

ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

Integrazione di un sistema di telecamere per videosorveglianza con i sistemi RTLs UWB

Elaborato in
Sistemi Embedded e Internet-of-Things

Relatore:
Prof. Alessandro Bevilacqua

Presentata da:
Elizabeta Budini

Correlatori:
Dott. Michele Dini
Dott. Ing. Nicolò Decarli

Anno Accademico 2018/2019

Abstract

Il progetto ha come scopo quello di far interagire un sistema di videosorveglianza (telecamere fisse e orientabili) con il sistema di localizzazione indoor UWB Sequitur. Sequitur permette di localizzare dei dispositivi (tags) tramite segnali radio a banda-ultralarga (UWB). Il sistema fornisce le coordinate (x, y, z) di ciascun dispositivo rispetto al sistema di riferimento scelto per il sistema. Il risultato del progetto è un software che, interfacciandosi al server Sequitur tramite REST API per ottenere la posizione dei dispositivi localizzati, sia in grado di rilevare quale telecamera in quel momento permetta di inquadrare un dispositivo tag predefinito (e quindi mostrare lo stream video). Inoltre, nel caso in cui la telecamera sia dotata di un sistema di orientamento PTZ, seguire il dispositivo localizzato inviando gli opportuni comandi di movimento (fornendo prima un livello di astrazione geometrico per il movimento, successivamente implementando il set di comandi per la telecamera in oggetto).

Indice

1	Introduzione	1
1.1	Descrizione del progetto	1
1.1.1	La nascita e l'obiettivo	1
1.2	Collaborazione con l'azienda	1
1.2.1	L'azienda	1
1.2.2	Il mio ruolo	2
2	UWB e RTLS	5
2.1	Banda-ultralarga (UWB)	5
2.1.1	Le caratteristiche	5
2.1.2	La storia	7
2.1.3	Gli ambiti applicativi	8
2.1.4	Indoor positioning	8
2.2	Sequitur RTLS	9
2.2.1	InGPS TWR	9
2.2.2	InGPS TDOA	10
2.2.3	InTracking TDOA	12
2.2.4	Coesistenza tra InTracking TDOA e InGPS TDOA	12
3	Analisi	14
3.1	Requisiti	14
3.1.1	Requisiti funzionali	14
3.1.2	Requisiti non funzionali	14
3.2	Analisi e modello del dominio	15
3.2.1	Descrizione dettagliata e UML del modello	15
3.2.2	Casi d'uso	16
4	Progettazione	17
4.1	Hardware e integrazione con il sistema esistente	17
4.2	Design architettonurale	18
4.3	Note di sviluppo e librerie	22
4.3.1	Testing automatizzato	22
4.3.2	Librerie di supporto	22
5	Implementazione	24
5.1	Interfaccia utente	24
5.2	Funzionamento del sistema	26
5.2.1	Camera Task	26
5.2.2	Tag Task	27

5.3	Gestione delle telecamere	29
5.4	Sviluppo comandi telecamera	29
5.4.1	Calcolo PAN	30
5.4.2	Calcolo TILT	32
5.4.3	Invio comando	34
5.5	Integrazione con Sequitur	34
5.5.1	Sequitur REST API	34
5.5.2	JSON Deserialization	35
5.6	Problemi incontrati e soluzioni	37
6	Conclusioni e sviluppi futuri	39
6.1	Sviluppi futuri	39
6.2	Considerazioni finali	40

Capitolo 1

Introduzione

1.1 Descrizione del progetto

1.1.1 La nascita e l'obiettivo

L'interesse verso questo progetto è nato qualche mese fa quando ho contattato l'azienda UNISET che da diversi anni collabora con il nostro corso di studi e ha contatti con i nostri professori. Parlando con il presidente dell'azienda, prof. Marco Tartagni, è emersa la necessità di progettare e implementare un'integrazione del sistema di indoor positioning già sviluppato da UNISET: Sequitur.

Il progetto ha come obiettivo quello di ampliare e integrare Sequitur con un sistema di videosorveglianza. Il software esistente sfrutta la tecnologia UWB per localizzare dei tag mobili all'interno di un ambiente prestabilito. In questo ambiente la localizzazione dei tag viene resa disponibile attraverso l'utilizzo di dispositivi fissi nell'ambiente, detti ancora, che comunicano con i tag. Il software che ho sviluppato sfrutta le informazioni elaborate da Sequitur e si interfaccia con le sue API per offrire un servizio integrato a delle videocamere di sorveglianza fisse e mobili. In questo modo è possibile monitorare non solo la posizione dei tag ma anche usufruire dello stream video proveniente dalla videocamera attiva nella stanza in cui si trova il tag. Qualora il tag si trovi in una zona sorvegliata da una camera mobile PTZ (PAN-TILT-ZOOM), il software, conoscendo la posizione in cui è installata la telecamera, calcola il punto verso cui questa deve ruotare per inquadrare il tag.

1.2 Collaborazione con l'azienda

1.2.1 L'azienda

Sulla scia del trend corrente in ambito di automazione per l'industria - si parla di Industria 4.0 - le fabbriche diventano sempre più digitalizzate e interconnesse. Grazie alle tecnologie IoT (Internet of Things) nascono le "smart factory", dove macchine, dispositivi, sensori e persone hanno la capacità di connettersi e comunicare tra loro. I sistemi cyber-physical, protagonisti della quarta rivoluzione industriale, monitorano i processi fisici, creando una rappresentazione virtuale del mondo fisico e permettendo una decentralizzazione delle decisioni. Sfruttan-

do l'IoT, questi sistemi comunicano e cooperano tra loro e con gli operatori umani in real-time.

In questa compagine tecnologica si inserisce l'azienda UNISET, spin-off dell'Università di Bologna¹, che si occupa di ricerca, sviluppo e produzione di sensori innovativi nel contesto dell'industria 4.0 e dell'IoT. In particolare opera nel settore di localizzazione indoor, outdoor e tracking, ma anche sistemi di sicurezza per il controllo degli accessi.

1.2.2 Il mio ruolo

L'azienda UNISET, per sviluppare e testare i propri progetti, ha a disposizione un edificio in cui il sistema di localizzazione Sequitur copre la parte evidenziata in blu nella figura 1.1. Il mio ruolo è stato quello di sviluppare e successivamen-



Figura 1.1: Mappa della zona coperta da Sequitur

te testare in loco un software che prevede l'installazione di due telecamere fisse, posizionate in uno degli uffici e nel corridoio, e una telecamera PTZ situata nel

¹<https://www.unisetcompany.com>

laboratorio del primo piano del suddetto edificio. Le zone coperte da videosorveglianza sono mostrate nella figura 1.2. Inoltre il software deve gestire, nel caso in cui gli oggetti da localizzare si trovino nella zona in cui è attiva la telecamera PTZ, l'invio dei comandi di movimento alla telecamera con lo scopo di seguire l'oggetto nella stanza.

Durante lo sviluppo dell'applicazione, l'azienda ha messo a mia disposizione una modalità di emulazione del server Sequitur, per poter analizzare e controllare il comportamento del software sviluppato senza la necessità di utilizzare l'hardware fisico di ancore e tag. In questa modalità di emulazione gli oggetti si muovono in modo automatico con la possibilità di impostare limiti in riferimento all'area in cui si trovano i device, il numero di device attivi, e altri parametri.

Le figure con cui ho interagito sono gli sviluppatori del software Sequitur, i quali inizialmente mi hanno fornito la documentazione necessaria a comprendere il sistema esistente. Successivamente abbiamo analizzato insieme i requisiti del nuovo software e nelle settimane seguenti, una volta chiarito come utilizzare le loro API e utilizzare il client Sequitur, ho potuto scegliere autonomamente la modalità di sviluppo dell'applicazione, portando avanti l'implementazione del software in autonomia. Non c'è stata una collaborazione diretta di più persone al codice, ma ho ricevuto numerosi consigli e soluzioni, anche riguardo alla scelta delle librerie più adatte alle necessità del sistema.

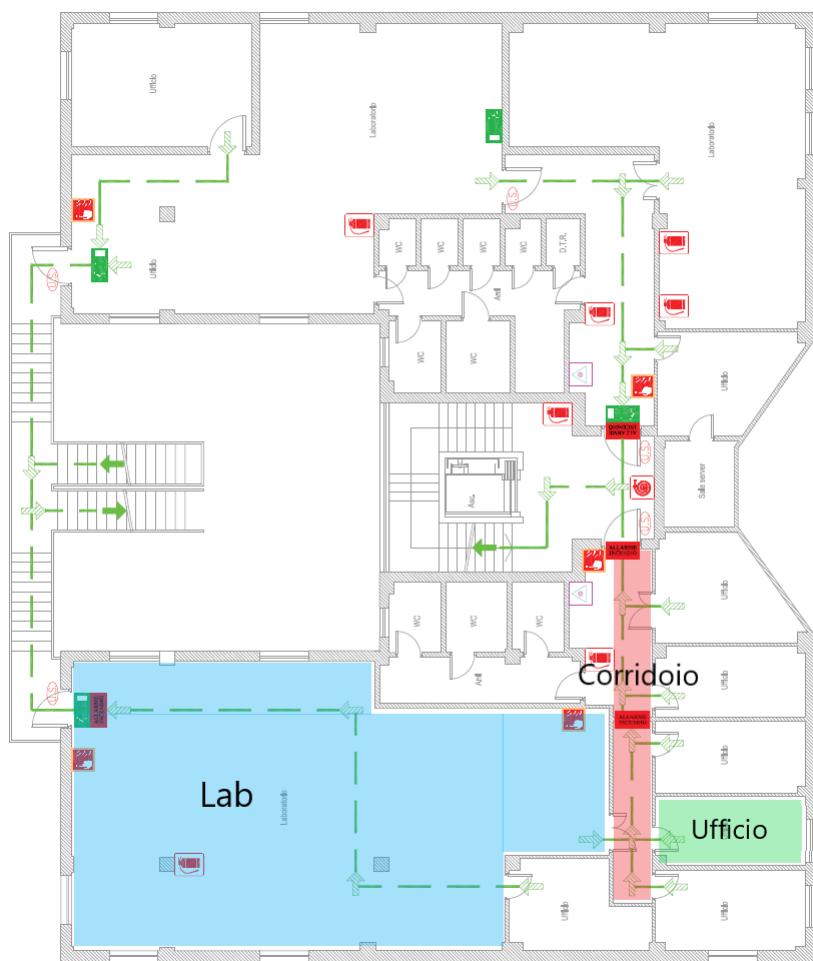


Figura 1.2: Zone coperte da videosorveglianza

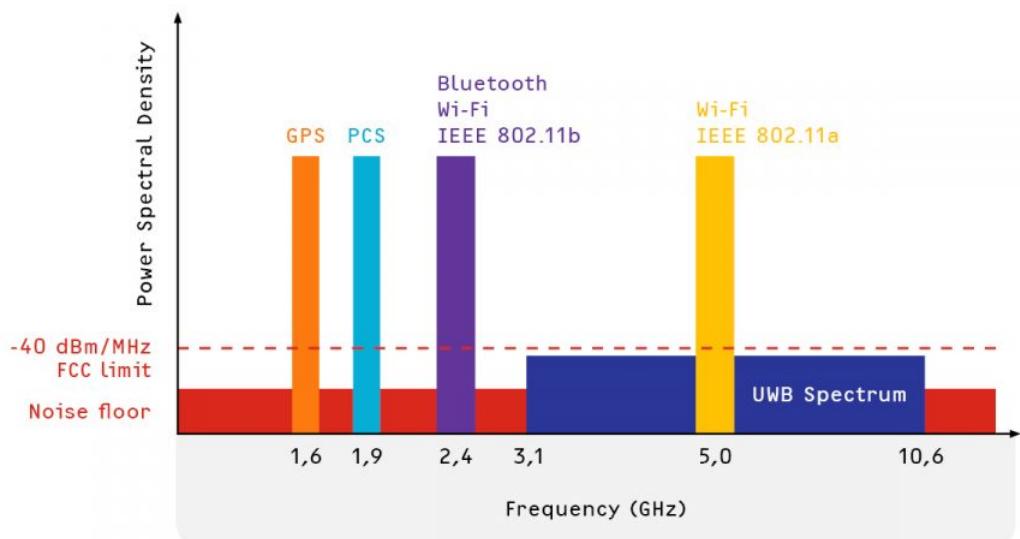
Capitolo 2

UWB e RTLS

2.1 Banda-ultralarga (UWB)

2.1.1 Le caratteristiche

Con il termine Ultra-wideband (banda-ultralarga) si fa riferimento a una tecnica di trasmissione che utilizza impulsi a radiofrequenza di durata molto breve (dell'ordine dei nanosecondi) e di conseguenza con occupazione spettrale molto ampia.

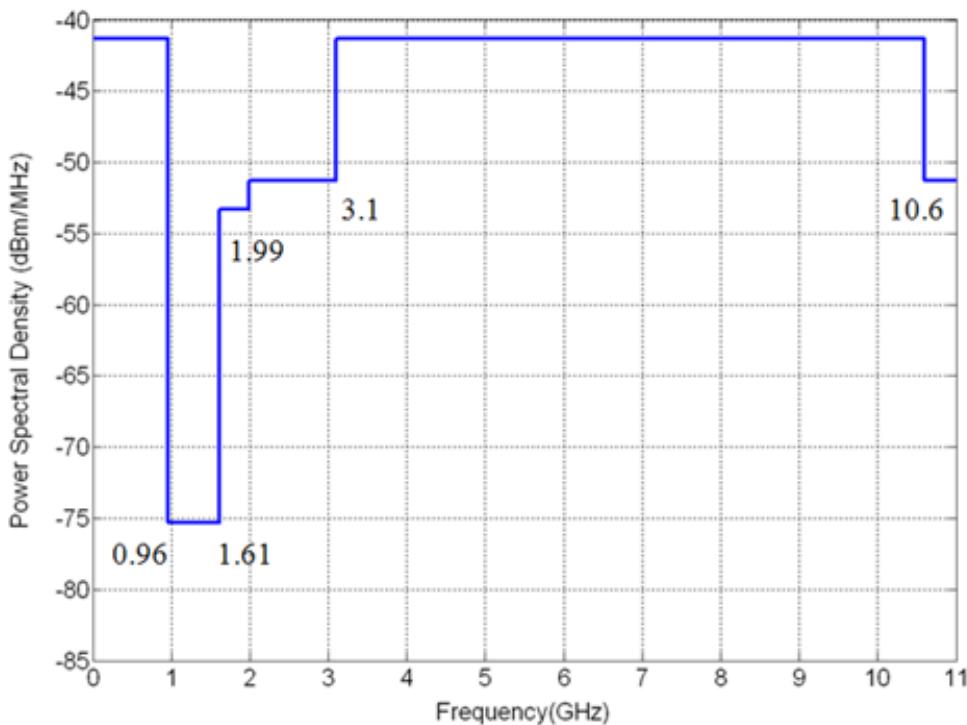


Fonte: www.eliko.ee

Figura 2.1: Occupazione spettrale di varie tecnologie.

