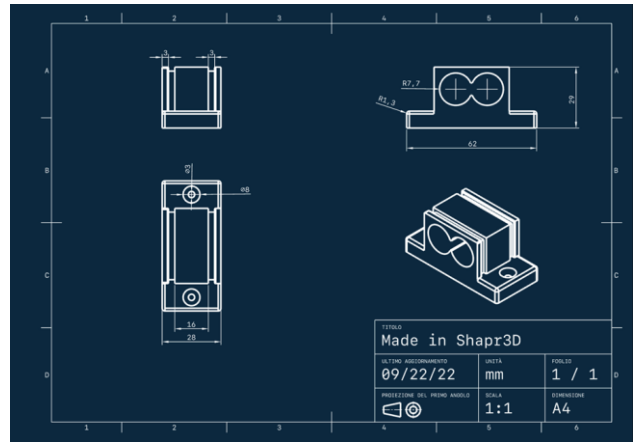








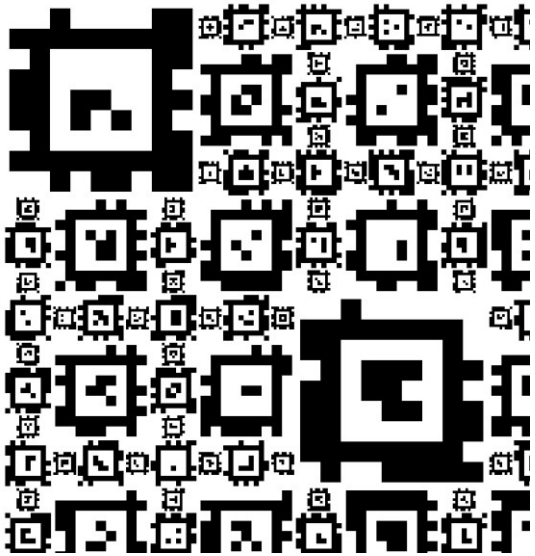




Mappa

La mappa utilizzata da noi è una porzione di quella creata da precedenti tesisti, di dimensioni 140x140 cm.

Essa è formata da AprilTags di 4 taglie identificate dalle lettere S, M, L, XL con le corrispondenti dimensioni: 5.75 cm, 11.5 cm, 23 cm e 46 cm. La presenza di questa varietà di dimensioni garantisce una rilevazione degli Markers a diverse altezze.



Altezza [m]	Finestra di acquisizione [m]	AprilTag "41h12"
0.15	0.1 x 0.17	S
0.3	0.2 x 0.34	S,M
0.6	0.4 x 0.68	M,L
1.0	0.6 x 1.0	L
1.5	1.2 x 2.0	XL
2.0	2.4 x 4.0	XL

Implementazione software

Per implementare questo progetto, abbiamo utilizzato il sistema operativo Ubuntu 20.04 su cui abbiamo installato ROS2 (Robotic Operation System) versione Foxy.

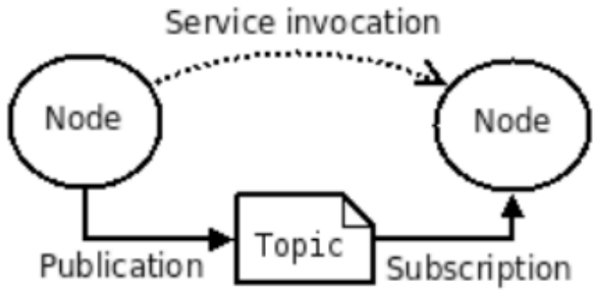
Questa fase si è suddivisa in due step principali:

- Creazione del workspace "dev_ws" in ROS2 dedicato esclusivamente al nostro progetto
- Installazione dei vari pacchetti necessari, i quali possono contenere nodi, definizioni di messaggi o codici

Pacchetti utilizzati

Nel nostro workspace creato precedentemente troviamo i seguenti pacchetti:

- **realsense2_camera** che contiene il nodo camera
- **apriltag_ros** che contiene il nodo di riconoscimento degli apriltag
- **apriltag_to_visual_odometry** che contiene il nodo per la localizzazione del drone
- px4_msgs che contiene le definizioni dei messaggi del client PX4
- apriltag_viz che è necessario al funzionamento di apriltag_ros
- apriltag_msgs che contiene le definizioni dei messaggi specifici per gli AprilTag



