



Università degli Studi di Salerno
Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

Progettazione di un Sistema Smart Lighting Posizionale con Integrazione KNX e Tracking Video

Relatori

Prof. Andrea Francesco Abate

Prof. Fabio Narducci

Candidato

Mattia Giuseppe Giella

Matricola: 0512109805

Abstract

KNX è uno standard mondiale per la domotica che facilita la gestione e la comunicazione tra dispositivi marchiati KNX, indipendentemente dal produttore, utilizzando il suo protocollo omonimo e codificando i dati utili secondo i tipi di dati standardizzati. Il sistema sviluppato si basa su un'installazione KNX che gestisce l'illuminazione nel laboratorio CASALab, grazie alla configurazione effettuata con il software ETS. Il sistema proposto può comunicare con l'interfaccia dell'installazione, tramite telegrammi KNX, permettendo così un'illuminazione smart. Tramite l'implementazione del modello di Object Detection YOLOv8, utilizzato ai fini del Video Tracking, che opera sul flusso video ottenuto da una telecamera IP PTZ, sarà possibile identificare la posizione all'interno dell'ufficio di un utente, attraverso una divisione in zone, e gestire l'illuminazione dinamicamente, rendendo l'ambiente in grado di adattarsi all'esigenze dell'utente in modo automatico. Attraverso questo sistema si punta all'impiantare una soluzione per l'integrazione di un sistema di Smart Lighting posizionale, pratica e accessibile grazie a un interfaccia grafica. In questa tesi saranno esplorati i temi dello Smart Lighting, considerando i vantaggi che questo porta in un ambiente in cui è utilizzato, e il problema del Object Tracking, facendo riferimento allo stato dell'arte e ai vari tipi di modelli esistenti. Sarà poi fornito qualche dettaglio sul funzionamento e le capacità di un sistema KNX, e dei vantaggi nell'utilizzo del modello YOLO. Infine sarà illustrato il sistema proposto, fornendo dettagli sull'installazione KNX, sulla telecamera utilizzata e sull'applicazione software sviluppata.

Indice

Abstract	I
1 Da KNX allo Smart Lighting	1
1.1 Installazione KNX	1
1.1.1 Cos'è KNX?	1
1.1.2 Caso di Studio: il Laboratorio CASALab	1
1.2 Telecamere disponibili	2
1.3 Obiettivo della tesi: Smart Lighting	3
1.4 Struttura della tesi	3
2 Smart Lighting e Video Tracking	5
2.1 Smart Lighting	5
2.1.1 Cosa si intende per Smart Lighting	5
2.1.2 I vantaggi di un sistema di Smart Lighting	5
2.2 Architettura del sistema	6
2.2.1 Dispositivi	6
2.2.2 Network	7
2.2.3 Un esempio pratico, Philips Hue	8
2.2.4 Controlli e Algoritmi	9
2.3 Video Tracking	9
2.3.1 Object Tracking	9
2.4 Tipologie di Tracker	10
2.4.1 Metodi basati sulle caratteristiche	10
2.4.2 Metodi basati sulla segmentazione	11
2.4.3 Metodi basati sulla stima	12
2.4.4 Metodi basati sull'aspetto	12
2.4.5 Metodi basati sull'apprendimento	13
2.5 CNN e YOLO	15
3 Componenti del Sistema	17
3.1 KNX	17
3.1.1 Architettura KNX	17
3.1.2 Modelli di applicazione	20

3.1.3	Interoperabilità	21
3.1.4	Modalità di configurazione	21
3.1.5	Gestione della rete	21
3.1.6	Sistema di comunicazione	22
3.1.7	Dispositivi e Profili	22
3.1.8	ETS	22
3.2	YOLO: You Only Look Once	22
3.2.1	Metodi a una fase e due fasi	23
3.2.2	Funzionamento del modello	23
3.2.3	Punti di forza e problemi di YOLOv1	24
3.2.4	Evoluzione negli anni	24
3.2.5	YOLO v8	25
4	Analisi del Sistema Implementato	27
4.1	Topologia del sistema	27
4.2	Installazione KNX CASALab	27
4.2.1	EIBPORT	27
4.2.2	Dispositivi dell'installazione	28
4.2.3	Dali-Gateway	30
4.2.4	Indirizzi di gruppo	31
4.2.5	Progetto ETS	32
4.3	Telecamere	34
4.3.1	Dahua IPC-HFW4830E-S	35
4.3.2	Kandao Obsidian R	36
4.3.3	PTZIP2M4XWS	37
4.3.4	Telecamera scelta per l'integrazione con il sistema	38
4.4	Sistema proposto	38
4.5	RTSP	38
4.5.1	Librerie utilizzate	39
4.6	Implementazione del sistema	40
4.6.1	Controllo della telecamera	41
4.6.2	Tracciamento	43
4.6.3	Interfaccia utente	45
4.6.4	Gestione delle zone	49
4.6.5	Gestione dell'illuminazione	54
5	Conclusione	57

Capitolo 1

Da KNX allo Smart Lighting

1.1 Installazione KNX

1.1.1 Cos'è KNX?

KNX rappresenta uno standard globale nel settore dell'automazione degli edifici agevolando la comunicazione istantanea tra dispositivi di vari produttori, a condizione che siano conformi allo standard.

Questa interoperabilità offre una notevole flessibilità nella selezione di dispositivi da integrare nel sistema, ampliando le opzioni di compatibilità [1].