Come possiamo notare questo sistema ha una ampiezza di banda notevolmente superiore a quella abitualmente associata a sistemi convenzionali [1]. La Federal Communications Commission (FCC) [2], definisce la banda-ultralarga come un "tipo di sistema di trasmissione senza fili a spettro diffuso che ha una banda larga almeno il 25% della frequenza centrale o, in alternativa, un'ampiezza di 500 MHz o più". La stessa FCC nel 2002, ha pubblicato il First Report and Order

(R&O) [2], che consentiva il funzionamento e l'impiego commerciale di dispositivi UWB senza licenza. Ci sono tre classi di dispositivi definiti nel documento: (1) sistemi di imaging (ad esempio, sistemi radar *ground-penetrating*, sistemi di *wall-imaging*, sistemi di *throughwall imaging*, sistemi di sorveglianza e sistemi medici), (2) sistemi radar veicolari e (3) sistemi di comunicazione e misurazione. La FCC ha assegnato un blocco di spettro radio senza licenza da 3,1 a 10,6 GHz con limite di 43 dBm/MHz per le applicazioni di cui sopra, dove a ciascuna categoria è stata assegnata una maschera spettrale specifica (come descritto nella decisione della FCC adottata il 14 febbraio 2002, pubblicata il 22 aprile 2002) [3]. Nella figura 2.2 è riportato un esempio di una maschera spettrale per sistemi commerciali interni secondo le normative FCC.



Fonte: www.eliko.ee

Figura 2.2: Maschera spettrale FCC per sistemi commerciali indoor.

L'UWB presenta alcune caratteristiche specifiche che lo rendono interessante per diverse applicazioni [4]:

- **Svariati GHz di spettro libero** che consente una grande flessibilità di comunicazione.
- **Nessuna interferenza** dipendente dalla limitatezza della potenza utilizzata, la quale minimizza le potenziali interferenze con altri sistemi di comunicazione.
- **Basse potenze di trasmissione**, perché i trasmettitori utilizzano tecnologia CMOS a basso consumo (meno di 1 mW) che permettono di utilizzare batterie per almeno 4-5 anni.
- **Impulsi brevi** che eliminano il problema dell'interferenza tra il segnale diretto e quelli che arrivano ritardati perché riflessi durante il percorso: il

segnaletico "proprio" è tanto breve da essere ricevuto ed elaborato prima che arrivi un segnale riflesso.

- **Alta velocità di trasmissione** dei dati a corta distanza: 500 Mbps a 3 metri, 200 Mbps a 10 metri.
- **Nessun ostacolo di trasmissione**, perché la capacità di penetrazione delle frequenze UWB è notevole, e poco influenzata da metalli e liquidi.
- **Geolocalizzazione dei tag**, perché è possibile definire le coordinate spaziali che identificano la posizione del tag con una precisione di 5-10 centimetri.

2.1.2 La storia

Da più di vent'anni queste tecnologie hanno avuto largo impiego nel settore militare, sia nei sistemi di comunicazione sia nei sistemi di identificazione e geolocalizzazione [4]. Alla fine del 2004, le autorità USA (la Federal Communications Commission), hanno permesso una liberalizzazione dell'uso di queste tecnologie, che è quindi diventato interessante anche per applicazioni civili.

La storia moderna di UWB è lunga più di 50 anni ed è stata monopolizzata dal settore militare:

- Il modello degli short pulses è stato scoperto negli anni '50 nei laboratori del MIT (Stati Uniti).
- I laboratori Sperry guidati da G. Ross hanno introdotto significativi contributi negli anni '60 [5, 6, 7].
- I brevetti e le applicazioni della DARPA (agenzia di ricerca militare statunitense) si sono sviluppati dal 1970 al 1990.
- Nel 2004 la prima applicazione UWB è stata declassificata dal segreto militare e resa disponibile.
- Nel periodo 1998-2004 la FCC è stata incaricata di studiare le modalità per rendere disponibile tecnologie UWB, poiché DARPA aveva sviluppato tecnologie più sofisticate a frequenze più alte.

Dopo aver verificato le condizioni necessarie per evitare interferenze con i sistemi radio esistenti, si è aperta la possibilità di sviluppare applicazioni civili UWB. La sua nascita risale all'inizio degli studi sull'elettromagnetismo applicato alle comunicazioni [4], con i primi esperimenti di Tesla sulle onde herziane che si sono sviluppati dal 1870 al 1890. Ma fu lo stesso Guglielmo Marconi che dall'anno 1894 al 1896 sviluppò un sistema di trasmissione a impulsi che fu poi brevettato dallo stesso Marconi. Successivamente Reginald Fessenden, nel 1900, riuscì a trasmettere la voce umana a più di un miglio utilizzando gli impulsi radio. Infine nel 1946 Black e altri, riuscirono a trasmettere e ricevere contemporaneamente fino a 1600 miglia di distanza utilizzando un sistema a microonde ad impulsi.



Fonte: www.radiomarconi.com

Figura 2.3: Brevetto di Marconi riguardante la comunicazione ad impulsi.

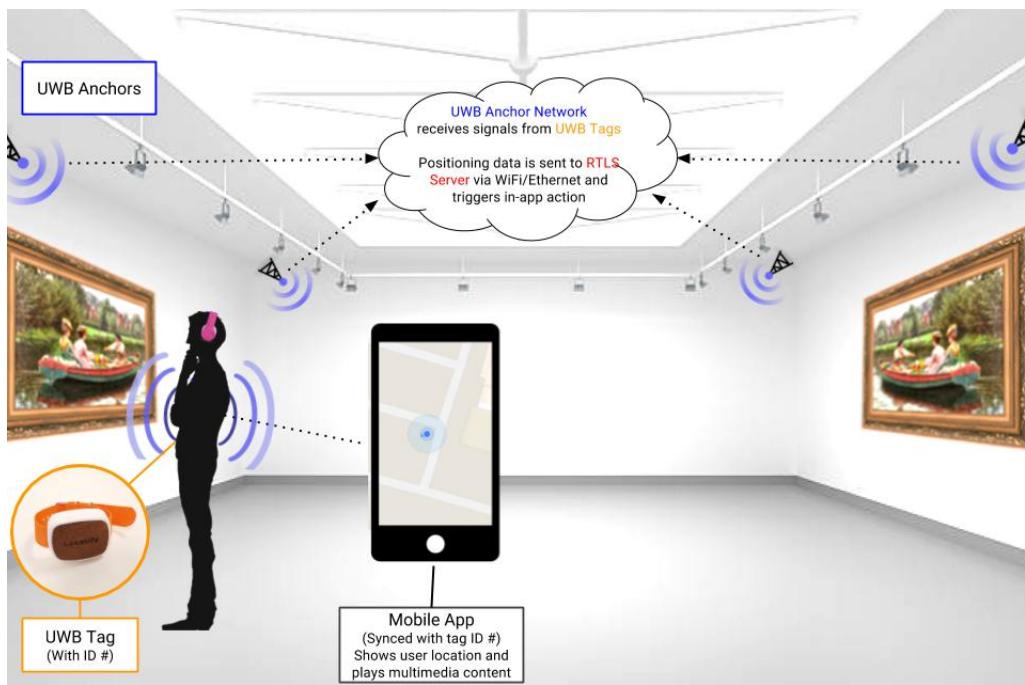
2.1.3 Gli ambiti applicativi

La tecnologia UWB viene utilizzata in svariate applicazioni di differente natura, le quali si possono raggruppare in tre principali ambiti [4]:

- **Sistemi di Comunicazione**, permettono di realizzare reti wireless con alta capacità di trasmissione a corto raggio e hanno la capacità di essere difficilmente intercettabile da terzi.
- **Sistemi di localizzazione**, grazie alla possibilità di operare stime precise dei tempi anche in ambienti indoor caratterizzati da numerose riflessioni dei segnali radio. Questo ambito si analizzerà dettagliatamente nella prossima sezione.
- **Radar**, in applicazioni quali sistemi per evitare ostacoli e collisioni, studi sulla penetrazione dei materiali, e rilevamento di oggetti.

2.1.4 Indoor positioning

Le tecnologie satellitari di posizionamento outdoor, come ad esempio il GPS, sono strumenti validi e affidabili in termini di precisione e accuratezza nel posizionamento. In ambito indoor queste tecnologie hanno come limite le ostruzioni fisiche che diminuiscono la potenza del segnale o ne alterano il percorso (multipath) [8]. Questo accade solitamente in ambienti indoor, ad esempio i centri commerciali, gli aeroporti, grandi negozi, stazioni ferroviarie, musei e tanti altri. Un esempio nella figura 2.4. In generale i servizi GNSS danno una precisione di 1-3 metri nella maggior parte dei luoghi outdoor, mentre per gli spazi indoor, i sistemi UWB hanno una precisione migliore di questa (fino a 5 cm). Questo è un aspetto importante, perché la scala naturale di tali spazi (allestiti da stanze e mobili) è più piccola [9].



Fonte: www.locatify.com

Figura 2.4: Esempio di un sistema di localizzazione UWB.

2.2 Sequitur RTLS

Sequitur è un Real Time Locating System che sfrutta la tecnologia UWB descritta nella sezione precedente. Si tratta di un sistema composto da elementi hardware e firmware, e da applicazioni per l’utente, per high-accuracy ranging, positioning, tracking e navigation [10]. Sequitur permette di localizzare un utente mobile, anche indoor, con una precisione di pochi centimetri e un raggio operativo di decine di metri. Sequitur si basa sulla tecnologia UWB (ultra-wide bandwidth) che fornisce una precisione di localizzazione senza precedenti nell’ordine dei centimetri, precisione non raggiungibile attraverso altre tecnologie wireless. Consente inoltre una comunicazione wireless robusta in ambienti complessi, come in contesti industriali. Il sistema si compone essenzialmente di due elementi: ancore e tag.

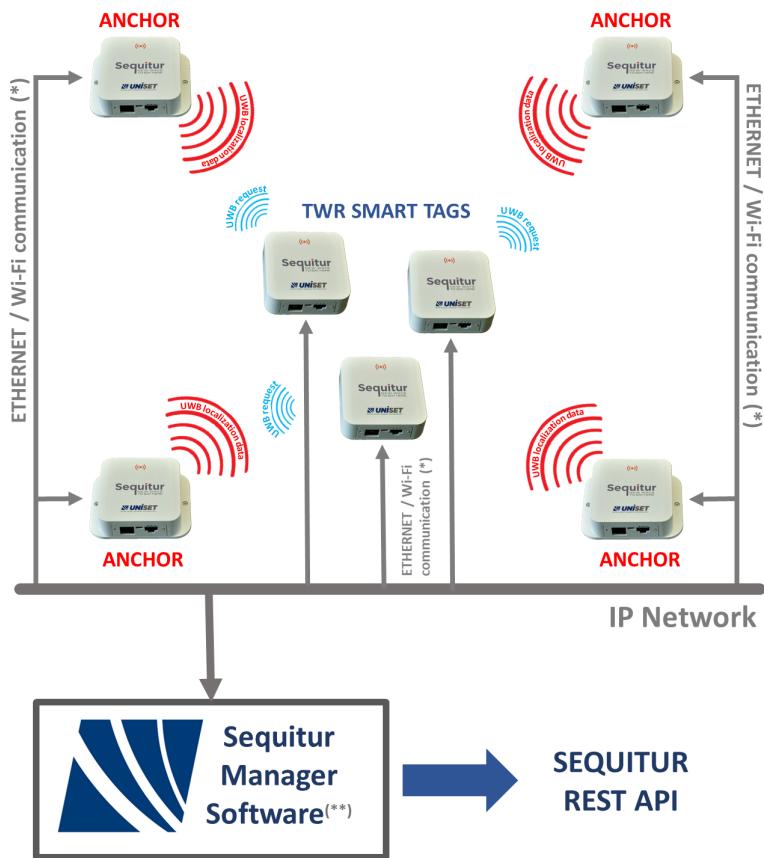
Le ancore sono dispositivi distribuiti in posizioni fisse e note che compongono l’infrastruttura che consente la localizzazione dei tag. La posizione e il numero di ancore dipende dall’area (o dal volume in caso di localizzazione 3D), dalla geometria e dall’ambiente all’interno del quale l’utente vuole localizzare i tag. Il sistema minimo è costituito da almeno 4 ancore.

I tag sono i dispositivi da localizzare. Esistono diversi tipi di tag, a seconda dell’applicazione e della capacità di stimarne internamente la posizione, come avviene per un ricevitore GNSS, o esternamente attraverso un’unità centrale.

2.2.1 InGPS TWR

Sequitur InGPS TWR (Two Way Ranging) utilizza la tecnica TWR, la quale si basa sulla misurazione della distanza tra un tag e una serie di ancore. Per ogni iterazione di localizzazione il tag chiede periodicamente ad un insieme di ancore di effettuare una misura della distanza. Quindi, utilizzando la posizione nota delle

ancore, il tag stima la propria posizione nell'ambiente in termini di coordinate (x, y, z). In questo caso il tag è sia un ricevitore UWB che un trasmettitore UWB. Per questo motivo, se due o più tag coesistono nella stessa area, potrebbero verificarsi collisioni radio (non c'è coordinamento tra i tag). Per collisione si intende la sovrapposizione di due segnali radio che non permette una corretta ricezione del segnale da parte del ricevitore. Per questo motivo InGPS TWR ammette solo pochi tag nella stessa area di localizzazione, e non permette la condivisione di una stessa ancora da parte di più tag nello stesso momento. Il funzionamento di InGPS TWR è decentralizzato, quindi il calcolo della posizione è realizzato a lato tag e la posizione è resa disponibile esternamente tramite API.



Fonte: Manuale Sequitur di UNISET [10]

Figura 2.5: Architettura di Sequitur InGPS TWR.