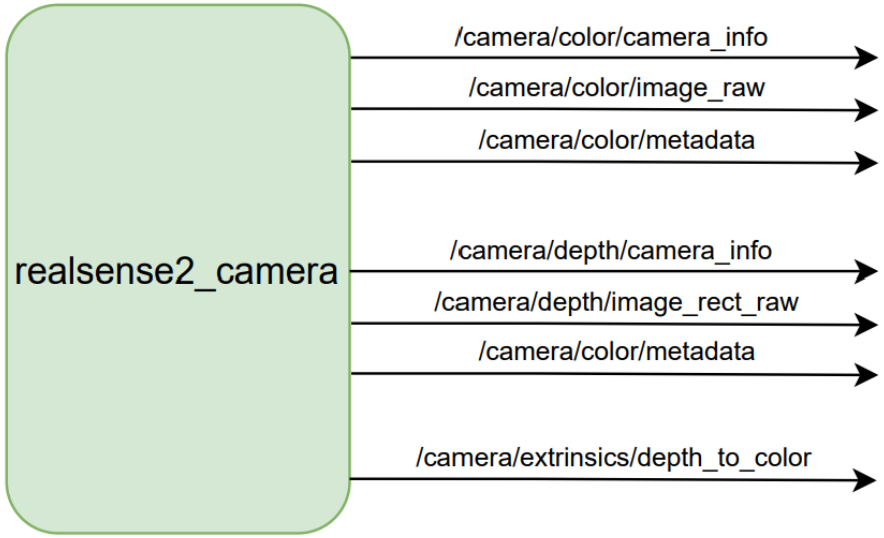
Nodo realsense2_camera

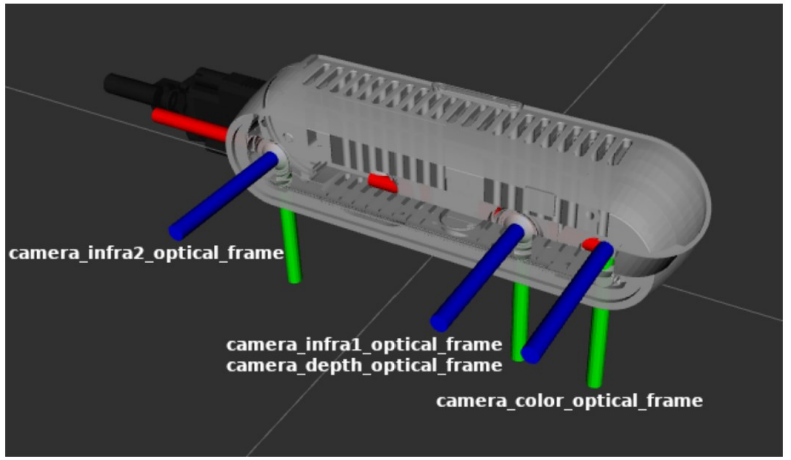
La stereo-camera utilizzata genera due flussi video: depth e rgb.

E’ possibile attivare entrambi i flussi lanciando il nodo attraverso il comando:

```
ros2 launch realsense2_camera rs_launch.py
```

Nel momento in cui inviamo il comando vengono pubblicati i seguenti Topic:





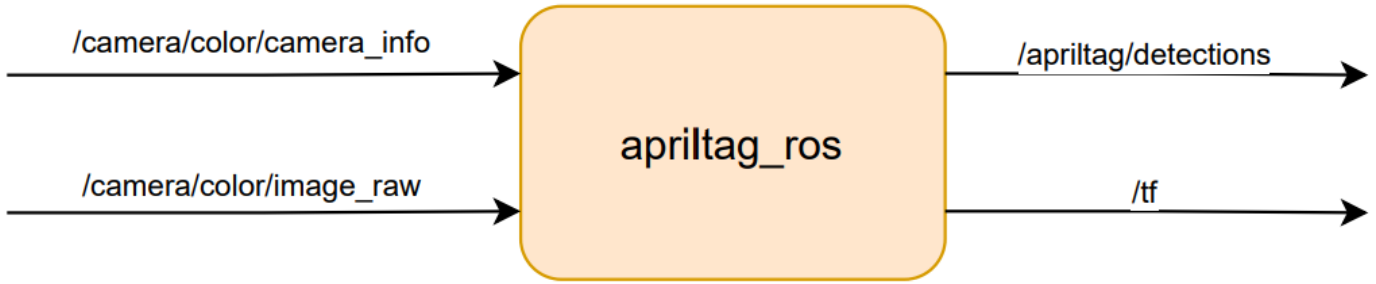
Nodo apriltag_ros

Il nodo apriltag_ros per ROS2 utilizza la libreria AprilTag e alcuni pacchetti aggiuntivi tra cui apriltag_viz e apriltag_msgs per rilevare gli AprilTag nelle immagini e pubblicare la loro posizione (/tf).

All'avvio di questo nodo parte il riconoscimento degli AprilTags della famiglia 41h12.

Il nodo può essere lanciato attraverso il seguente comando:

```
ros2 launch apriltag_ros tag_41h12_all.launch.py
```

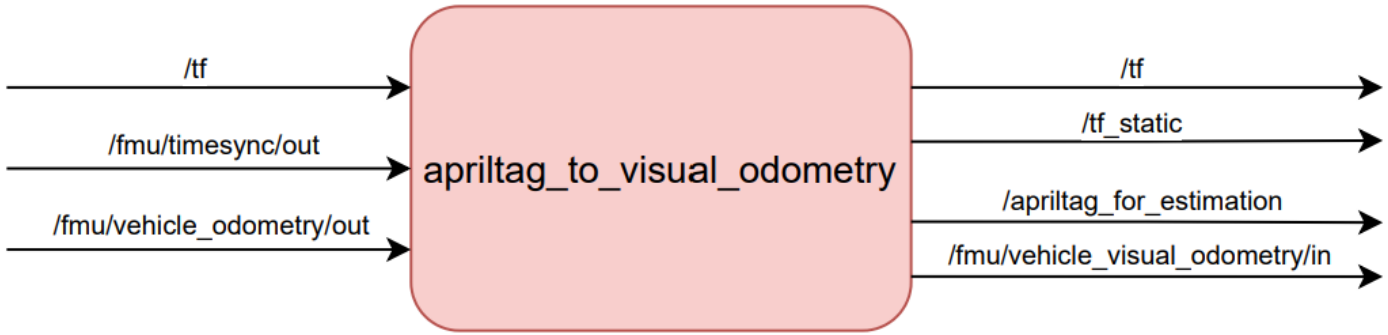


Nodo apriltag_to_visual_odometry

Tale nodo è in grado si generare istante per istante una stima di odometria del drone a partire dalle posizioni degli AprilTags riconosciuti dal nodo apriltag_ros.