Il nucleo del sistema KNX è costituito da un bus logico che funge da canale di comunicazione per lo scambio di pacchetti dati tra gli attuatori e i sensori. Questo bus supporta una comunicazione bidirezionale, consentendo agli attuatori di ricevere comandi dai sensori e viceversa. In aggiunta, il bus KNX facilita l'integrazione di diversi dispositivi all'interno di una singola rete, generando un ecosistema coeso e coordinato.

L'utilizzo di software di progettazione KNX, come ETS (Engineering Tool Software), semplifica la configurazione e il controllo centralizzato di tutti i dispositivi all'interno del sistema, agevolando un'implementazione efficiente e gestione ottimale.

1.1.2 Caso di Studio: il Laboratorio CASALab

L'illuminazione nel laboratorio è completamente integrata attraverso il protocollo KNX, offrendo un controllo avanzato e personalizzato tramite un pannello di display touch dedicato. Le luci sono organizzate in tre gruppi distinti, divisi in due ambienti:

Luci nella Sala Principale:

- Luci a Soffitto
- Luci Lateralì

Per entrambi questi gruppi è possibile gestire temperatura colore, luminosità, e l'accensione/spegnimento. Questa suddivisione consente un controllo dettagliato dell'illuminazione in diverse zone della sala principale.

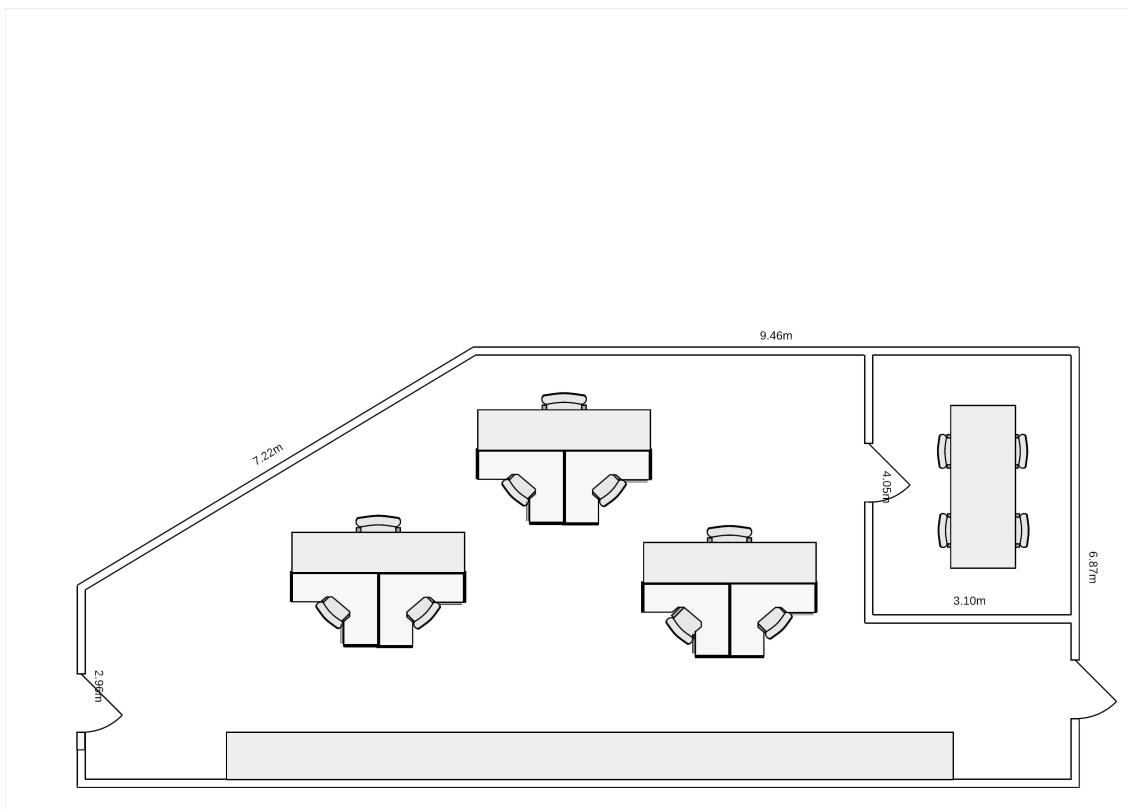


Figura 1.1: Laboratorio CASALab

Luci RGB dell’Ufficio:

Per questo gruppo è possibile selezionare il colore specifico e modificare la luminosità.

L’installazione è stata realizzata con il software di KNX, ETS (Engineering Tool Software), semplificando la connessione delle luci agli interruttori e al display. Questo approccio agevola l’integrazione delle varie funzionalità offerte dalle luci, garantendo un controllo efficace e centralizzato su tutto il sistema di illuminazione.

1.2 Telecamere disponibili

Il laboratorio è anche attrezzato con tre tipi di telecamere IP infrarossi, collegabili a un cavo di rete e provviste di IP, dalle quali è possibile ottenere il flusso video.

Dahua IPC-HFW4830E-S

- **Modello:** Telecamera bullet
- **Caratteristiche:** Lenti fisse con un campo visivo ristretto. Progettate per garantire sicurezza e monitoraggio in specifiche aree predefinite.

Kandao Obsidian R

- **Modello:** Telecamera 360°
- **Caratteristiche:** Offre una visione panoramica completa a 360°, permettendo di monitorare l'intero ambiente senza punti ciechi.