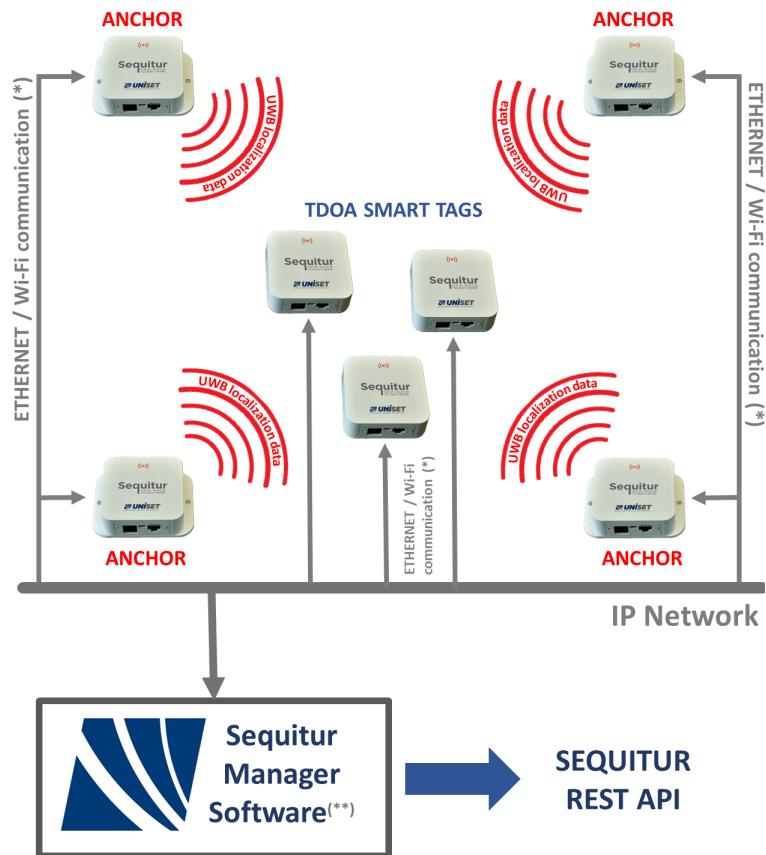
2.2.2 InGPS TDOA

Sequitur InGPS TDOA utilizza la tecnica TDOA, la quale si basa sulla stima a lato tag della differenza di tempo tra i segnali emessi dalle ancore. Il principio operativo TDOA può essere spiegato utilizzando un'analogia con un sistema di navigazione GNSS. Le ancore hanno lo stesso ruolo dei satelliti e il tag può essere visto come un ricevitore GNSS. Questa tecnica di localizzazione [11] prevede la misura della differenza dei tempi di propagazione dei segnali tra il terminale (tag) e due ancore (nel nostro caso le ancore); Misurato il tempo t_i impiegato dal segnale a propagarsi tra terminale e trasmettitore e nota la velocità con cui il

segnale si propaga nello spazio ($c = 3 \cdot 10^8 \text{ m/s}$), ciascuna misura, riferita ad una coppia di ancore poste nei punti di coordinate (x_i, y_i) e (x_j, y_j) , individua una iperbole, nei cui fuochi sono posizionate le ancore stesse, di equazione

$$d = c \cdot (t_i - t_j) = \sqrt{(X_i - x)^2 + (Y_i - y)^2} - \sqrt{(X_j - x)^2 + (Y_j - y)^2}$$

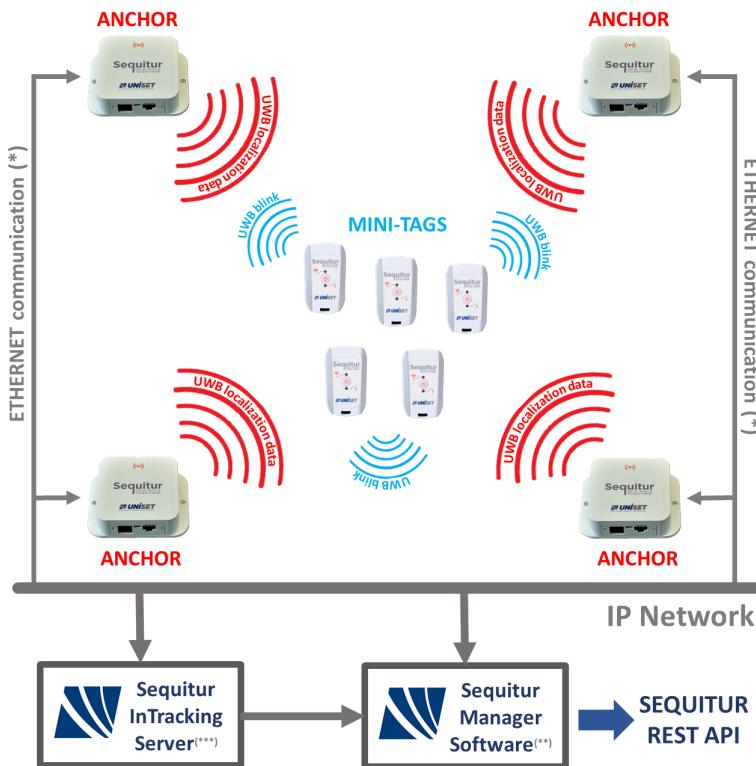
dove (x, y) sono le coordinate incognite del punto in cui si trova il tag. Realizzando questa misura con due coppie indipendenti di ancore, si individuano due iperbole la cui intersezione costituisce il punto cercato. Il tag ascolta continuamente il canale radio per ricevere i dati dalle ancore e calcola il tempo di arrivo dei diversi segnali ricevuti, che è correlato alla distanza tra l'ancora e il tag. Quindi, utilizzando la posizione nota delle ancore, il tag stima la propria posizione nell'ambiente. In questo caso il tag è un semplice ricevitore UWB, di conseguenza i tag non trasmettono sul canale radio permettendo la coesistenza simultanea di un numero illimitato di tag, senza determinare una degradazione della qualità della localizzazione. Come accadeva nella modalità descritta nella sezione precedente, il funzionamento di InGPS TDOA è decentralizzato, ovvero l'algoritmo di localizzazione gira a lato tag.



Fonte: Manuale Sequitur di UNISET [10]
Figura 2.6: Architettura di Sequitur InGPS TDOA.

2.2.3 InTracking TDOA

Sequitur InTracking TDOA (Time Difference of Arrival) è un'architettura RTLS centralizzata, che utilizza la tecnica TDOA che abbiamo descritto nella sezione precedente. Nell'approccio centralizzato, un'unità centrale è collegata a tutte le ancore tramite rete Ethernet e stima le posizioni di tutti i mini-tags nell'ambiente. I mini-tags InTracking, durante il normale funzionamento, trasmettono periodicamente solo un breve UWB blink che viene ricevuto dalle ancore. Il tempo di arrivo del messaggio inviato dal tag viene poi elaborato dall'unità centrale per stimare la posizione del tag. I mini-tag InTracking TDOA non calcolano internamente la propria posizione, la quale è disponibile solo nell'unità centrale. Dal momento che tutti i mini-tag trasmettono gli UWB blink per la localizzazione, solo un numero limitato di essi è supportato e dipende dal blinking rate dei tag. Generalmente, l'alta densità di tag è consentita quando il blinking rate è basso.



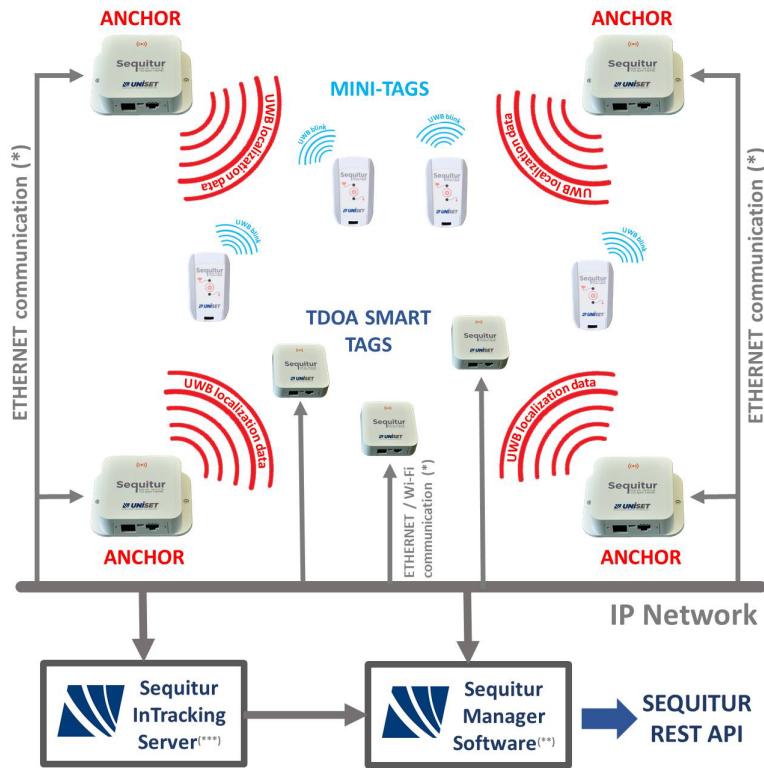
Fonte: Manuale Sequitur di UNISET [10]

Figura 2.7: Architettura di Sequitur InTracking TDOA.

2.2.4 Coesistenza tra InTracking TDOA e InGPS TDOA

Sequitur InGPS TDOA e InTracking TDOA possono anche coesistere. La configurazione è l'unione di Sequitur InGPS TDOA e Sequitur InTracking TDOA. Grazie alla compatibilità delle tecnologie Sequitur TDOA, questa configurazione permette di combinare diversi tipi di tag. Sia i tag che i mini-tag possono operare nello stesso ambiente con la stessa infrastruttura (ancore). Il comportamento dei tag è lo stesso per ogni rispettiva famiglia come descritto nella sezione precedente. L'unica differenza che deve essere presa in considerazione è la densità di

mini-tag che può avere un impatto anche sulla qualità di localizzazione dei tag (InGPS TDOA). Per questa configurazione la posizione dei mini-tags (InTracking TDOA) è disponibile solo a lato server mentre quella dei tags (InGPS TDOA) è disponibile a lato tag.



Fonte: Manuale Sequitur di UNISET [10]

Figura 2.8: Architettura di coesistenza tra Sequitur InTracking TDOA e InGPS TDOA.

Capitolo 3

Analisi

3.1 Requisiti

Il software, commissionato dall'azienda UNISET¹, mira alla realizzazione di uno strumento di videosorveglianza integrato al sistema di indoor positioning Sequitur descritto nel dettaglio nel capitolo precedente.

3.1.1 Requisiti funzionali

- Il suddetto software dovrà occuparsi di reperire informazioni circa la posizione di un determinato tag, dato un mondo in input dall'utente. Un mondo è definito come un insieme di stanze in cui l'oggetto si può muovere; una o più stanze possono formare una zona.
- E' sufficiente seguire un tag alla volta, analizzando informazioni in tempo reale sulle sue coordinate, la zona in cui si trova e nel caso si trovasse in una zona con telecamera attiva, dovrà essere disponibile lo stream video della suddetta telecamera. Il sofware deve comunque permettere di poter selezionare il tag desiderato in qualsiasi momento.
- Il mondo attuale deve essere selezionabile da una lista di mondi disponibili in quel momento nell'environment.
- Deve essere disponibile un modo per gestire le telecamere. Deve esistere per l'utente la possibilità di aggiungere, eliminare o modificare le telecamere a seconda delle necessità. Nell'aggiungere una nuova telecamera si dovrà specificare, tra le altre informazioni, anche in quale zona è attiva.

3.1.2 Requisiti non funzionali

- Il software dovrà essere in grado di mostrare informazioni (coordinate, zone) aggiornate almeno ogni secondo, non devono essere presenti latenze nel passaggio tra lo stream video di una telecamera e un'altra, o ritardi nel caricamento di informazioni nel passaggio tra diversi tag.

¹<http://www.unisetcompany.com/>

3.2 Analisi e modello del dominio

3.2.1 Descrizione dettagliata e UML del modello

Il software dovrà essere in grado di accedere all'insieme dei mondi presenti nell'environment messo a disposizione dalle API di Sequitur. Gerarchicamente, ogni environment è composto da almeno un mondo di default, in ogni mondo ci possono essere zero, una o più zone rappresentate da un identificatore e un nome. I tag, anche detti device, si muovono all'interno di un mondo assumendo posizioni (x, y, z) che sono note al server Sequitur, inoltre è nota anche l'eventuale zona in cui si trova in un determinato momento. L'applicazione dovrà utilizzare queste informazioni per attivare la telecamera corrispondente e nel caso di telecamera PTZ elaborare i calcoli necessari a seguire l'oggetto nella stanza. Delle telecamere disponibili si deve conoscere la posizione in cui è stata montata e la zona in cui è attiva.

Si propone un riassunto delle entità fondamentali del dominio:

- **World**: corrispondente al mondo nella descrizione del dominio.
- **Camera**: entità collegata al concetto di videocamera e alle sue funzionalità, si trova in una Position ed è attiva in una Zone
- **Position**: entità necessaria a esprimere in coordinate la localizzazione di un oggetto.
- **Device**: entità che rappresenta un tag.

Questi elementi costitutivi del problema sono sintetizzati nell'immagine 3.1, che mostra anche la loro interazione con il resto del programma.

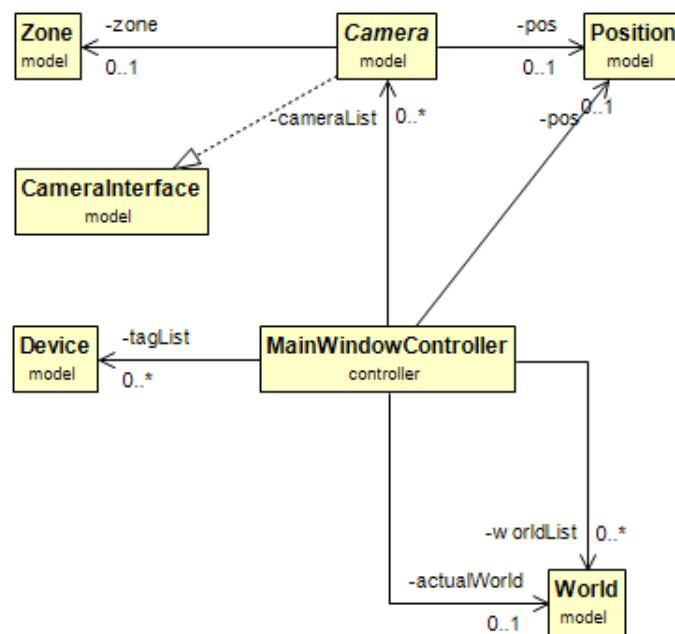


Figura 3.1: Schema UML del dominio.