Il nodo può essere lanciato attraverso il seguente comando:

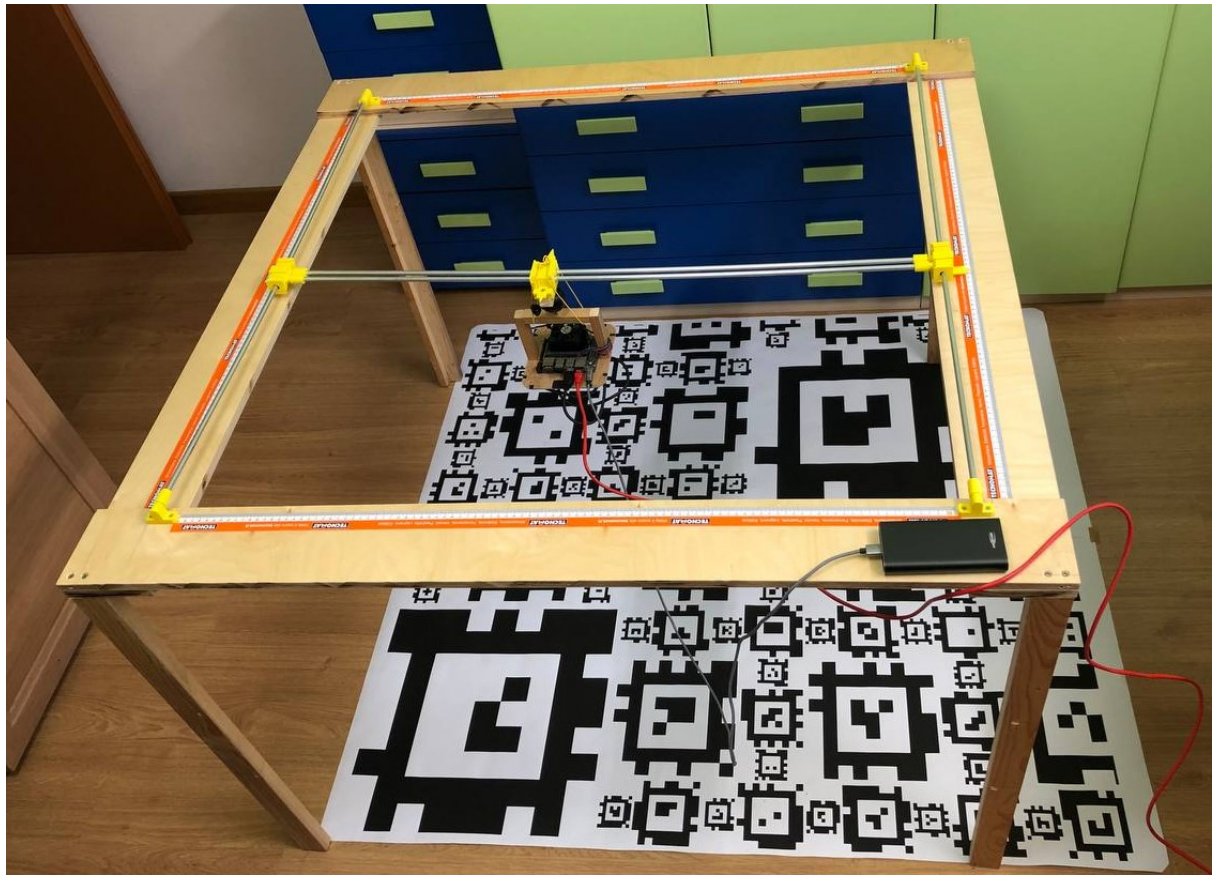
```
ros2 launch apriltag_to_visual_odometry
    apriltag_to_visual_odometry.launch.py
```



Risultato effettivo

Il prototipo finale per i test è così composto dalla porzione della mappa stampata e l'hardware montato sulla parte mobile della struttura.

Il tappetto degli Apriltags è costituito dall'unione di due fogli 70x140 cm.

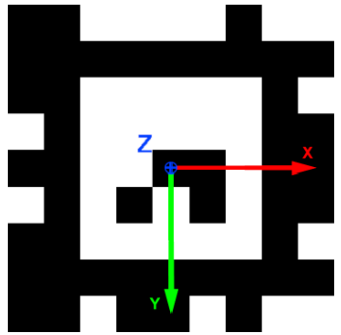


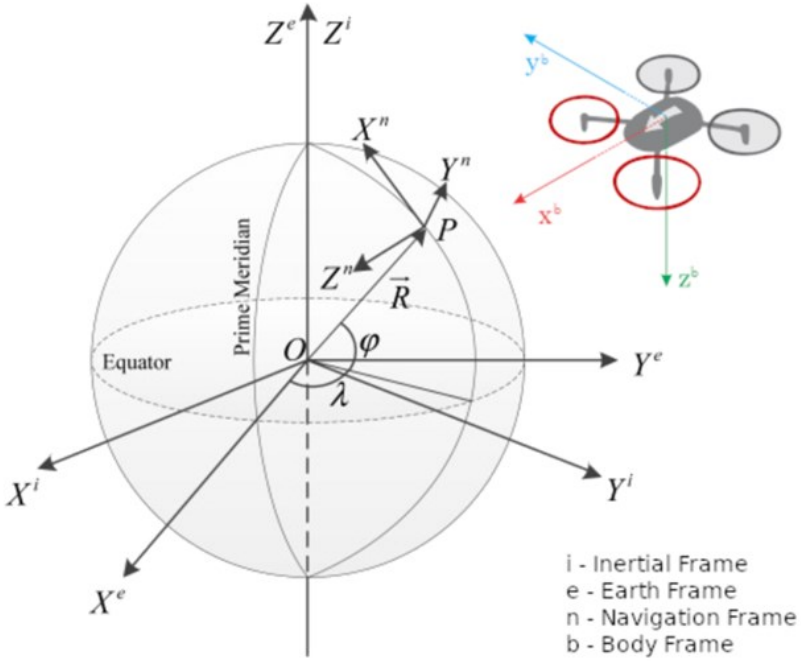
Sistemi di riferimento

I sistemi di riferimento utili per localizzare un drone sono:

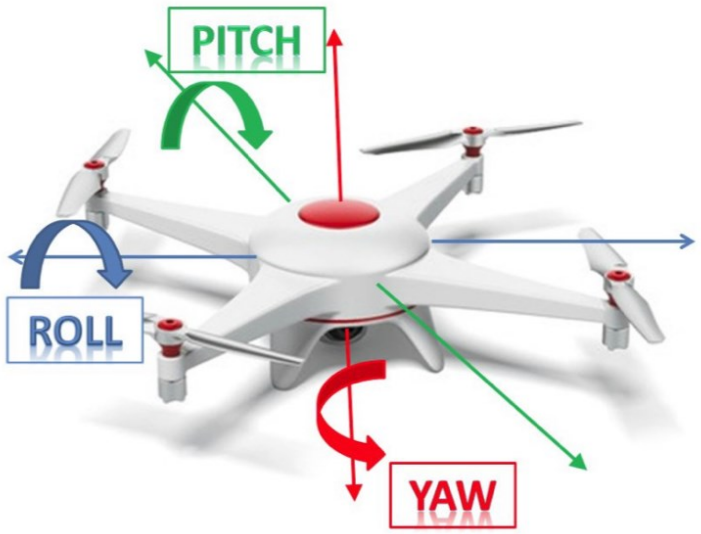
- Sistema di riferimento inerziale (i)
- Sistema di riferimento terrestre (e): sistema di localizzazione utilizzato dal GPS
- Sistema di riferimento di navigazione (n): sistema utilizzato per identificare il sistema di riferimento del drone
- Sistema di riferimento del drone (b): tale sistema viene chiamato body-frame, è solidale col drone, la cui origine coincide con il centro di massa (COM)

(n)





L'orientazione del sistema



Per individuare l'orientazione in tre dimensioni del nostro prototipo, come avviene anche con i velivoli, vengono usati tre angoli:

- roll (rollio)
- pitch (beccheggio)
- yaw (imbarcata)

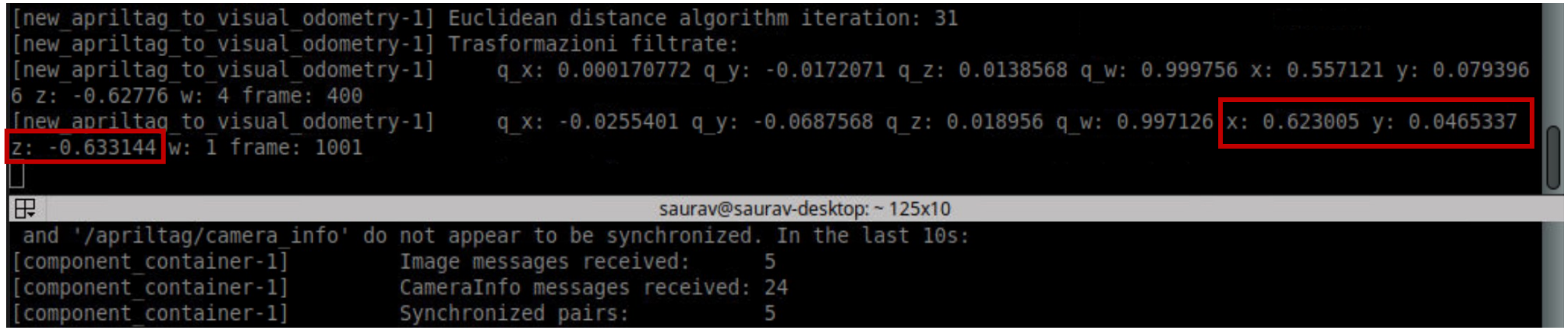
Tali angoli rappresentano le rotazioni intorno agli assi x, y e z rispettivamente.

Fase di test

Lo scopo del test è di andare a verificare, posizionando il sistema in vari punti noti, la precisione degli output calcolati dalla scheda verificando così l'errore presente.

Prima misura

Spostando il sistema nella posizione x=60 cm, y=5cm e z=63cm l'output generato sul terminale è stato il seguente:



UNIVERSITÀ DEGLI STUDI DI PADOVA

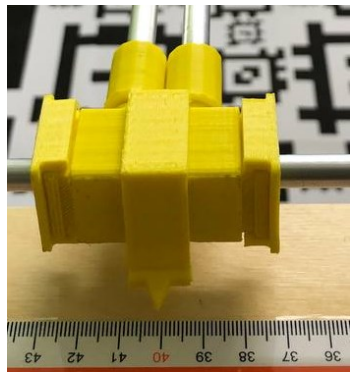
Test assi lineari x, y, z

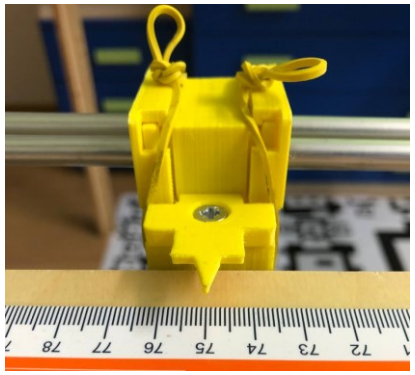
Altre prove: x=40cm, y=75cm e z=63 cm

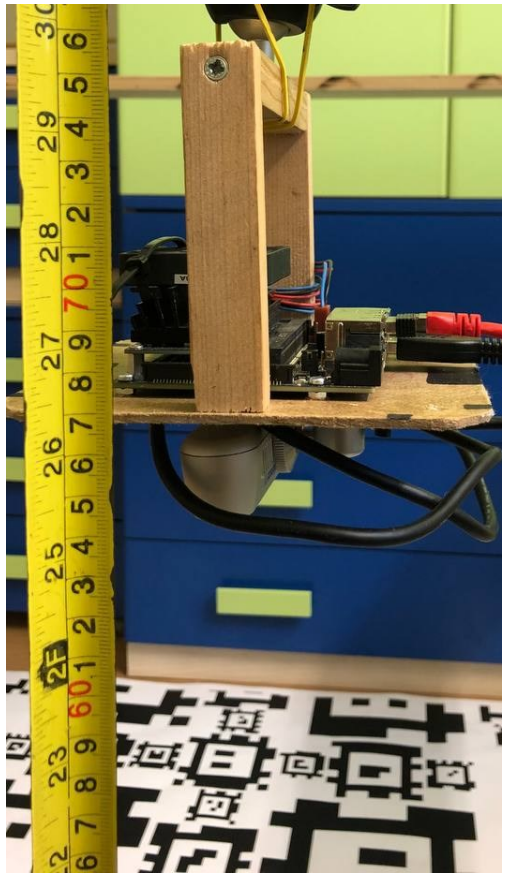
```
[new_apriltag_to_visual_odometry-1]     q_x: 0.0360983 q_y: 0.00674368 q_z: 0.720661 q_w: 0.692314 x: 0.38632 y: 0.792685 z:
.626811 w: 1 frame: 1421
[new_apriltag_to_visual_odometry-1] Euclq_x: 0.00456434 q_y: 0.0227928 q_z: 0.717931 q_w: 0.695726 x: 0.40136 y: 0.749919 z:
[new_apriltag_to_visual_odometry-1] Tras
[new_apriltag_to_visual_odometry-1]     q_x: 0.0156546 q_y: 7.21079e-05 q_z: 0.720157 q_w: 0.693635 x: 0.411414 y: 0.780527 z
-0.634519 w: 4 frame: 451
[new_apriltag_to_visual_odometry-1]     q_x: -0.0054159 q_y: 0.0050438 q_z: 0.72254 q_w: 0.69129 x: 0.427579 y: 0.757762 z: -
 -0.626559 w: 4 frame: 452
[new_apriltag_to_visual_odometry-1]     q_x: -0.0148657 q_y: 0.00301506 q_z: 0.719537 q_w: 0.694288 x: 0.436781 y: 0.750239 z
z: -0.630759 w: 4 frame: 437
```

saurav@saurav-desktop: ~ 125x10

```
 and '/apriltag/camera_info' do not appear to be synchronized. In the last 10s:
[component_container-1]         Image messages received:      7
[component_container-1]         CameraInfo messages received: 28
[component_container-1]         Synchronized pairs:           7
```