PTZIP2M4XWS

- **Modello:** Telecamera PTZ Wireless
- **Caratteristiche:** Consente movimenti controllati PAN (panoramica), TILT (inclinazione), e ZOOM, fornendo la flessibilità di regolare il campo visivo della telecamera in base alle esigenze specifiche.

1.3 Obiettivo della tesi: Smart Lighting

Il laboratorio dispone di tutte le caratteristiche necessarie per l'implementazione di un sistema di Smart Lighting.

L'introduzione di un tale sistema consentirà l'utilizzo dell'illuminazione in modo automatico, eliminando la necessità di accendere manualmente le luci tramite interruttori offrendo, inoltre, un miglioramento in termini di efficienza energetica.

L'installazione del sistema KNX sarà impiegata per gestire l'accensione delle luci in base alla posizione delle persone rilevate dalle telecamere grazie a un sistema di Video Tracking. Questa integrazione di tecnologie contribuirà a un utilizzo più efficiente dell'illuminazione.

1.4 Struttura della tesi

Nel contesto di questa tesi, verrà esplorato lo stato dell'arte relativo al problema dello Smart Lighting, concentrando sulla struttura di un sistema di questo tipo e sui suoi vantaggi. Sarà inoltre analizzato il Video Tracking, con riferimento ai vari modelli esistenti.

Successivamente, sarà presentata la scelta finale e le ragioni dietro di essa, mettendo in luce le caratteristiche distintive del modello selezionato. L'obiettivo è fornire una panoramica completa delle opzioni disponibili, evidenziando le motivazioni dietro le scelte effettuate per la soluzione proposta.

Si approfondirà il funzionamento del sistema KNX, illustrando l'interazione dei dispositivi di questo sistema. Sarà inoltre esaminato YOLO, il modello di Object Detection utilizzato, con dettagli sul suo funzionamento, sui punti di forza e sulle criticità, e mostrando la sua evoluzione nel tempo. Questo contribuirà a chiarire le caratteristiche e le potenzialità delle tecnologie coinvolte.

Infine, sarà descritta la configurazione dell'installazione KNX nel laboratorio, con riferimento al progetto ETS. Si analizzerà anche l'installazione delle tre telecamere proposte, esaminando le caratteristiche positive e negative che hanno influenzato la scelta di una di esse per l'in-

tegrazione nel sistema. La tesi si concluderà con un'analisi dettagliata del sistema proposto, focalizzandosi in particolare sul codice implementato.

Capitolo 2

Smart Lighting e Video Tracking

Nel prossimo capitolo, sarà esaminato il concetto di Smart Lighting, approfondendo i benefici associati a questa tecnologia e analizzando l'architettura di un sistema di illuminazione intelligente. Presenteremo un esempio concreto di un sistema completo disponibile sul mercato, il Philips Hue. Successivamente, sarà affrontato il tema del Video Tracking, esplorando diverse tipologie di tracker, da metodi classici a tecniche di apprendimento profondo per infine focalizzarsi sul modello specifico scelto.

2.1 Smart Lighting

2.1.1 Cosa si intende per Smart Lighting

Per Smart Lighting si intende un sistema di illuminazione intelligente, sia per ambienti interni che esterni, che va oltre il controllo convenzionale tramite interruttori. Questo sistema utilizza il feedback degli utenti e/o sensori per gestire non solo l'accensione e lo spegnimento delle luci, ma anche parametri più avanzati come luminosità, temperatura colore e persino il colore RGB della luce.

Con questo approccio, le luci possono essere automatizzate in modo da eliminare la necessità di interventi manuali. Questa automazione non solo ottimizza l'efficienza energetica, ma offre anche un elevato livello di personalizzazione e comfort nelle impostazioni luminose degli ambienti. L'integrazione di sensori e feedback degli utenti contribuisce a creare un ambiente illuminato più adattabile e intuitivo, rendendo l'esperienza luminosa più efficiente e personalizzata.

2.1.2 I vantaggi di un sistema di Smart Lighting

La preferenza per un sistema di illuminazione smart, oltre al risparmio energetico comprovato, che varia dal 17% al 94% in base a diversi fattori come controllo manuale, ottimizzazioni ambientali e comportamento dell'utente [2], è motivata dalla prospettiva di migliorare la qualità dell' illuminazione, aumentare la produttività e integrare funzionalità Human Centric.

Un'illuminazione Human Centric regola la luminosità in modo da esporre le persone alla giusta quantità di luce, in modo da rispettare il ciclo circadiano, garantendo un miglioramento

dell’attenzione, secrezioni ormonali, temperatura corporea e qualità del sonno. Questo approccio mira a prevenire problemi a lungo termine, quali obesità e diabete, e a preservare le normali funzioni fisiologiche del corpo [3]. Inoltre, la possibilità di adattarsi dinamicamente alle esigenze degli utenti rende i sistemi di Smart Lighting una scelta versatile e centrata sul benessere complessivo di coloro che occupano gli spazi illuminati.