La difficoltà primaria sarà quella mantenere le informazioni richieste al server coerenti con il device di cui l'utente vuole conoscere la posizione. Questo

richiederà di memorizzare quale tag si sta analizzando in un determinato lasso di tempo.

Il requisito non funzionale riguardante l'interfaccia utente richiederà l'utilizzo di thread diversi su cui effettuare operazioni che potrebbero, richiedendo troppo tempo per essere completate, bloccare l'interfaccia utente in uno stato non aggiornato in tempo reale.

3.2.2 Casi d'uso

L'interfaccia dovrà permettere all'utente di selezionare un mondo di cui si vogliono avere informazioni, successivamente si mostrano tutti i tag disponibili in quel mondo. Nel caso non vi siano telecamere attive, sarà comunque possibile ottenere dati sulla posizione e visualizzare una mappa della localizzazione in tempo reale del tag. L'utente può aggiungere nuove telecamere, associandole a una determinata zona. Nella figura 3.2 presentiamo uno diagramma dei casi d'uso principali per l'interfaccia utente.

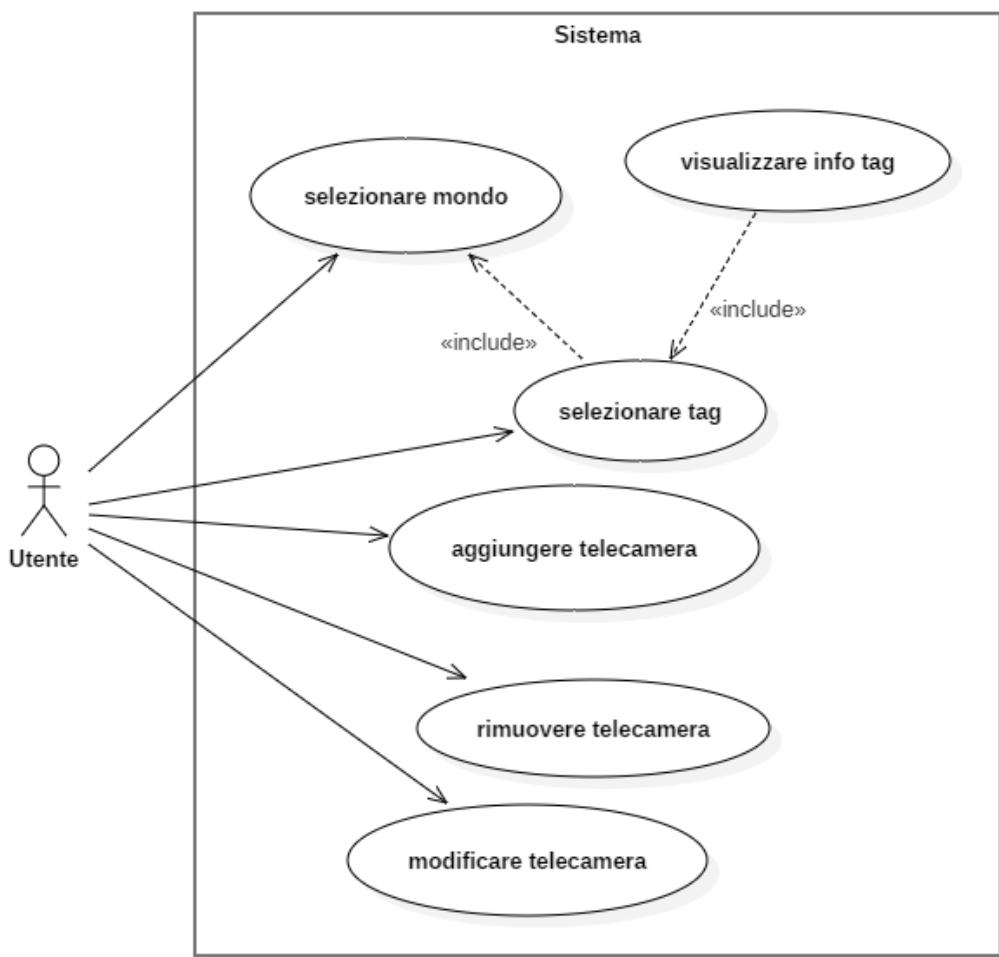


Figura 3.2: Diagramma UML che rappresenta i casi d'uso principali dell'interfaccia

Capitolo 4

Progettazione

4.1 Hardware e integrazione con il sistema esistente

Per quanto riguarda la parte hardware del sistema, di seguito si mostra un diagramma UML di deployment che sintetizza le principali interazioni tra le parti del sistema esistente e il sistema sviluppato.

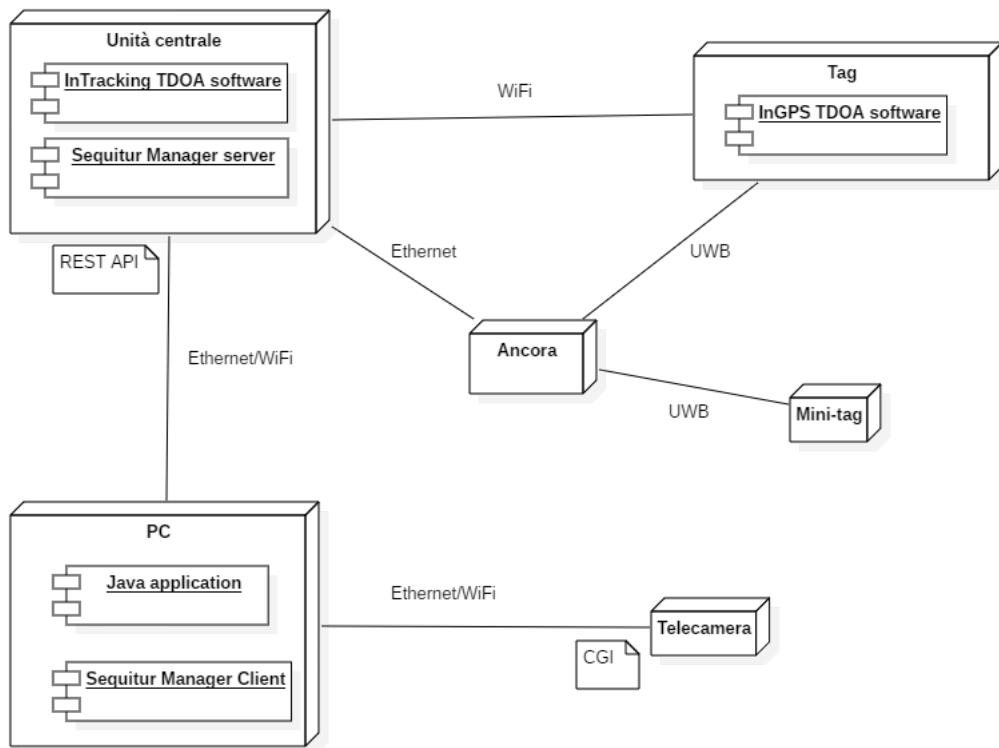


Figura 4.1: Diagramma UML di deployment per l'integrazione col sistema esistente

Come si nota nel diagramma appena mostrato, la modalità con cui Sequitur viene eseguito in questo caso è InGPS TDOA e InTracking TDOA coexistence, infatti sono presenti sia tag che mini-tag. Per il software sviluppato la differenza tra i tipi di tag e la modalità di esecuzione di Sequitur non è determinante. I tag e i mini-tag comunicano con le ancore in modo che queste possano inviare

informazioni sulla posizione dei device, attraverso la rete Ethernet, all’unità centrale che esegue il software di accentramento dati Sequitur Manager e, nel caso dei mini-tag, elabora la posizione attraverso l’algoritmo InTracking. I tag (anche detti Smart tag) invece elaborano la propria posizione eseguendo il software InGPS TDOA e comunicano l’informazione all’unità centrale via WiFi. Il software sviluppato ottiene informazioni dal server attraverso le REST API dell’unità centrale, e a sua volta comunica con le telecamere utilizzando i comandi CGI messi a disposizione da queste.

4.2 Design architetturale

Innanzitutto si è sviluppato un diagramma UML flowchart che riassume la logica del software, per poi analizzare più nello specifico quali classi si occuperanno di gestirla. In figura 4.2 si nota che inizialmente sarà necessario richiedere l’host e la porta su cui opera il server Sequitur, si tenterà quindi un collegamento al server e se questo va a buon fine si potrà avviare l’applicazione. In secondo luogo si dovrà richiedere al server la lista di mondi disponibili e, considerando che almeno un mondo di default è sempre disponibile, si potrà procedere ad ottenere la lista dei tag presenti nel primo mondo, o nel mondo selezionato dall’utente. Una volta ottenuta la lista di tag, si possono ottenere informazioni sul tag selezionato dall’utente e verificare in quale zona si trova l’oggetto. In base alla posizione di quest’ultimo, l’interfaccia dovrà informare l’utente se:

- Il tag si trova fuori dall’area di localizzazione coperta da Sequitur.
- Il tag si trova fuori dalle zone predefinite per il mondo selezionato.
- Il tag si trova in una delle zone predefinite ma non è attiva una telecamera in quella zona.
- Il tag si trova in una zona in cui è attiva una telecamera.

I primi due casi si risolvono mostrando all’utente solo la mappa della localizzazione dell’oggetto e le informazioni sulla posizione, mentre nell’ultimo caso sarà necessario interfacciarsi con le telecamere. Nel caso la telecamera attiva nella zona in cui si trova l’oggetto sia mobile, prima di mostrare il frame dallo stream video sarà necessario elaborare il comando di movimento alla telecamera, ovvero calcolare l’angolo di PAN e TILT. Come si può notare da questa breve descrizione, alcune operazioni dovranno essere svolte in parallelo e ripetutamente, come richiedere dati sul tag al server o mostrare i frame della telecamera attiva in una certa zona. Si è deciso quindi di utilizzare dei task che si possano occupare parallelamente di compiti necessari all’aggiornamento continuo dell’interfaccia utente.

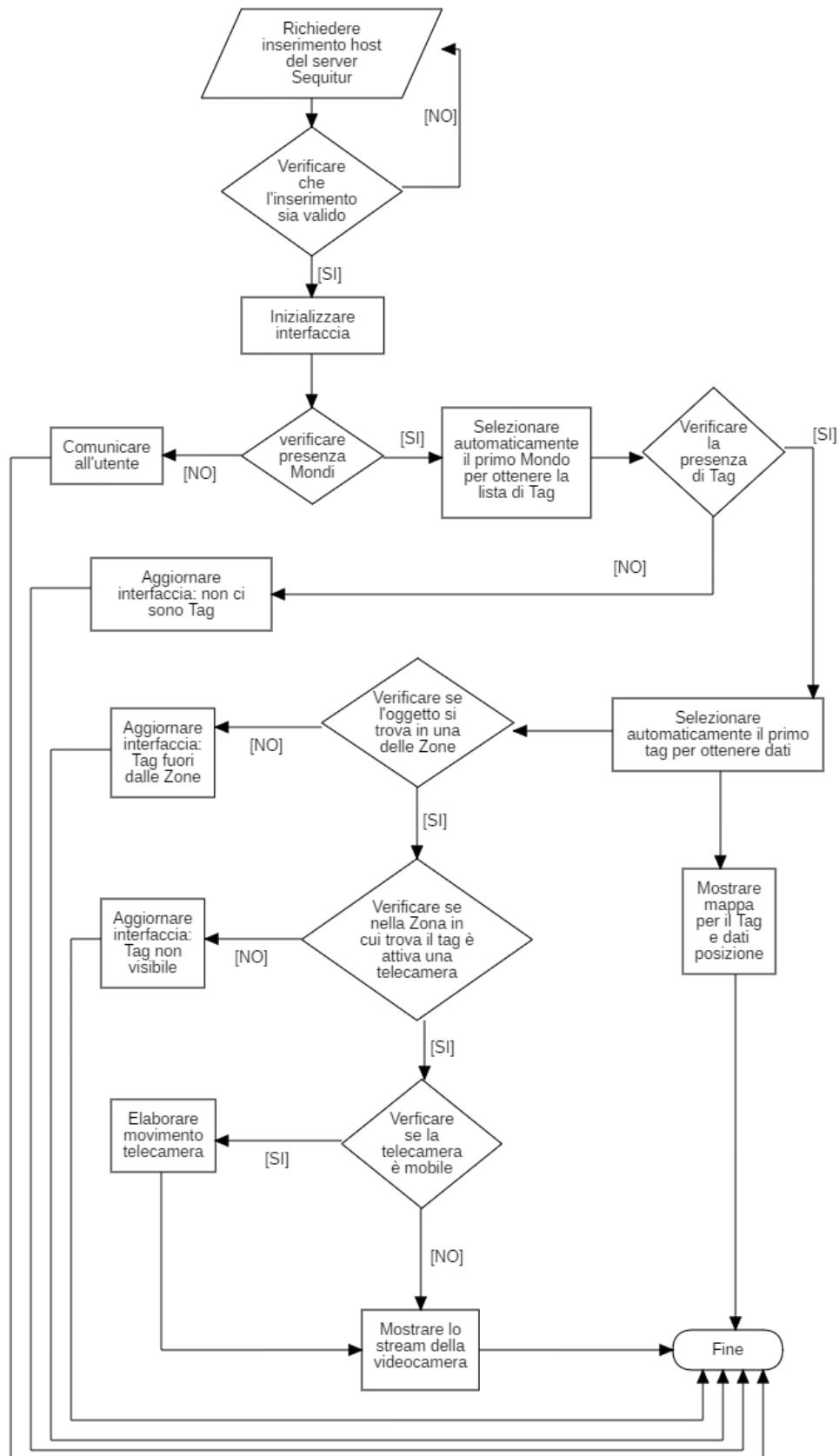


Figura 4.2: Diagramma UML flowchart del funzionamento del sistema

Come accennato la logica del software risiede principalmente nei due task gestiti dal controller della finestra principale.