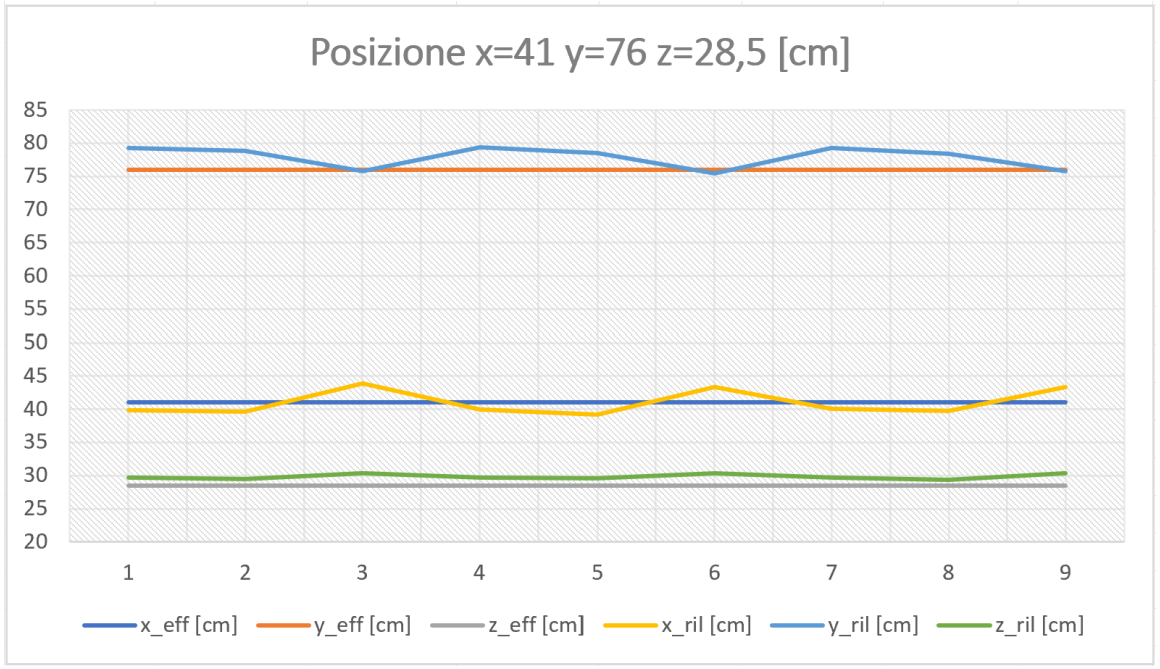




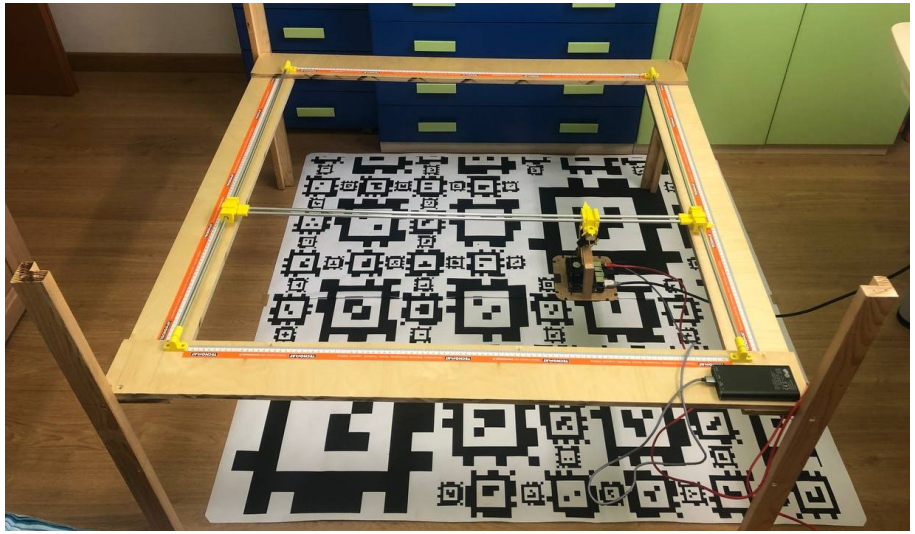
Analisi risultati

Prova effettuata a:

x=41cm, y=76cm e z=28,5cm



	x_eff [cm]	y_eff [cm]	z_eff [cm]	x_ril [cm]	y_ril [cm]	z_ril [cm]	errore x	errore y	errore z
	41	76	28,5	39,9	79,3	29,71	1,1	3,3	1,21
	41	76	28,5	39,65	78,82	29,45	1,35	2,82	0,95
	41	76	28,5	43,93	75,79	30,33	2,93	0,21	1,83
	41	76	28,5	39,98	79,36	29,73	1,02	3,36	1,23
	41	76	28,5	39,14	78,48	29,59	1,86	2,48	1,09
	41	76	28,5	43,32	75,47	30,37	2,32	0,53	1,87
	41	76	28,5	40,07	79,31	29,75	0,93	3,31	1,25
	41	76	28,5	39,7	78,37	29,39	1,3	2,37	0,89
	41	76	28,5	43,33	75,8	30,32	2,33	0,2	1,82
Max							**2,93**	**3,36**	**1,87**
Media				**41,002**	**77,856**	**29,849**			



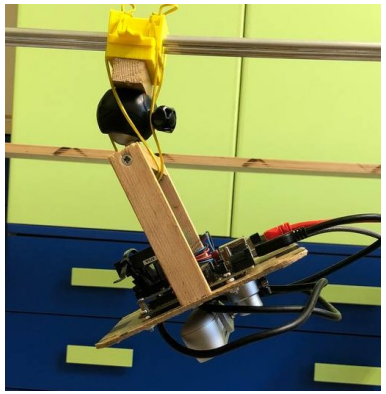
Test roll, pitch, yaw

```
[component_container-1]          Image messages received:      3
[component_container-1]          CameraInfo messages received: 29
[component_container-1]          Synchronized pairs:           3
```

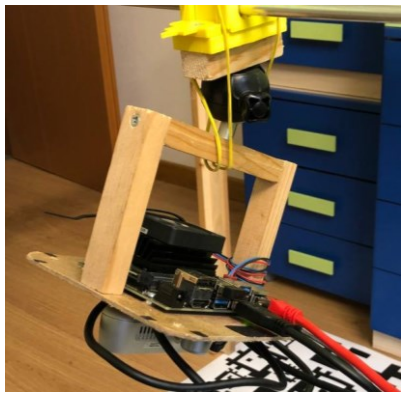
saurav@saurav-desktop: ~ 143x12

```
[new_apriltag_to_visual_odometry-1] ------------------------------------------------
[new_apriltag_to_visual_odometry-1] Trasformazioni ricevute:
[new_apriltag_to_visual_odometry-1]     q_x: 0.160343 q_y: -0.180519 q_z: 0.413488 q_w: 0.877913 x: 0.484549 y: 0.702132 z: -0.319287 w: 4 fram
e: 437
[new_apriltag_to_visual_odometry-1]     q_x: 0.164917 q_y: -0.0970136 q_z: 0.417418 q_w: 0.888343 x: 0.430252 y: 0.680866 z: -0.331177 w: 1 fra
me: 1393
[new_apriltag_to_visual_odometry-1] Trasformazioni filtrate:
[new_apriltag_to_visual_odometry-1]     q_x: 0.164917 q_y: -0.0970136 q_z: 0.417418 q_w: 0.888343 x: 0.430252 y: 0.680866 z: -0.331177 w: 1 fra
me: 1393
```

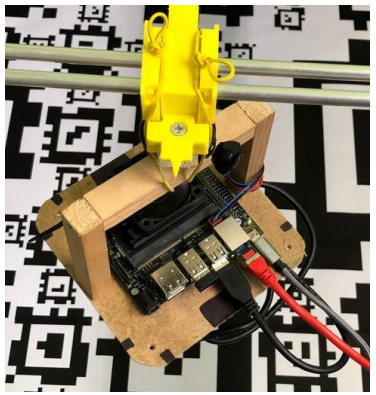
Roll (x)



Pitch (y)



Yaw (z)