2.2 Architettura del sistema

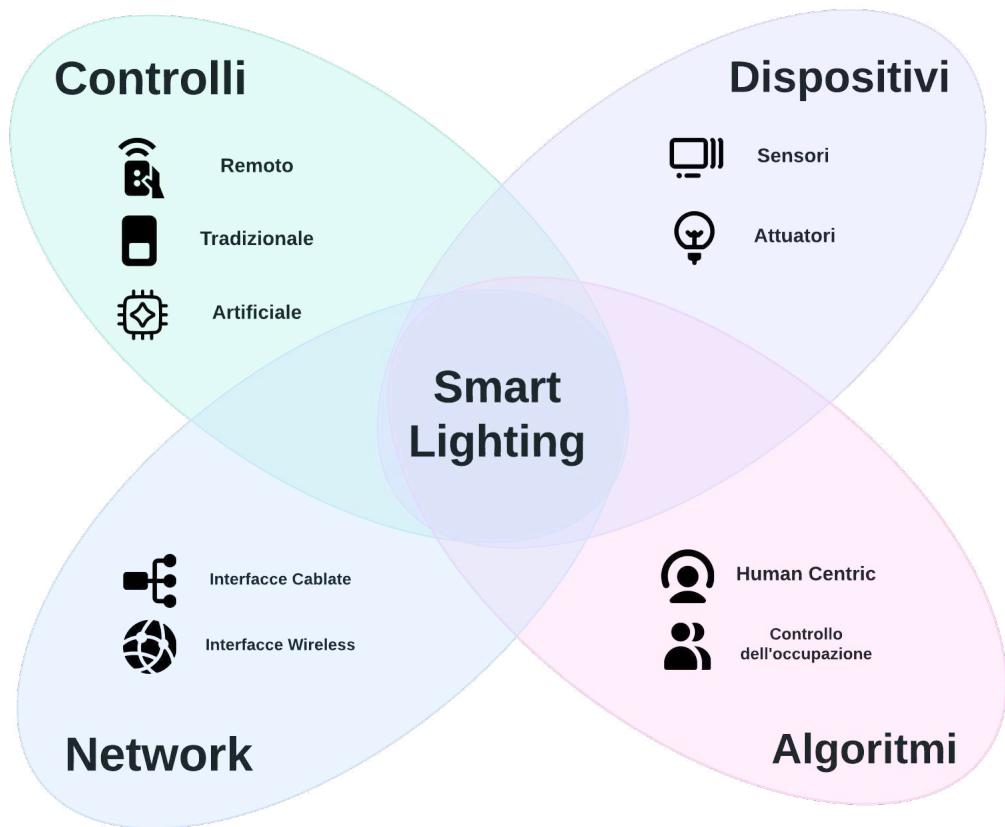


Figura 2.1: Componenti di un sistema Smart Lighting

Un sistema di Smart Lighting è costituito da dispositivi, che possono essere semplici attuatori collegati a sensori, oppure dispositivi dotati di entrambe le componenti con all’interno algoritmi specifici, che comunicano e si sincronizzano fra loro grazie a un network [4], come è possibile osservare dalla figura 2.1.

2.2.1 Dispositivi

Un sistema di illuminazione di questo tipo trae molto vantaggio dai diodi a emissione di luce(LED) grazie al grande controllo sulla luminosità, la velocità di accensione e spegnimento, oltre che alla durata e al risparmio energetico; in aggiunta alla tecnologia LED usata per gli attuatori, un sistema di Smart Lighting ha bisogno di sensori per dare la possibilità al sistema

di adattarsi all'ambiente circostante.

I sensori oltre al poter essere integrati all'interno di dispositivi che presentano attuatori grazie ad algoritmi che permettono il monitoraggio della quantità di luce od occupazione delle zone illuminate, possono essere anche dispositivi a se stanti come:

- **Sensori crepuscolari:** In grado di monitorare il livello di illuminazione nell'area circostante.
- **Sensori di colore RGB:** Utili a capire il grado di bianco nelle luci LED.
- **Sensori a infrarossi passivi:** Utilizzati per rilevare la presenza di persone in un'area.

Problemi dei sensori di occupazione

Una particolare attenzione va posta sui sensori a infrarossi passivi che presentano un livello di incertezza nella rilevazione a causa di un limitato campo visivo. Per sopperire a questa mancanza sono spesso introdotti timer di spegnimento, che limitano la possibile riduzione di spreco energetico in aree con uso sporadico e se il timer non è ben calibrato potrebbe addirittura portare all'aumento del consumo energetico. Un altro problema che potrebbe insorgere a causa della poca visibilità è la possibilità di non effettuare una rilevazione se l'area occupata è fuori dal campo visivo del sensore [5]. L'uso di videocamere con appositi algoritmi di tracking potrebbe essere una soluzione ai problemi riscontrati in questo tipo di sensori.

2.2.2 Network

Al fine di comunicare e sincronizzarsi, i dispositivi fanno parte di un network, questo può essere sia cablato (come KNX, DALI, Ethernet o cablaggio tradizionale) che wireless (come ZLL, Bluetooth, Wi-Fi) e contenere anche altri dispositivi oltre ai quelli che hanno solo il ruolo di illuminazione. Analizziamo nel dettaglio alcuni dei protocolli citati.

DALI (Digital Addressable Lighting Interface): Protocollo standard per il controllo delle luci digitali, è basato sull'uso di un bus logico 2.2 che permette ai dispositivi controllati di ricevere e mandare messaggi. [6]

ZLL (ZigBee Light Link): Profilo di applicazione pubblico basato sul protocollo wireless ZigBee PRO, possiede meccanismi di sicurezza e gestione del Network. ZLL specifica due tipi di dispositivi luci e controller. Tramite un dispositivo controller, chiamato initiator, è possibile aggiungere le luci alla rete, identificandole con un indirizzo [8].