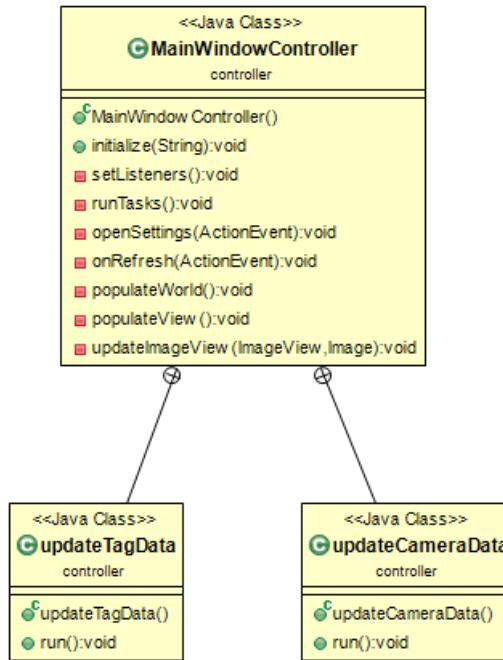


Figura 4.3: Interazione tra i task e il controller della finestra principale

- **UpdatetagData**: questo task si dovrà occupare di richiedere al server la posizione del tag selezionato dall'utente e aggiornare l'interfaccia in accordo con queste informazioni.
- **UpdateCameraData**: questo task dovrà gestire l'interazione con le videocamere, in base alla zona in cui si trova il tag corrente, aggiornando l'interfaccia con il frame della telecamera attiva (o un frame vuoto nel caso non siano attive videocamere in quella zona) e inviando il comando di movimento alle telecamere mobili.

Un altro componente fondamentale nella logica dell'applicazione è la classe **Controller** che avrà il compito di interagire con il server, ovvero gestire direttamente le chiamate alle REST API di Sequitur e di gestire le telecamere attive durante l'esecuzione dell'applicazione. I compiti di questa classe dovranno essere:

- Aggiungere, rimuovere videocamere
- Ottenere lista dei tag
- Ottenere lista dei mondi
- Ottenere posizione di un tag
- Ottenere mappa per la posizione di un tag
- Ottenere la lista delle zone

- Mantenere aggiornata l'informazione su qual è l'ultimo tag di cui è stata richiesta la posizione e l'ultima videocamera attiva

Nella figura 4.4 si mostra l'interazione di questa classe con il controller della finestra principale e con la classe Camera, inoltre solo elencati i metodi pubblici che soddisfano i compiti appena elencati.

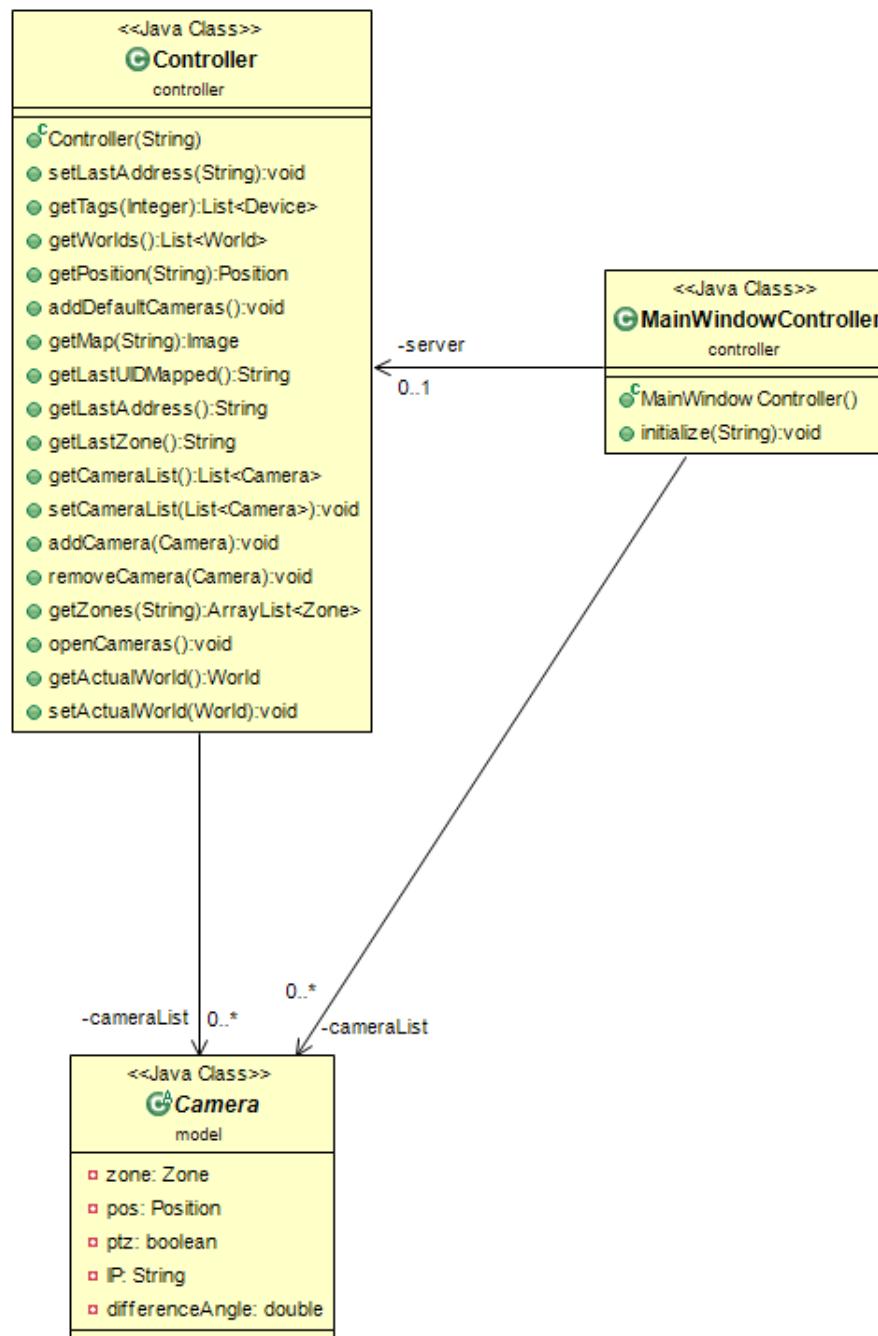


Figura 4.4: Interazione tra il controller della finestra principale e la classe Controller dell'applicazione e la classe Camera

4.3 Note di sviluppo e librerie

4.3.1 Testing automatizzato

Il testing automatizzato è stato effettuato con JUnit, attraverso l'implementazione della classe CameraTest che si occupa di controllare se le operazioni del Controller che gestiscono le telecamere vanno a buon fine anche in casi di errore. I metodi implementati verificano il corretto inserimento di una videocamera, la modifica e l'eliminazione, l'inserimento di tre telecamere di default e inoltre controllano che i campi del Controller siano inizializzati e modificati correttamente in base allo stato dell'applicazione. I test implementati sono stati utilizzati anche per verificare gli stati di errore in cui il sistema deve trovarsi in determinati casi, come ad esempio quando non esiste nessun mondo di default o quando l'indirizzo del server non è valido, quando una richiesta http non va a buon fine e così via.

4.3.2 Librerie di supporto

OpenCV

Per lo sviluppo delle funzionalità riguardanti la gestione delle telecamere si è individuato OpenCV¹ come libreria più adatta e completa rispetto alle esigenze del software. Questa libreria include diverse centinaia di algoritmi di computer vision. La versione utilizzata è la 4.1.1 e di seguito viene presentato un elenco delle funzionalità e dei metodi utilizzati.

- `new VideoCapture ()`
Costruttore per l'oggetto VideoCapture che servirà per interagire con la videocamera.
- `VideoCapture.open(String filename)`
Apre un file video oppure uno stream video IP passato come argomento.
- `VideoCapture.grab()`
Il metodo cattura dalla videocamera il frame successivo e restituisce true in caso di successo.
- `VideoCapture.retrieve(Mat image)`
Il metodo decodifica e restituisce il frame appena catturato. Se non ci sono frame catturati (la camera è stata disconnessa o non ci sono più frame nel file video) il metodo restituisce false e la funzione restituisce un'immagine vuota.

Gson

Questa libreria di Google è stata utilizzata per la decodifica degli oggetti Json richiesti al server Sequitur e per la persistenza in memoria dei dati delle telecamere. L'utilizzo di questa libreria è descritta nel dettaglio nella sezione 5.5.2.

¹<http://opencv.org/>

Jfoenix

Per quanto riguarda l'interfaccia grafica, la libreria Jfoenix² ha fornito un valido modello per rendere il design della GUI piacevole, attuale e semplice. Questa libreria open source di Java implementa Google Material Design usando i componenti di Java. E' stato necessario modificare il codice della libreria per sistemare alcuni aspetti dell'interfaccia.

²<http://www.jfoenix.com/>

Capitolo 5

Implementazione

5.1 Interfaccia utente

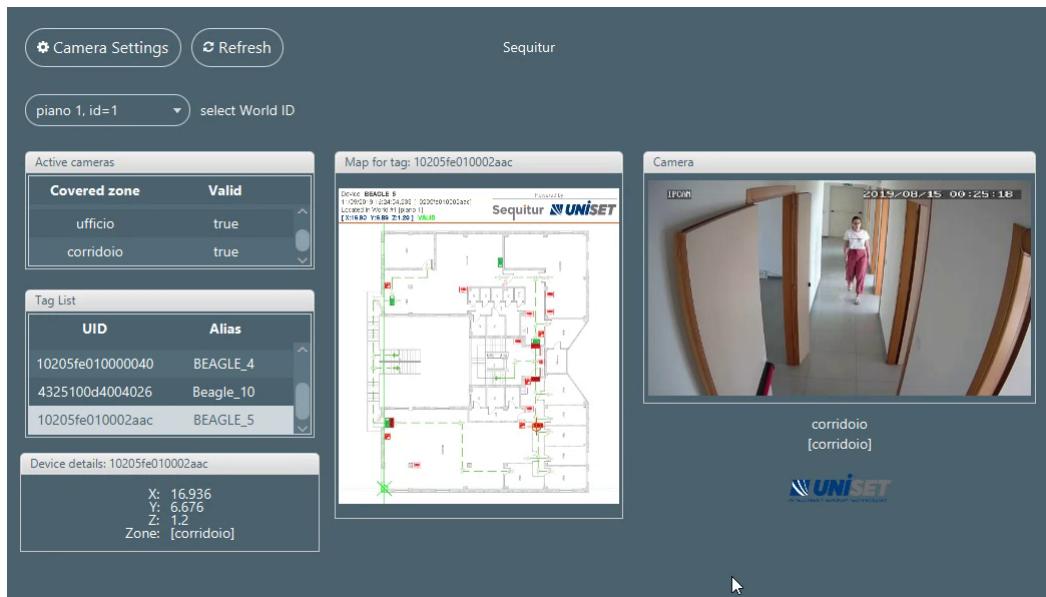


Figura 5.1: Finestra principale dell'applicazione sviluppata.

Nella finestra principale dell'interfaccia è possibile visualizzare a sinistra l'elenco delle telecamere, di cui si mostra la zona di pertinenza e se è attiva, l'elenco dei tag, e i dettagli del tag attualmente selezionato. Nella parte centrale dell'interfaccia si mostra la mappa con la localizzazione del tag corrente e a destra no stream video per il tag selezionato. In alto sono disponibili i comandi per le impostazioni delle videocamere, l'elenco dei mondi e un comando per aggiornare l'elenco dei mondi. La lista dei tag si aggiorna automaticamente quando si modifica il mondo attuale.

Quando il tag si trova in una zona in cui non è presente nessuna videocamera attiva si segnala all'utente come mostrato in figura 5.2.

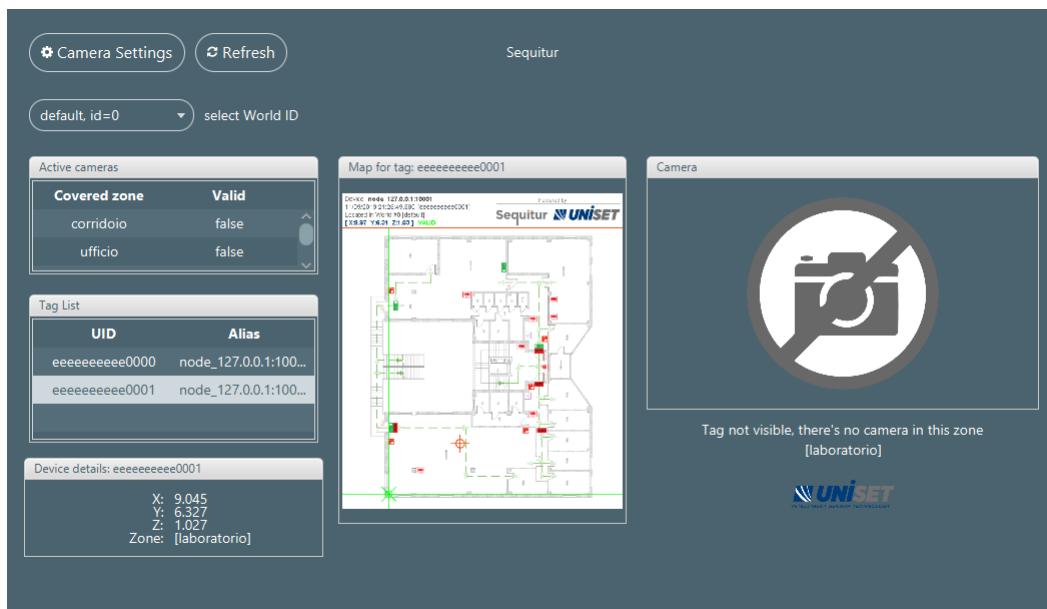


Figura 5.2: Finestra principale dell'applicazione in caso di tag non visibile.

Accedendo alla finestra delle impostazioni è possibile modificare aggiungere o rimuovere le videocamere. Quando si salvano e si chiudono le impostazioni la lista delle telecamere viene aggiornata e vengono chiamati i metodi per interagire con le nuove telecamere. Se l'interazione con queste va a buon fine vengono segnate come attive e il loro stream è disponibile.

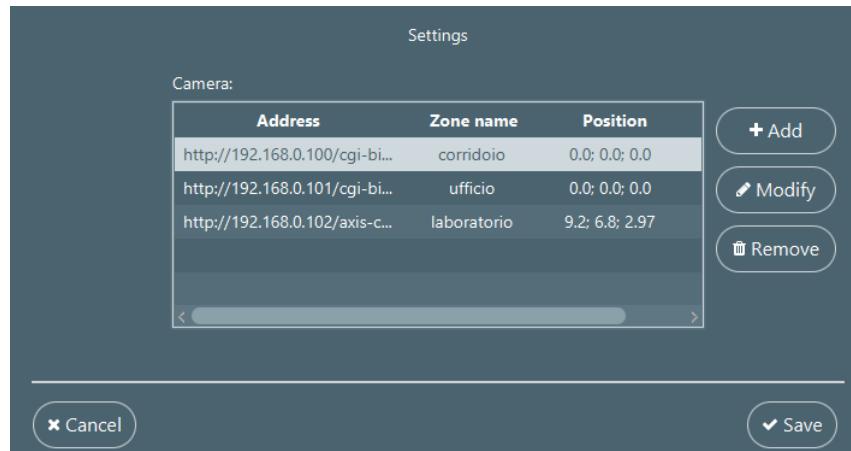


Figura 5.3: Finestra di impostazione delle videocamere.