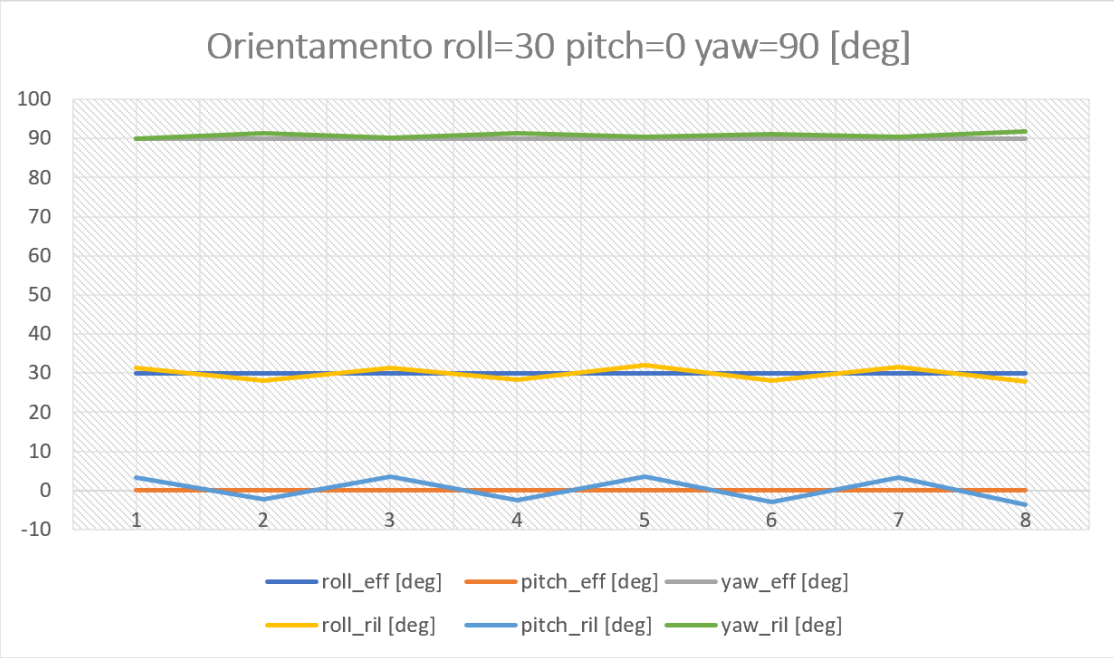


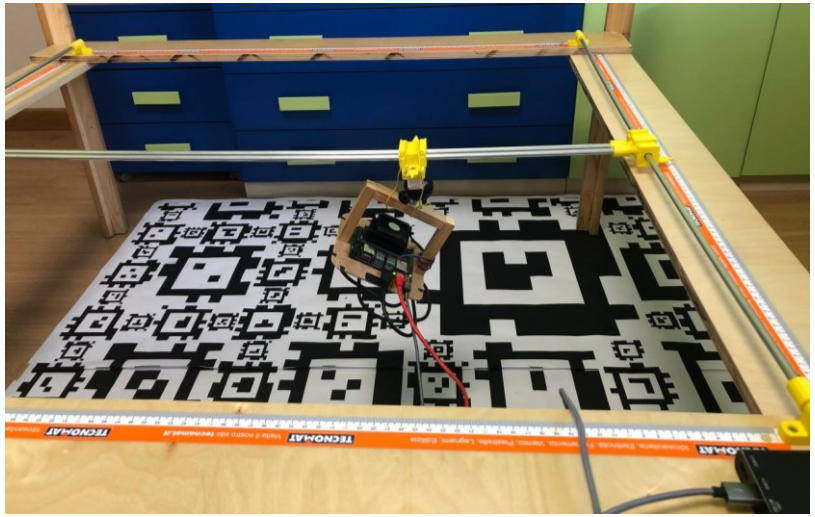
Test roll, pitch, yaw

Prova effettuata a:

roll=30°, pitch=0° e yaw=90°

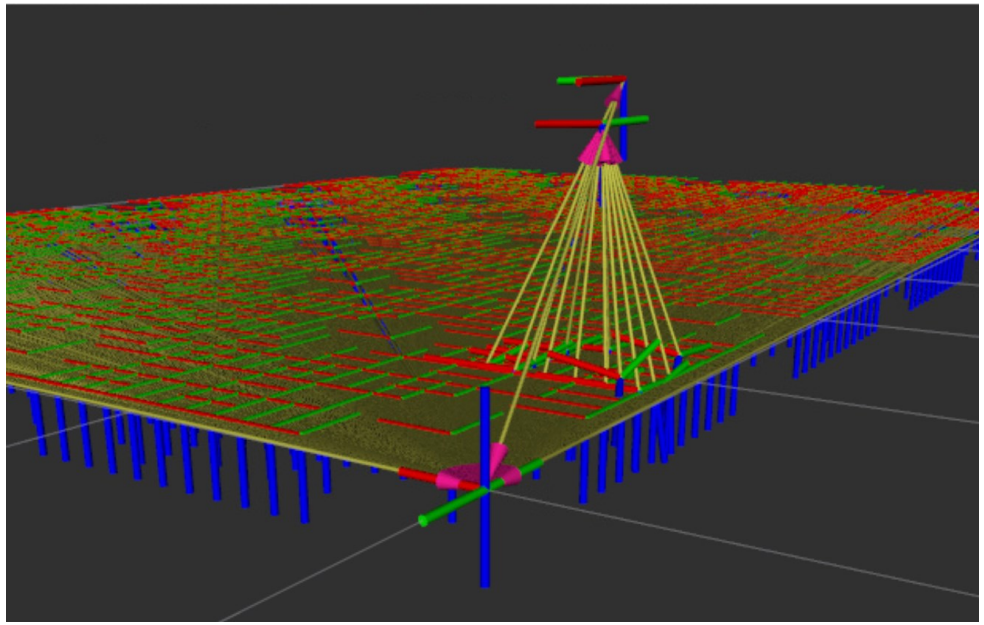
Angoli in deg [°]									
	roll_eff	pitch_eff	yaw_eff	roll_ril	pitch_ril	yaw_ril	errore roll	errore pitch	errore yaw
	30	0	90	31,44	3,36	89,9	1,44	3,36	0,1
	30	0	90	28,11	-2,26	91,38	1,89	2,26	1,38
	30	0	90	31,3	3,43	90,1	1,3	3,43	0,1
	30	0	90	28,27	-2,45	91,43	1,73	2,45	1,43
	30	0	90	32,06	3,46	90,4	2,06	3,46	0,4
	30	0	90	28,03	-2,86	91,05	1,97	2,86	1,05
	30	0	90	31,6	3,22	90,51	1,6	3,22	0,51
	30	0	90	27,77	-3,67	91,85	2,23	3,67	1,85
Max							**2,23**	**3,67**	**1,85**
Media				**29,823**	**0,279**	**90,828**			