KNX: Si basa su un'architettura decentralizzata. Per la trasmissione dei dati, il protocollo utilizza quattro tipi di mezzi differenti: coppia intrecciata (TP), linea di alimentazione (PL), radiofrequenza (RF) e protocollo Internet (IP), tra i quali i più diffusi sono KNX/IP e KN-TP. [9]

A oggi, inoltre, molti dispositivi sono pensati per essere integrati in sistemi IoT.

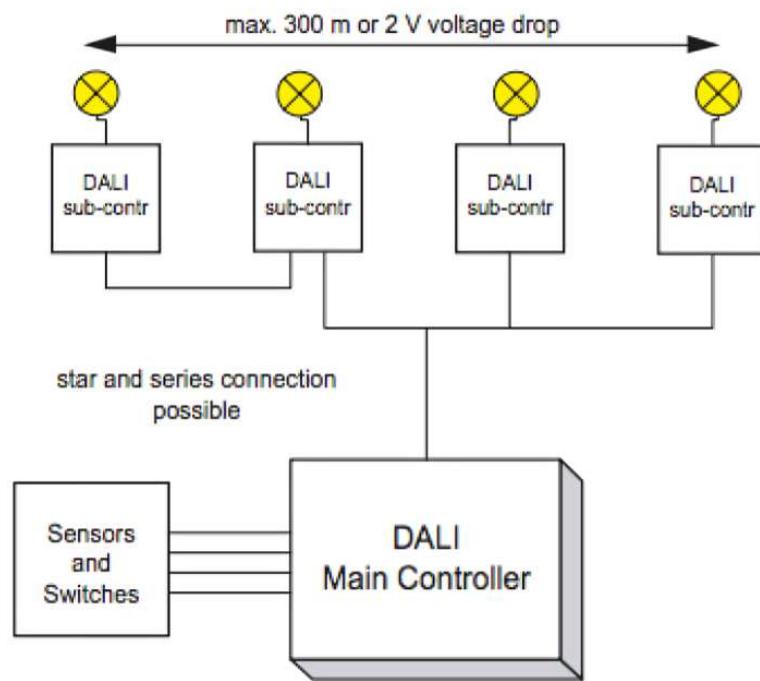


Figura 2.2: Bus DALI, immagine ottenuta da [7]

2.2.3 Un esempio pratico, Philips Hue

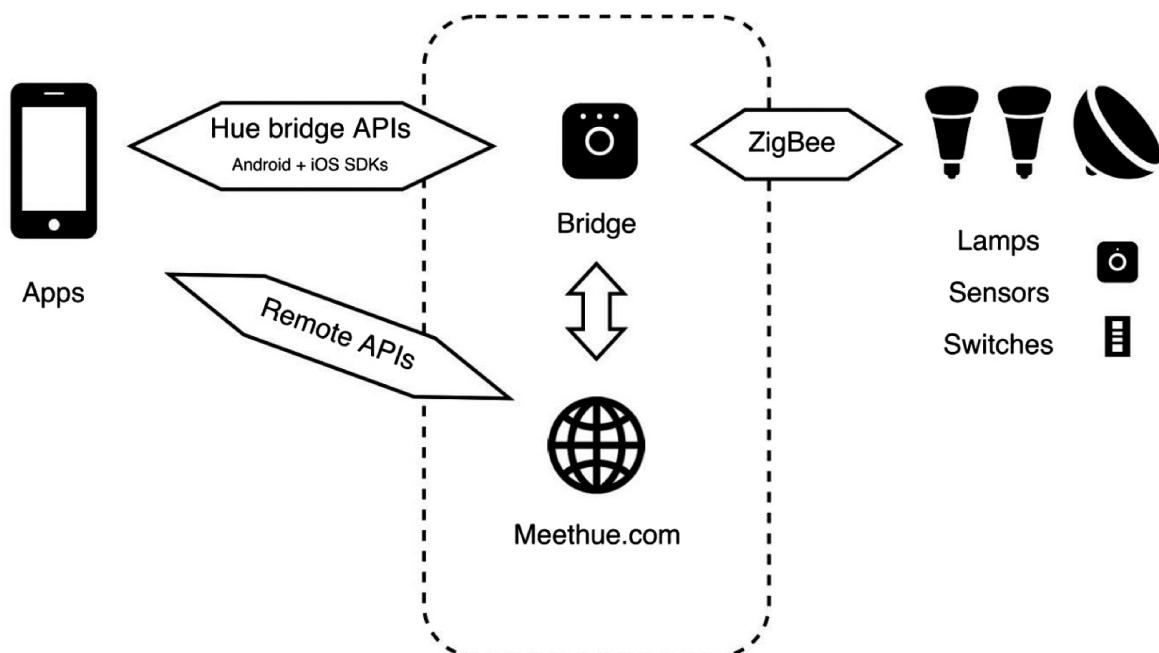


Figura 2.3: Architettura di un sistema Philips Hue, immagine ottenuta da [10]

La piattaforma Philips Hue è un sistema di illuminazione IoT facilmente acquistabile e configurabile, il sistema è composto da lampadine a LED dotate di un chip in grado di ricevere

comandi attraverso un "bridge" centrale, un hub connesso alla rete Wi-Fi locale, utilizzando il protocollo di comunicazione ZLL.