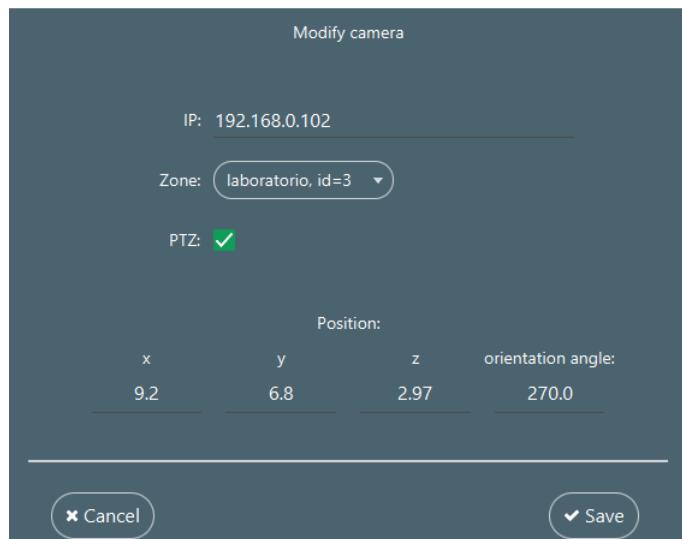


Figura 5.4: Form per modificare le informazioni di una videocamera.

5.2 Funzionamento del sistema

Si descrive di seguito il funzionamento dettagliato del sistema. La logica del sistema si basa sull’implementazione di due task che si occupano di ricavare dati dal server Sequitur ed elaborarli al base al tag selezionato dall’utente.

5.2.1 Camera Task

Questo task si occupa di analizzare la posizione del tag che l’utente ha selezionato ed elaborare le informazioni da mostrare nell’interfaccia. Come mostrato in figura 5.5, inizialmente si controlla se nel mondo corrente esiste almeno un tag, se questa condizione non si verifica si comunica all’utente questa informazione (NO TAGS), altrimenti si procede ad ottenere la posizione del tag corrente, se la posizione non è valida significa che il tag si trova fuori dall’area di localizzazione del software Sequitur, in questo caso si aggiorna l’interfaccia con questa informazione (NOT VALID). In caso contrario si ottiene la zona in cui si trova il tag corrente, se questo campo è vuoto, il tag si trova fuori dalle zone predefinite (OUT OF ZONES). Se invece il tag si trova all’interno di una zona predefinita, si controlla se esistono telecamere attive in quella zona. In caso negativo il tag è considerato non visibile (NOT VISIBLE). Se invece si è trovata una telecamera attiva in quella zona, si mostra lo stream video dell’oggetto localizzato, eventualmente muovendo la telecamera per inquadrarlo. Questo task viene eseguito ogni 50 ms e tutte le operazioni al suo interno che aggiornano l’interfaccia vengono chiamate attraverso l’uso di Platform.runLater() seguendo la regola di JavaFX per cui tutte le modifiche alla GUI devono essere attuate su un Thread FX. Questo metodo può essere chiamato da qualsiasi Thread, esegue il Runnable specificato in un certo momento nel futuro inserendolo in una coda di eventi per poi ritornare immediatamente al chiamante.

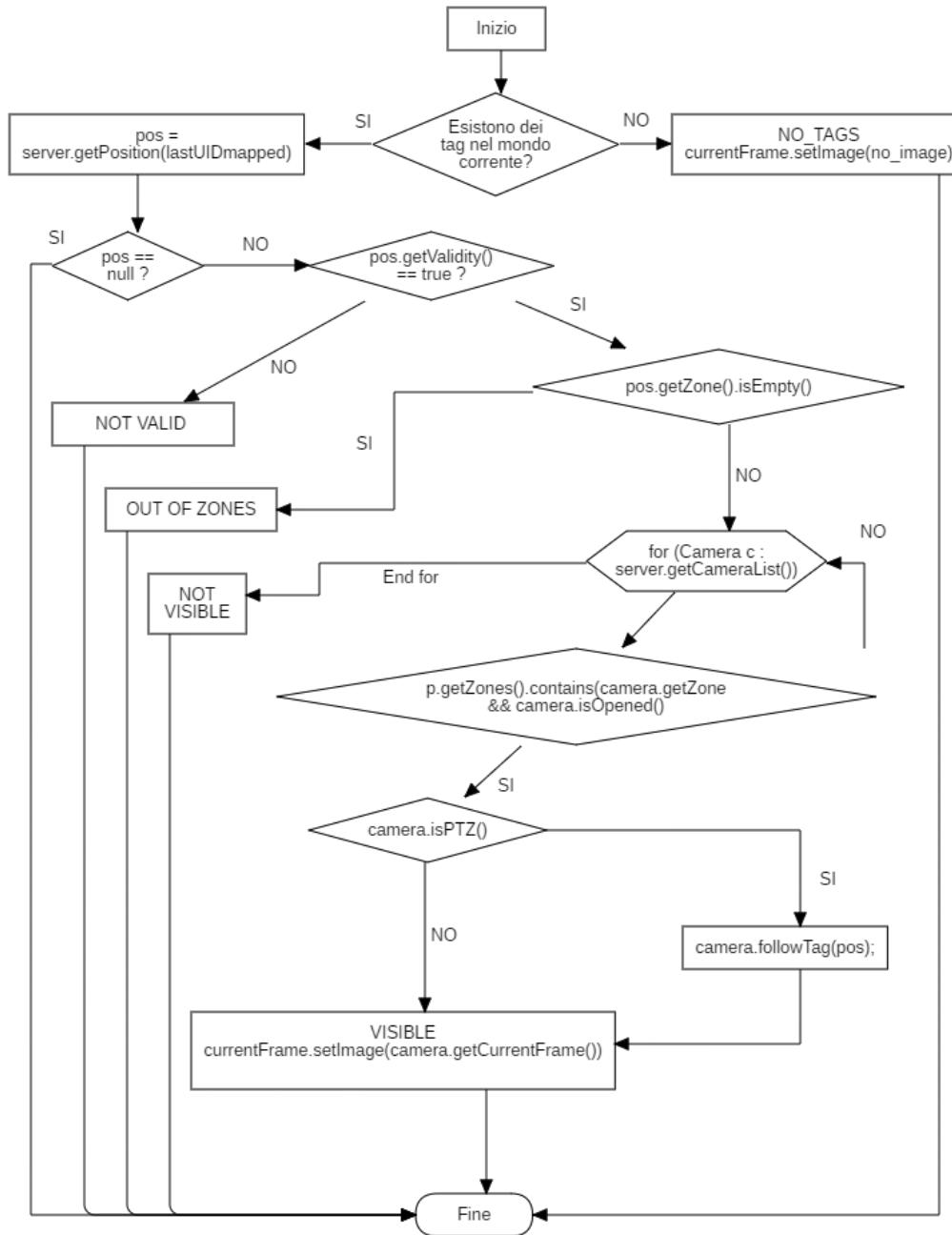


Figura 5.5: Diagramma UML flowchart per il Task che gestisce le telecamere.

5.2.2 Tag Task

Questo task si occupa di aggiornare la mappa della localizzazione del tag corrente e le informazioni sulla posizione del dispositivo (coordinate e zona). Seppur simile al task precedente in quanto ai compiti che deve svolgere, ovvero aggiornare l’interfaccia in base alla posizione dei tag, questo task è attivo anche quando l’applicazione non dispone di telecamere che seguano i tag. Quindi la separazione concettuale e implementativa di questo task dal precedente è stata mantenuta per mettere a disposizione dell’utente un’interfaccia che funzioni correttamente anche senza telecamere attive. Analizzando il funzionamento di questo task,

mostrato in figura 5.6, notiamo che come in precedenza si controlla la presenza di tag nel mondo corrente e successivamente si individua il tag selezionato nella lista di device. Un controllo di sicurezza determina se esiste un tag selezionato, ma questo solitamente non può essere null, dato che alla selezione dell'utente di un mondo, automaticamente si seleziona il primo elemento della lista se esistono tag. Dopo questa verifica si procede a ottenere la mappa (immagine) di localizzazione dell'oggetto nel mondo e le coordinate. Se l'immagine restituita dal server è vuota significa che il mondo selezionato non ha associata una mappa, se l'immagine è valida si mostra nella GUI e si aggiornano le coordinate dell'oggetto in relazione alle informazioni fornite dal server.

Sia questo task, che quello precedentemente descritto, sono schedulati da un timer che li esegue a intervalli definiti.

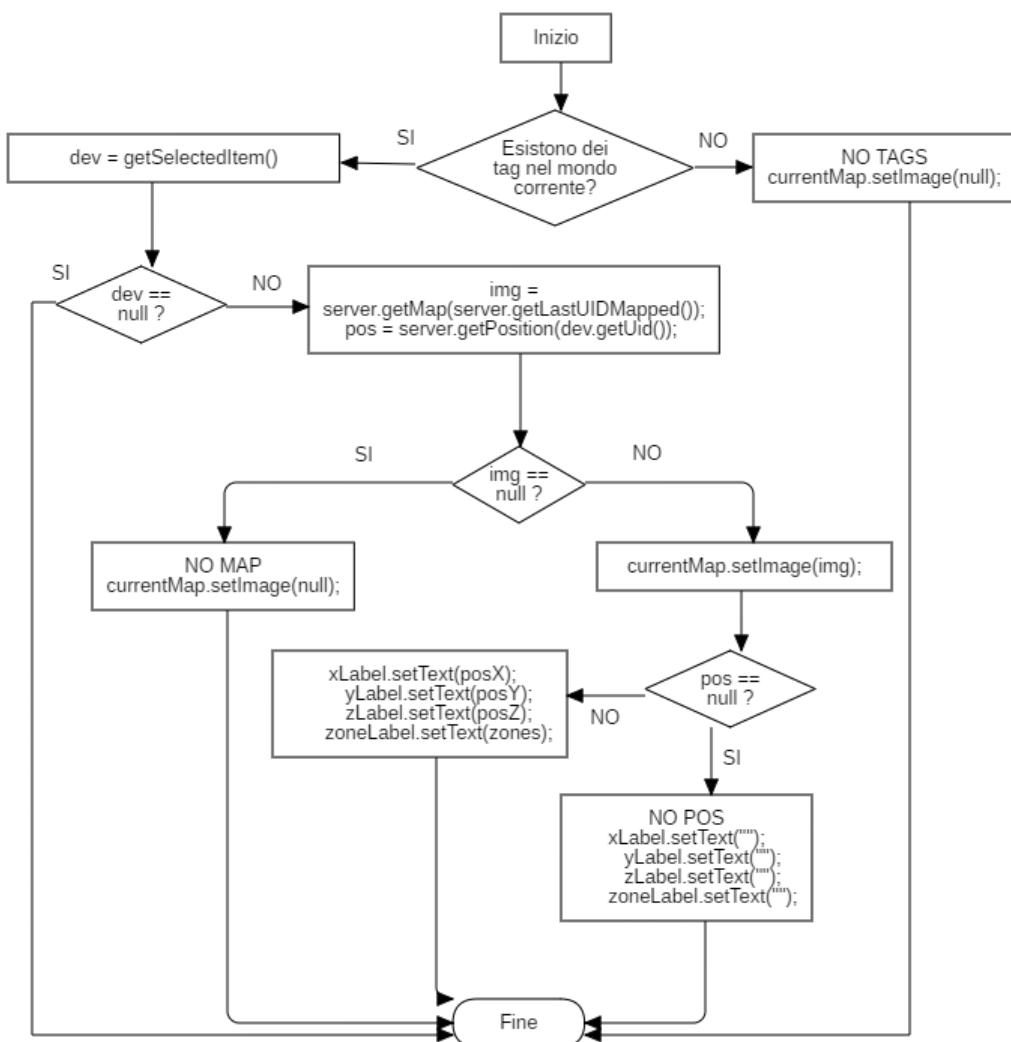


Figura 5.6: Diagramma UML flowchart per il Task che gestisce la mappa e le informazioni sulla posizione.

5.3 Gestione delle telecamere

Durante la ricerca della giusta sintassi per i comandi da inviare alle telecamere, è risultato evidente che questi variano notevolmente in base alla marca e al modello delle telecamere, per questo si è optato per l'utilizzo di una classe astratta Camera, in modo da poter modellare il concetto più generale di videocamera che verrà poi implementato a seconda delle caratteristiche e funzionalità specifiche di ogni modello, in questo caso AXISCamera e SV3CCamera sono le classi che estendono la classe astratta Camera. Inoltre in questo modo è possibile anche utilizzare altre librerie, alternative a OpenCV, per interfacciarsi con le videocamere, senza dover modificare parti di codice che non riguardano direttamente questo ambito.