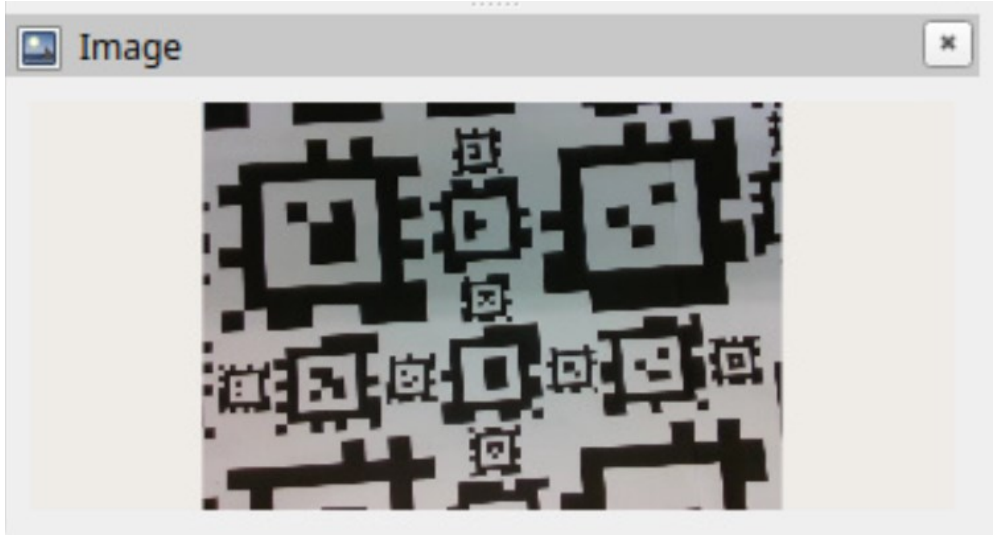




Interfaccia grafica RViz

Utilizzando il programma Rviz è possibile ottenere un'immagine visiva della posizione della telecamera calcolata in base agli AprilTags visualizzati.
Esso permette di visualizzare all'interno di uno spazio tridimensionale qualsiasi dato il software pubblichi attraverso i suoi topic .





Conclusioni

La tematica principale è stata quella di testare la scheda *Nvidia Jetson Nano* e capire quali risultati si possono ottenere rispetto a quelli ottenuti fino ad oggi con altro hardware.

Eseguendo diversi test si è giunti ad una precisione in termini di posizionamento nello spazio, nel caso migliore di 1-2 cm ma con picchi che variano dai 10 cm ai 30 cm. Risultati migliori sono stati ottenuti nel orientamento, dove l'errore massimo non supera i 4 gradi esadecimali.

Alcuni problemi che sono stati riscontrati sono:

- Ingombro del suolo dovuto alla presenza della mappa, che per massimizzare l'uso di tutto lo spazio necessita l'intera occupazione. Una futura soluzione può essere il posizionamento ,in punti strategici, degli Apriltags
- Nonostante le capacità della fotocamera in termini di risoluzione, è risultato conveniente impostare la risoluzione minima in modo da non rallentare eccessivamente l'elaborazione dei dati da parte della scheda

UNIVERSITÀ DEGLI STUDI DI PADOVA

Ringraziamenti

Ringraziamenti speciali vanno al nostro relatore Stefano Michieletto e correlatrice Giulia Michieletto i quali ci hanno fornito tutto il materiale necessario e le basi per intraprendere il progetto. Quest'ultimo ci ha coinvolto dall'inizio alla fine in quanto riguarda una tematica moderna e futuristica che ci ha ispirato a farne parte.

Un grande grazie a Massimiliano Bertoni che ci ha assistiti tempestivamente in diversi casi di difficoltà incontrati nel percorso.

Ringraziamo anche tutte le persone che ci sono state vicino e sostenuto durante questa esperienza fino a raggiungere questo grande traguardo.

Bibliografia

Ros2 Foxy Documentation : URL https://docs.ros.org/en/foxy/index.html

Apriltag : URL https://april.eecs.umich.edu/software/apriltag

Pacchetti installati da GitHub :

- *realsense2_camera :* URL https://github.com/IntelRealSense/realsense-ros.git
- *apriltag_ros :* URL https://github.com/christianrauch/apriltag_ros.git
- *apriltag_to_visual_odometry :* URL https://github.com/C-square-unipd/apriltag_to_visual_odometry.git
- *px4_msgs :* URL https://github.com/PX4/px4_msgs.git
- *apriltag_viz :* URL https://github.com/christianrauch/apriltag_viz.git
- *apriltag_msgs :* URL https://github.com/christianrauch/apriltag_msgs.git