Le lampadine smart Hue includono tre tipi di LED RGB che permettono di generare dinamicamente una vasta gamma di colori e intensità. La possibilità di collegare le luci in una rete mesh consente a ciascuna di trasmettere messaggi alla successiva, estendendo così il raggio d'azione e migliorando la robustezza del sistema.

A ciascuna luce è assegnato un URL nella rete locale, e i vari parametri controllabili di queste luci sono memorizzati come URL locali. Ciò significa che il controllo delle luci può essere facilmente ottenuto inviando nuovi valori a questi URL locali.

Dopo essere stato configurato, il sistema, in aggiunta al controllo tradizionale, è gestibile attraverso comandi vocali utilizzando l'applicazione per smartphone o tramite assistenti vocali come Amazon Alexa, Google Assistant o Siri, inoltre è possibile aggiungere al sistema un Philips Hue Motion sensor: un accessorio che attiva automaticamente le lampade connesse quando rileva un movimento, ad esempio qualcuno che cammina in corridoio o apre una porta, e le spegne dopo un determinato periodo di tempo. Inoltre numerosi sviluppatori indipendenti hanno contribuito alla creazione di diverse app come prodotti complementari [10].

2.2.4 Controlli e Algoritmi

Come visto nell'esempio precedente un sistema di Smart Lighting ha la possibilità di esser controllato anche con sensori o voce, oltre al metodo tradizionale. Possono essere però anche aggiunti controlli intelligenti come ad esempio algoritmi che, in base alla situazione di luce, regolano l'intensità, o accendono e spengono le luci, in base alla presenza di una persona all'interno di una stanza. Quindi è possibile creare un algoritmo di Video Tracking che vada a risolvere i possibili problemi causati dai sensori di occupazione citati precedentemente al 2.2.1.

2.3 Video Tracking

In questa sezione sarà analizzato in generale che cos'è il Video Tracking, chiamato più nello specifico Object Tracking, facendo riferimento a varie tipologie di tracker. Sarà infine presentato il modello scelto per il sistema proposto.

2.3.1 Object Tracking

Per 'Object Tracking' si intende il problema di stimare la traiettoria di un oggetto nel piano dell'immagine mentre si muove attraverso una scena. Un tracker assegna etichette coerenti agli oggetti tracciati in diversi frame di un video e può inoltre fornire informazioni specifiche sull'oggetto, come l'orientamento, l'area o la forma [11].

Data la sua natura, l' Object Tracking è utilizzato in molti campi come la sorveglianza, guida autonoma, monitoraggio del traffico e interazioni uomo-macchina, come la gesture recognition.

Le variabili che influenzano le prestazioni di un tracker sono molteplici e possono essere correlate all'ambiente, come la luminosità, l'occlusione, la stabilità della telecamera e la velocità di elaborazione, o legate all'oggetto che si sta cercando di tracciare, come forme o posizioni complesse e velocità di movimento [12]. Devono essere quindi soddisfatte tre caratteristiche fondamentali quando si va a progettare un algoritmo di Object Tracking [13]:

- **Robustezza:**

Il tracciamento deve essere eseguibile anche in condizioni complesse, come sfondi affollati, occlusioni e variazioni di illuminazione.

- **Adattabilità:**

L'algoritmo deve essere in grado di adattarsi ai cambiamenti rapidi e intricati dell'oggetto, mantenendo la capacità di rilevare e tracciare il bersaglio.

- **Elaborazione in Tempo Reale delle Informazioni:**

Data la necessità di una veloce elaborazione nell'analisi di sequenze di immagini, l'algoritmo deve assicurare prestazioni elevate.

2.4 Tipologie di Tracker

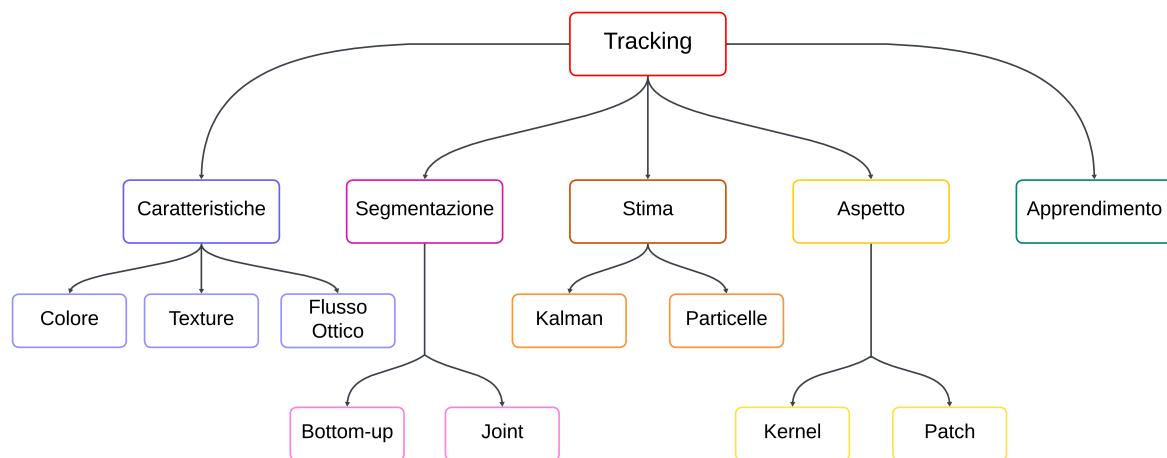


Figura 2.4: Tipologie di Tracker

Come è possibile osservare nella figura 2.4, i metodi di tracker vengono suddivisi generalmente in 5 categorie [14]:

2.4.1 Metodi basati sulle caratteristiche

Questi metodi si basano sull'estrazione di caratteristiche univoche degli oggetti per renderli facilmente distinguibili nella scena.