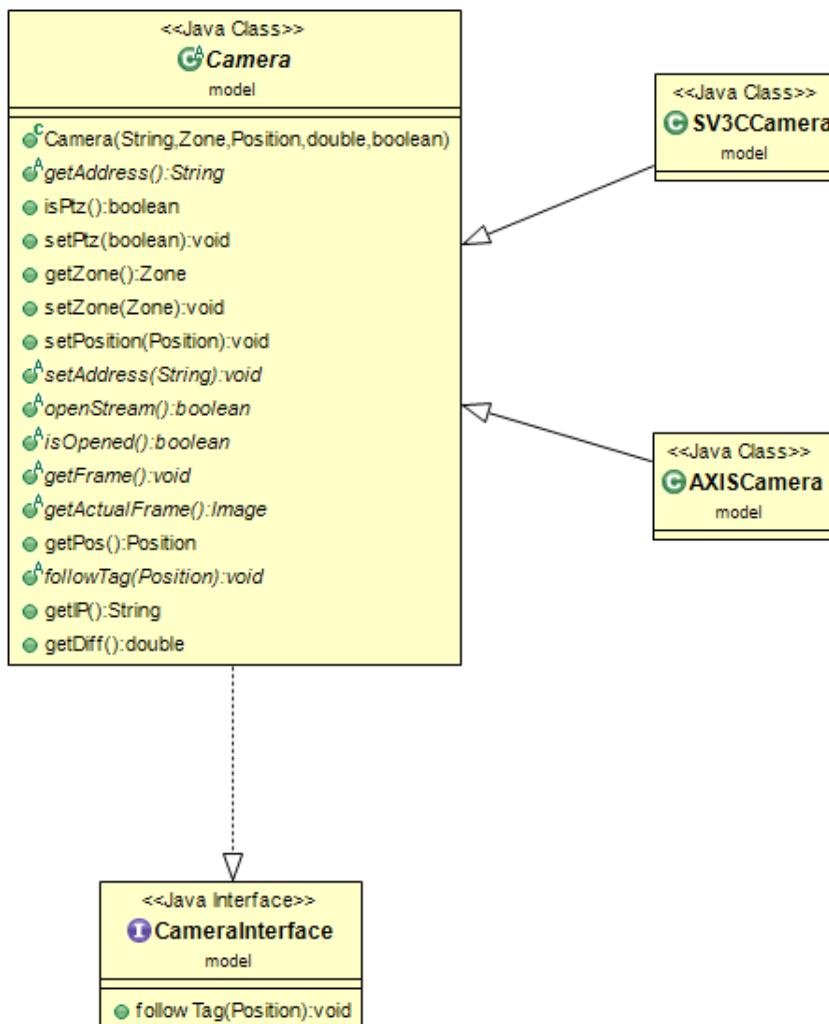


Figura 5.7: Implementazione della classe astratta Camera.

5.4 Sviluppo comandi telecamera

Si fornisce ora un livello di astrazione geometrico per il movimento delle telecamere PTZ analizzando nel dettaglio i calcoli effettuati. Successivamente si fornisce l'implementazione in Java per il problema dato e l'analisi dei casi particolari.

5.4.1 Calcolo PAN

Analizzando la documentazione della telecamera PTZ che è stata installata¹, si conosce l'angolo massimo e minimo che il PAN - ovvero il movimento orizzontale della lente - può assumere, in questo caso da 179° a -179° . Conoscendo le coordinate (x, y, z) del tag e della posizione in cui è installata la telecamera, si può rappresentare il problema secondo lo schema in figura 5.8. Inoltre considerando anche l'angolo α che potrebbe esistere tra il sistema di riferimento delle telecamera (in arancione) e quello della stanza (in blu) nella figura 5.8, ricaviamo l'angolo ϕ , il PAN, che successivamente si invia alla telecamera in modo che questa ruoti orizzontalmente rispetto al piano della stanza. Considerando:

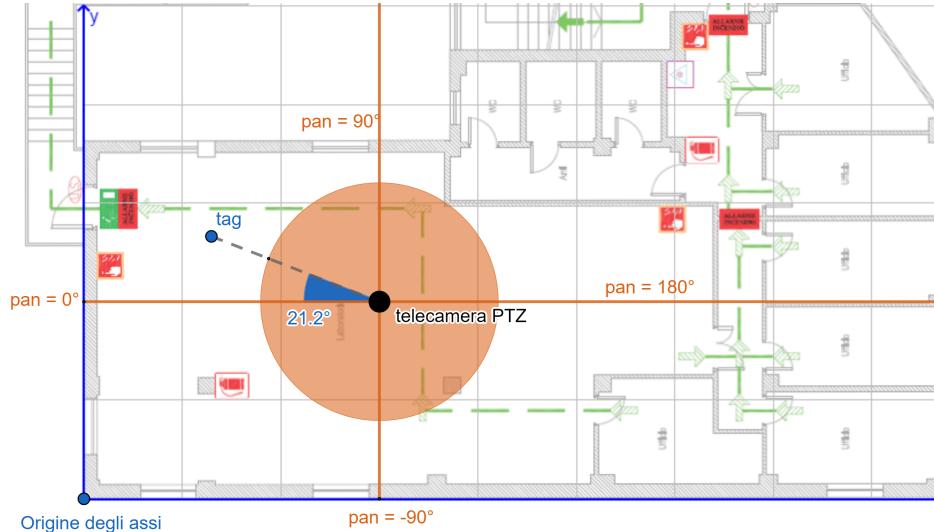


Figura 5.8: Rappresentazione del calcolo del PAN della telecamera

rando anche l'angolo α che potrebbe esistere tra il sistema di riferimento delle telecamera (in arancione) e quello della stanza (in blu) nella figura 5.8, ricaviamo l'angolo ϕ , il PAN, che successivamente si invia alla telecamera in modo che questa ruoti orizzontalmente rispetto al piano della stanza. Considerando:

- $(x_{cam}, y_{cam}, z_{cam})$, coordinate della posizione in cui è installata la telecamera.
- (x, y, z) , coordinate della posizione in cui si trova il tag.
- ϕ , PAN (in gradi)
- α , angolo tra il sistema della telecamera e il sistema della stanza, espresso in gradi da 0° a 360° seguendo il sistema di riferimento della videocamera.

Per calcolare l'angolo si utilizza la funzione `java.lang.Math.atan2(double y, double x)` messa a disposizione dalla libreria Math di Java. Questa funzione può essere definita nei termini della funzione arcotangente secondo quanto segue:

¹<https://www.axis.com/products/axis-m5054/>

$$\arctan2(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{se } x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{se } x < 0 \wedge y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{se } x < 0 \wedge y < 0 \\ +\frac{\pi}{2} & \text{se } x = 0 \wedge y > 0 \\ -\frac{\pi}{2} & \text{se } x = 0 \wedge y < 0 \\ \text{non definita} & \text{se } x = 0 \wedge y = 0 \end{cases}$$

Quindi si può calcolare il valore dell'angolo ϕ come:

$$\phi = \arctan2(y - y_{cam}, x - x_{cam}) + \alpha$$

Determinare il segno dell'angolo

Nella funzione arcotangente2, gli angoli con segno positivo si trovano nei due quadranti superiori, come mostrato in figura 5.9, mentre il sistema di riferimento della videocamera, essendo montata al soffitto della stanza, nei due quadranti superiori ha angoli con segno negativo. Considerando quindi che i due sistemi di riferimento sono invertiti rispetto all'asse x, si può utilizzare la funzione atan2 per ricavare l'angolo tra il tag e la videocamera, ma il segno è sempre invertito.

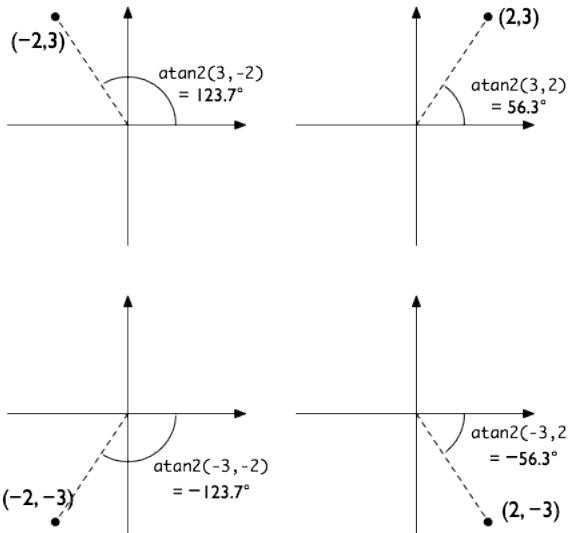


Figura 5.9: Rappresentazione della funzione atan2.

Casi particolari

Sommando α , che va da 0° a 360° , può accadere che l'angolo risultante cada fuori dall'intervallo accettato dalla telecamera. Quindi in questo caso è necessario rimappare il valore dentro l'intervallo di angoli disponibili.

Implementazione del metodo

```
// Utils.java
/**
 * Method for retrieving the PAN angle
 *
 * @param tagPos
 *         device position (x, y, z)
 * @param cameraPos
 *         camera position (x, y, z)
 * @param differenceAngle
 *         angle in degrees between camera's reference system and
 *         world's reference system
 * @return
 *         PAN angle in degrees
 */
public static double getPANAngle(Position tagPos, Position cameraPos,
        double differenceAngle) {
    double angle = 0.0;
    double X = tagPos.getX() - cameraPos.getX();
    double Y = tagPos.getY() - cameraPos.getY();
    if(!(X == 0 && Y == 0)) {
        angle = Math.toDegrees(Math.atan2(Y,X));
    }
    angle = -angle;
    angle += differenceAngle;
    if (angle > 180) {
        angle -= 360;
    } else if (angle < -180) {
        angle += 360;
    }
    return angle;
}
```

5.4.2 Calcolo TILT

Analizzando di nuovo la documentazione della telecamera PTZ², si conosce l'angolo massimo e minimo che il TILT può assumere, in questo caso da 0° a -90°. Conoscendo le coordinate (x, y, z) del tag e della posizione in cui è installata la telecamera, si può rappresentare il problema secondo lo schema in figura 5.10.

Considerando:

- $(x_{cam}, y_{cam}, z_{cam})$, coordinate della posizione in cui è installata la telecamera.
- (x, y, z) , coordinate della posizione in cui si trova il tag.
- θ , TILT (in gradi)

²<https://www.axis.com/products/axis-m5054/>

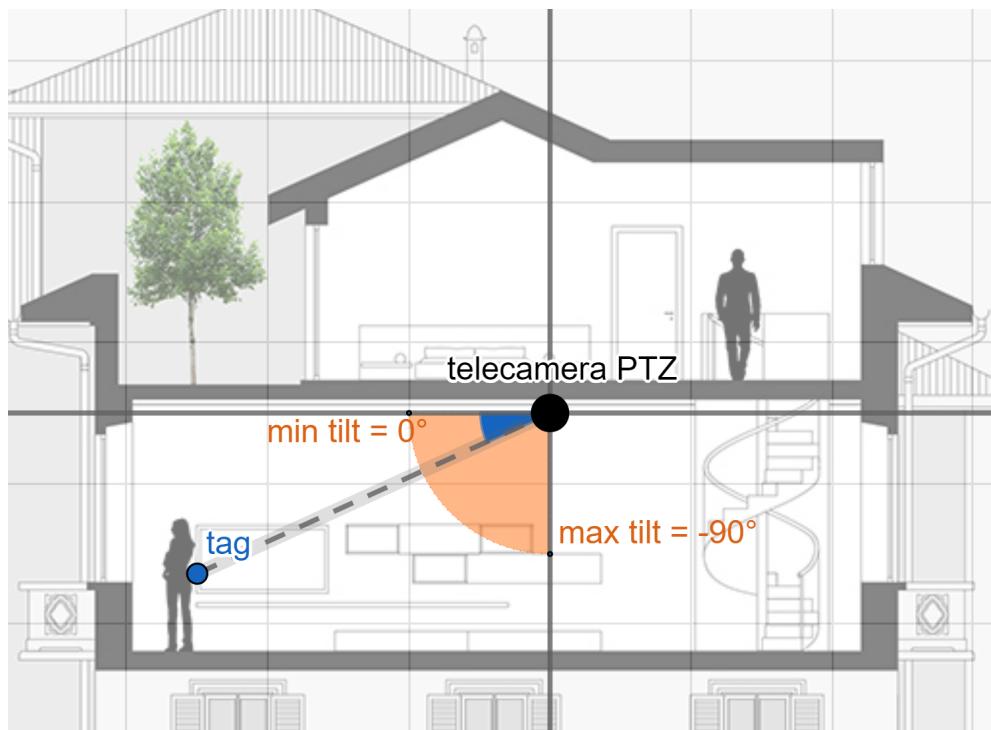


Figura 5.10: Rappresentazione del calcolo del TILT della telecamera.

Si può calcolare il TILT θ come:

$$\theta = \arctan \left(\frac{|z - z_{cam}|}{\sqrt{(x - x_{cam})^2 + (y - y_{cam})^2}} \right)$$

Determinare il segno dell'angolo

Il segno dell'angolo è sempre negativo perché il range di valori che la telecamera assume per quanto riguarda il TILT va da 0° a -90° .

Casi particolari

Il caso particolare che si prende in considerazione è di nuovo la divisione per 0. Questo accade in un solo caso, ovvero quando $(x - x_{cam})^2 + (y - y_{cam})^2 = 0$, se questo è soddisfatto allora si è erti che l'angolo $\theta = -90^\circ$.

Implementazione del metodo

```
// Utils.java

/**
 * Method for retrieving the TILT angle
 *
 * @param tagPos
 *         device position (x, y, z)
 * @param cameraPos
 *         camera position (x, y, z)
```

```

* @return
*          TILT angle in degrees
*/
public static double getTILTAngle(Position tagPos, Position
    cameraPos) {
    double angle = 0.0;
    double d = Math
        .sqrt((Math.pow(cameraPos.getX() - tagPos.getX(), 2) +
            Math.pow(cameraPos.getY() - tagPos.getY(), 2)));
    if (d == 0) {
        angle = 90.0;
    } else {
        double distanceZ = Math.abs(cameraPos.getZ() - tagPos.getZ());
        angle = Math.toDegrees(Math.atan(distanceZ / d));
    }
    return -angle;
}

```

5.4.3 Invio comando

Le videocamere AXIS che sono state utilizzate durante lo sviluppo del progetto, hanno un comando preciso per impostare PAN e TILT. Consultando la documentazione AXIS riguardante la videocamera installata, si è individuato questo comando per inviare i dati rispetto all'angolo assoluto della telecamera per inquadrare l'oggetto:

`http://serveraddr/axis-cgi/com/ptz.cgi?pan=x&tilt=z`

Quando il tag si trova nella zona in cui è attiva questa telecamera, il comando viene inviato a intervalli definiti di tempo.

5.5 Integrazione con Sequitur

Per quanto riguarda l'integrazione con il software esistente, si utilizzano le REST API messe a disposizione da Sequitur Manager. Nello specifico di seguito vengono elencate le API utilizzate e la funzionalità che mettono a disposizione del nuovo sistema.

5.5.1 Sequitur REST API

- /api/sys/environment

Funzione: ottenere lista dei mondi.

La richiesta al server dell'elenco dei mondi è fondamentale per il corretto funzionamento del software. Infatti questa è la prima richiesta che viene fatta durante l'inizializzazione dell'interfaccia. La lista dei mondi disponibili non può mai essere vuota, e questo è garantito dalla presenza di un

mondo di default con wid=0. Una volta ottenuta la lista dei mondi, questa viene resa disponibile all'utente, il quale può selezionare il mondo di cui si vogliono localizzare i device. In base alla selezione del mondo la lista dei device viene modificata dinamicamente e mostrata in una tabella, specificando l'uid e l'alias. Nel caso in cui la lista di device sia vuota, è possibile selezionare un altro mondo. All'avvio dell'applicazione si seleziona in automatico il primo mondo della lista e il primo device.

- /api/dev/group/wid

Funzione: ottenere la lista di device appartenenti a un dato gruppo (group) localizzato in un dato mondo (wid).

Questa richiesta al server viene fatta per ogni mondo presente nella lista dei mondi, serve ad ottenere la lista di device con gruppo = alltags, ovvero si includono sia i tag che i mini-tag. Questa lista viene mostrata all'utente il quale può selezionare uno degli elementi per ottenere informazioni e seguire l'oggetto con le telecamere.

- /api/dev/img/uid/uid

Funzione: ottenere la mappa corrispondente alla localizzazione di un dato device (uid). Questa richiesta viene inoltrata al server per poter ottenere un'immagine che contenga la mappa del mondo corrente e la localizzazione sulla mappa di un determinato device. Questa mappa viene mostrata nell'interfaccia quando l'utente seleziona un device. La mappa è disponibile anche quando nessuna telecamera è attiva nel mondo corrente, infatti la localizzazione nel mondo non dipende dalla presenza delle telecamere, ma è gestita dal software Sequitur.

- /api/dev/uid/uid

Funzione: ottenere informazioni su un dato device (uid).

Questa richiesta è la più frequente e viene utilizzata per mantenere aggiornate le informazioni riguardo la posizione di un device. E' possibile conoscere le coordinate del device (x, y, z) e la zona in cui si trova. Queste informazioni vengono mostrate in real-time all'utente e vengono utilizzate dal software per accedere allo stream video della telecamera presente nella zona in cui si trova l'oggetto.

- /api/zone/zoneSet

Funzione: ottenere informazioni su un dato set di zone.

Nelle impostazioni delle videocamere, l'utente deve specificare in quale zona ogni videocamera è attiva. La scelta della zona viene effettuata attraverso una lista di zone disponibili.

5.5.2 JSON Deserialization

Le REST API di Sequitur restituiscono in JSON le informazioni richieste. In riferimento alle richieste al server appena gestite, analizziamo come nello specifico i device, i mondi, le zone e le posizioni restituite in JSON del server diventano oggetti delle classi Device, World, Position e Zone nel software sviluppato.

Questa operazione viene detta *Deserialization*, consiste nel convertire un oggetto JSON in un oggetto Java. La libreria che abbiamo scelto di utilizzare è Gson di Google, che permette di utilizzare un metodo specifico per questa operazione. Come si legge nella documentazione della libreria [12], è necessario creare una classe con campi privati per i quali non è necessario indicare se va incluso o meno nella deserializzazione, questo perchè tutti i campi della classe sono inclusi di default. I casi null vengono gestiti dalla libreria, ad esempio quando c'è una entry mancante nel risultato JSON dato dal server il campo corrispondente della classe viene messo a null.

Ad esempio la richiesta al server della posizione di un device restituisce la struttura JSON seguente:

```
{
  "device": [
    {
      "valid": true,
      "uid": "EEEEEEEEEE0000",
      "pos": {
        "elevation": 0,
        "sx": 0,
        "sy": 0,
        "sz": 0,
        "latitude": 0,
        "zones": [
          "laboratorio"
        ],
        "wid": 0,
        "x": 9.153,
        "y": 7.354,
        "z": 1.012,
        "validity": true,
        "longitude": 0,
        "ts": 1567698752856
      }
    }
  ]
}
```

La classe Position.java contiene i seguenti campi:

```
//Position.java

private double elevation;
private double wid;
private double sx;
private double sy;
private double sz;
private double latitude;
private double x;
private double y;
private double z;
```

```
private Boolean validity;
private float longitude;
private float ts;
private List<String> zones;
```

La deserializzazione è necessaria ad ogni richiesta della posizione in cui si trova il tag di cui l'utente richiede le informazioni. Con il metodo `getPosition()` si attua la deserializzazione sfruttando le funzioni della libreria Gson.

```
public Position getPosition(String devUID) throws IOException {
    Gson gson = new Gson();
    ...

    JsonObject jsonObject = gson.fromJson(reader, JsonObject.class);
    JSONArray jsonArray = (JSONArray) jsonObject.get("device");
    ...
    Position pos =
        gson.fromJson(jsonArray.get(i).getAsJsonObject().get("pos") .
                      toString(), Position.class);
    ...
}
```

Questo approccio viene utilizzato anche per le altre richieste al server descritte in precedenza. In questo modo il software ha a disposizione tutte le informazioni necessarie per mostrare all'utente la situazione in real-time del tag e, attraverso l'utilizzo della classe `Camera.java`, per ottenere lo stream necessario a seguire l'oggetto nelle varie zone in cui si trova.

5.6 Problemi incontrati e soluzioni

Nella realizzazione dell'attività di tesi si sono riscontrati i seguenti problemi:

1. Difficoltà nel mantenere una coerenza continua tra i dati e l'interfaccia utente. La GUI dipende in modo considerevole dalla concorrenza, ma il multi-thread di Java richiede la ripetizione di parti di codice quasi identiche per poter funzionare correttamente.
2. Un altro problema incontrato è riferito all'acquisizione dello stream dalle videocamere: inizialmente il modo più opportuno per ottenere l'immagine corrente dalla telecamera attiva era quello di chiamare il metodo `VideoCapture.grab()` e successivamente `VideoCapture.retrieve(Mat)` solo nel momento in cui il controller della finestra principale individuava una determinata videocamera come attiva. Questo determinava uno strano comportamento: un accumulo di frame durante l'inattività della videocamera, che risultava in una carrellata di frame vecchi alla riattivazione della videocamera.
3. Il metodo `VideoCapture.open()`, della libreria OpenCV, impiega diversi secondi a terminare quindi non può essere chiamato nel momento in cui l'utente seleziona un tag per poter seguire le sue informazioni in tempo reale.

Questo metodo inoltre ha un timeout di svariati secondi il che rende ancora meno gradevole la gestione di un indirizzo IP scorretto nell'inserimento di una nuova videocamera. Quando l'indirizzo IP della videocamera è errato il metodo impiega più di 30 secondi a ritornare mostrando l'errore.

Riguardo ai problemi appena elencati si sono trovate le seguenti soluzioni:

1. Utilizzo della libreria JavaFX. Con l'introduzione di questa libreria nel progetto, si è ottenuto un codice più snello, semplicità nella gestione della coerenza tra i dati e la GUI (utilizzo di bind), sviluppo più rapido della parte grafica attraverso l'utilizzo di SceneBuilder. Inoltre l'utilizzo di file FXML per definire l'interfaccia dell'applicazione, ha reso semplice la separazione della view dal resto dal codice, rendendo il progetto più flessibile in sviluppi futuri e rispettando alla lettera il pattern MVC. La gestione del multi-thread e della concorrenza risulta più snella e senza ripetizioni di codice.
2. Come soluzione a questo problema si è creato un campo di tipo Image nella classe Camera, la sua utilità è quella di tenere in memoria l'ultimo frame acquisito attraverso l'utilizzo di un thread separato, indipendentemente dal fatto che venga mostrato nella GUI oppure no. In questo modo non c'è accumulo di frame nei metodi di OpenCV.
3. Essendo abbastanza difficile modificare il timeout del metodo della libreria OpenCV, la soluzione scelta è quella di eseguire in background le operazioni di apertura delle videocamere e mantenere aggiornato l'utente sull'andamento dell'operazione.

Capitolo 6

Conclusioni e sviluppi futuri

In questa tesi, si è realizzato un software per il controllo automatico di telecamere fisse e PTZ per videosorveglianza, mediante interfacciamento con sistema di posizionamento RTLS basato su tecnologia UWB. Tale software permette di monitorare con telecamere le posizioni di un utente che si muova in ambiente indoor munito di tag, così da operare una selezione delle camere da utilizzare in base all'ambiente ove si trova il soggetto taggato e poter seguire i movimenti attraverso camere motorizzate. Il lavoro svolto durante la tesi mette a disposizione dell'azienda uno strumento interessante sia a livello progettuale, che come dimostratore per possibili clienti della stessa, che abbiano la necessità di monitorare oggetti o persone all'interno di un edificio.

6.1 Sviluppi futuri

Per migliorare il sistema esistono una serie di possibili sviluppi futuri che vengono elencati di seguito.

Stabilità stream videocamera PTZ

Un primo miglioramento al sistema è sicuramente trovare un modo per rendere più fluido e stabile il movimento della videocamera PTZ che a volte risulta a scatti. Durante lo sviluppo del progetto si è tentato di raggiungere questo obiettivo effettuando una media aritmetica sulle ultime posizioni del device e calcolando il movimento della videocamera in base a questo valore. Naturalmente questo approccio genera un leggero ritardo nei movimenti della videocamera rispetto all'oggetto. Un altro modo è quello di applicare la ponderazione esponenziale che tenga conto dell'ultimo rilevamento della posizione. Un approccio differente, invece, sarebbe quello di seguire l'oggetto inviando comandi incrementali rispetto alla posizione attuale della videocamera, anziché comandi assoluti, come attualmente è implementata l'applicazione.

Seguire più tag contemporaneamente

Un possibile sviluppo alternativo a quello attuale, potrebbe essere quello di aprire una nuova finestra per ogni tag che l'utente seleziona in modo da poter seguire più oggetti contemporaneamente.

Sviluppo comando ZOOM

Le telecamere PTZ mettono a disposizione anche la possibilità di impostare lo zoom della lente, questo aspetto si potrebbe sfruttare per attivare lo zoom su oggetti che si trovano lontani dalla telecamera, aumentando e diminuendo questo valore in relazione alla vicinanza dell'oggetto.

Migliorare integrazione con Sequitur

E' previsto che vengano integrati maggiormente i due sistemi in modo che possano trarre beneficio l'uno dall'altro, innanzitutto inserendo dei messaggi di stato prodotti dal software sviluppato e destinati a Sequitur, con informazioni sulla connessione e sullo stato del tracking di un determinato tag.

6.2 Considerazioni finali

Lo sviluppo di questo progetto aveva come scopo quello di realizzare un sistema di videosorveglianza integrato a Sequitur per l'azienda UNISET. Realizzare questo sistema ha richiesto un'analisi e comprensione del sistema di localizzazione esistente, per poter sfruttare ed elaborare le informazioni che questo fornisce all'utente. Ci sono state diverse occasioni di confronto e collaborazione con gli sviluppatori di Sequitur, i quali mi hanno guidato e aiutato nella realizzazione del nuovo software, notando difetti e possibili miglioramenti con occhio esperto e un atteggiamento costruttivo nei miei confronti. Per questo ritengo questa esperienza, oltre che utile a livello professionale, decisamente positiva dal punto di vista personale. Realizzare inoltre un prodotto interessante per un'azienda è soddisfacente e rende il lavoro di progettazione e implementazione impegnativo e rigoroso.

Ringraziamenti

Il primo ringraziamento va sicuramente ai colleghi dell'azienda UNISET: Michele l'Enciclopedia, memorabili le sue gesta nel risolvere gli errori di importazione di OpenCV su Eclipse, Federico che mi ha ricordato che l'utente non può modificare il sorgente inserendo l'indirizzo del server ogni volta, Nicolò massimo esperto di L^AT_EX, oltre ad aver minuziosamente corretto questa tesi, ci ha allegramente intrattenuto con la lettura del manuale dei dispositivi Samsung, Matteo e Panco con il loro elicottero con pale taglienti hanno reso il laboratorio di via Venezia un posto più sicuro, prof. Tartagni e l'investimento in telecamere svedesi ha dato la svolta ricca al progetto. Sono felice di aver lavorato con persone qualificate e professionali al mio fianco, grazie.

Un grazie ai miei compagni di viaggio: Rosso, Filo, Luchino, Nardo, Sambu, Galle, Coro, Ale, la mia coinqui Giuli, Bro e Ciora. E' stata una bella avventura ripulire casa ogni volta che ci avete messo piede, ma la verità è che non sarei qua se avessi fatto questo percorso da sola.

Grazie ai miei genitori, mamma che piangerà per due settimane e babbo lo stesso, mio fratello che mi prenderà in giro perchè lui ha due lodi. Vi voglio bene.

Bibliografia

- [1] Charles A. Fowler, J. N. Entzminger, and J. Corum. Assessment of ultra-wideband (uwb) technology. *IEEE Aerospace and Electronic Systems Magazine*, 5(11):45–49, Nov 1990.
- [2] Federal Communications Commission (FCC). First report and order, revision of part 15 of the commission’s rules regarding ultra-wideband transmission systems. http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-02-48A1.pdf, 2002.
- [3] Roberto Verdone, Davide Dardari, Gianluca Mazzini, and Andrea Conti. *Wireless Sensor and Actuator Networks: Technologies, Analysis and Design*. Academic Press, Jan 2008.
- [4] Luigi Battezzati and J. L. Hygounet. *RFID. Identificazione automatica a radiofrequenza. Tecnologie e applicazioni*. Hoepli, Sep 2003.
- [5] Gerald F Ross. Time limited impulse response antenna, June 22 1971. US Patent 3,587,107.
- [6] Gerald F Ross and Joseph D De Lorenzo. Generator for short-duration high-frequency pulse signals, October 12 1971. US Patent 3,612,899.
- [7] Gerald F Ross and David Lamensdorf. Balanced radiator system, April 25 1972. US Patent 3,659,203.
- [8] Horea Bendea, Alberto Cina, Marco Piras, Gianluca Marucco, and Paolo Mulassano. Posizionamento indoor con ricevitori a basso costo: quali prestazioni. In *14a Conferenza nazionale ASITA*, pages 1–5, 2010.
- [9] SJ Ingram, D Harmer, and M Quinlan. Ultrawideband indoor positioning systems and their use in emergencies. In *PLANS 2004. Position Location and Navigation Symposium (IEEE Cat. No. 04CH37556)*, pages 706–715. IEEE, 2004.
- [10] UNISET. *Sequitur Reference Manual r1v0*, 2019.
- [11] Gianluca Boiero, Massimo Colonna, and Dario Parata. La localizzazione nei sistemi radiomobili. *Notiziario Tecnico Telecom Italia, Anno 15 Numero 3*, 2014.
- [12] Google. Gson. <https://github.com/google/gson/>, Sep 2019. [Online; accessed 5. Sep. 2019].