Una volta estratte tali caratteristiche, vengono utilizzate come criterio di similarità per individuare gli oggetti nei frame successivi. Questi approcci sono considerati i più semplici, ma

possono incontrare difficoltà nella capacità di distinguere più oggetti nella scena, poiché richiedono l'estrazione di caratteristiche uniche, esatte e affidabili per rendere i bersagli facilmente distinguibili. Alcune di queste caratteristiche possono essere:

Colore:

Uno degli approcci più comuni di utilizzare il colore come caratteristica consiste nell'estrarrre da ogni oggetto un istogramma di colore.

Questo istogramma rappresenta la distribuzione dei colori in un'immagine e il numero di pixel per ciascun colore. Tuttavia, oggetti diversi potrebbero avere lo stesso istogramma, poiché vengono ignorate sia la forma che altre caratteristiche visive dell'immagine essendo l'unico criterio considerato è il colore.

Texture:

Per texture si intende una sequenza di informazioni, od organizzazioni strutturali, ripetuta a intervalli regolari ottenuti dalla preelaborazione dell'immagine. Può essere usata per descrivere un oggetto in aggiunta all'istogramma.

Flusso Ottico:

Rappresenta il moto apparente degli oggetti in base alla variazione di luminosità nell'immagine.

Gli algoritmi che utilizzano il flusso ottico per prevedere il movimento analizzano la variazione di luminosità dei pixel dell'immagine da un frame all'altro.

2.4.2 Metodi basati sulla segmentazione

Gli algoritmi di questo tipo si fondano sulla separazione degli oggetti in primo piano, che rappresentano i soggetti in movimento e costituiscono il bersaglio del tracciamento, da quelli di sfondo.

Uno dei metodi impiegati è il metodo basato su Bottom-Up, che si compone di due fasi distinte: la prima effettua la segmentazione a basso livello del primo piano per estrarre le regioni dal frame, mentre la seconda si occupa del tracciamento degli oggetti in primo piano utilizzando le caratteristiche estratte dalle regioni.

Poiché eventuali errori nella fase di segmentazione possono propagarsi, al fine di migliorare le prestazioni complessive del sistema le due fasi sono state integrate, in un approccio combinato noto come Joint based method,

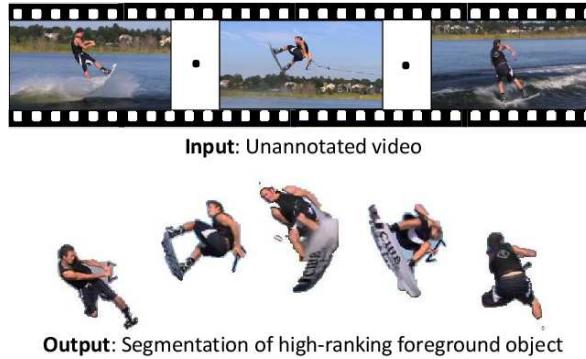


Figura 2.5: Esempio di processo di segmentazione [15]

2.4.3 Metodi basati sulla stima

Il problema di tracciamento è trattato come un problema di stima in cui l'oggetto è rappresentato come un vettore di stato che descrive i suoi comportamenti dinamici come posizione e velocità.

Il comportamento dell'algoritmo è dato dai metodi statistici Bayesiani¹. In ogni frame avvengono due passi ricorsivamente, il primo è la predizione, in cui viene stimata la prossima posizione del bersaglio, il secondo è l'aggiornamento, in cui viene modificato il vettore degli stati.

Due metodi generalmente usati per il tracciamento di oggetti sono il filtro di Kalman e il filtro di particelle.

2.4.4 Metodi basati sull'aspetto

Si basano sulle caratteristiche più complesse dell'oggetto, non limitate a forma e colore.

Ai fini del tracciamento le caratteristiche dell'oggetto e i suoi cambiamenti vanno implementati nell'algoritmo. Sono in genere suddivisi in due categorie:

Tracciamento basato su kernel

I bersagli sono rappresentati come forme geometriche primitive che servono a rendere la forma dell'oggetto costante durante il tracciamento 2.6.

A ogni pixel è associato un peso grazie al quale, insieme alle coordinate attuali, è calcolato il moto secondo una funzione di distribuzione.

¹Metodi basati sul teorema di Bayes in cui le informazioni disponibili sui parametri statistici sono aggiornati con i dati osservati [16]

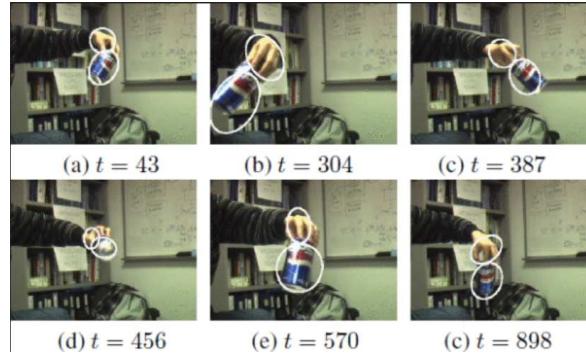


Figura 2.6: Esempio di tracciamento effettuato tramite filtraggio Bayesiano basato su kernel [17]

Tracciamento basato su patch

Il bersaglio viene diviso in sezioni più piccole denominate "patch", ciascuna conserva le informazioni specifiche della propria porzione.

L'aggregato di tutte le patch rappresenta integralmente il bersaglio.

2.4.5 Metodi basati sull'apprendimento

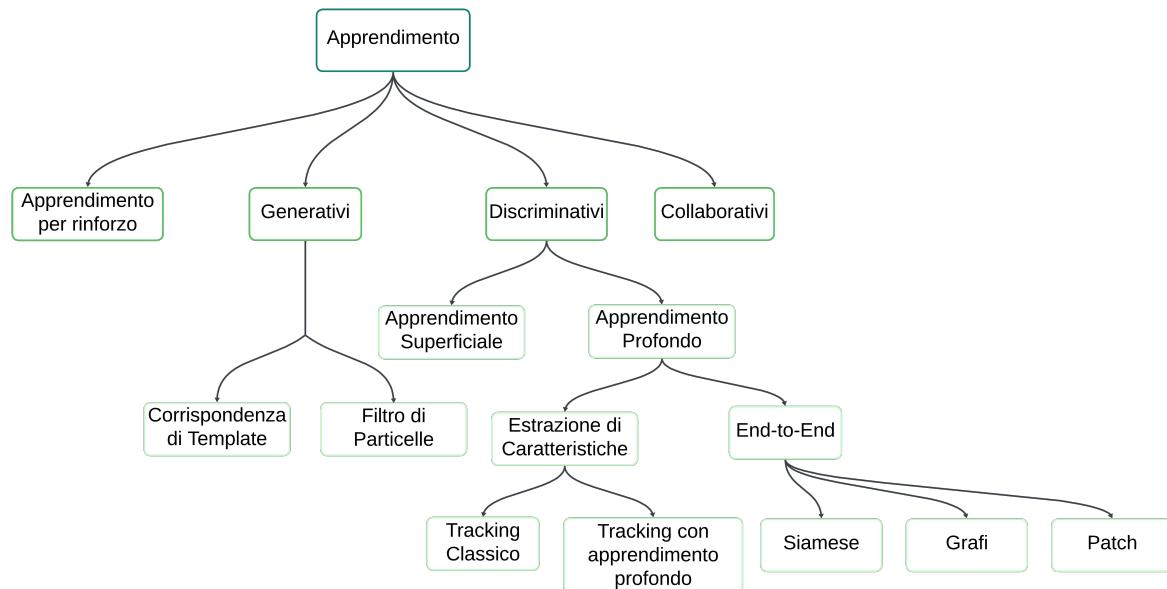


Figura 2.7: Tipologie di Tracker basati sull'apprendimento

Eseguono il tracciamento basandosi sull'apprendimento dell'aspetto, delle caratteristiche e delle predizioni effettuate precedentemente. I tracker che utilizzano l'apprendimento possono essere suddivisi in basati sull'apprendimento per rinforzo, generativi, discriminativi e collaborativi.

Apprendimento per rinforzo

Alcuni algoritmi di Object Tracking sfruttano l'apprendimento per rinforzo che consiste in un agente che, in diversi intervalli di tempo, dapprima osserva lo stato, seleziona un'azione e la applica al processo, modificando lo stato. Tramite l'esperienza e i tentativi apprende le azioni ottimali da effettuare. [18]

Generativi

Data una prima informazione riguardo l'oggetto nel primo frame, cercano il bersaglio nelle aree più simili all'oggetto dei frame successivi. Questi tracker sono in genere suddivisi in due categorie:

- **Corrispondenza di template:** ogni oggetto è rappresentato con un template, che viene usato per il confronto con i frame precedenti tramite un metodo di misura di similarità.
- **Filtro di particelle:** la posizione del bersaglio è predetta in base a una stima ottenuta dalle osservazioni fatte nei frame precedenti.

Discriminativi

Sono algoritmi che considera il tracking come un problema di classificazione che separa il bersaglio da tracciare dal background. Sono suddivisi in:

- **Apprendimento superficiale:** Il classificatore è addestrato a discriminare l'oggetto tracciato dallo sfondo, dopodiché decide nella fase di test se l'oggetto è il bersaglio. Come criterio di discriminazione, il classificatore può usare le caratteristiche estratte dall'oggetto oppure filtri di correlazione.

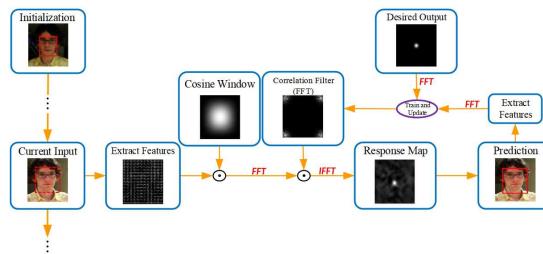


Figura 2.8: Funzionamento di un tracker che utilizza filtri di correlazione [19]

- **Apprendimento profondo:** I tracker di questo tipo si sono dimostrati più precisi, robusti e veloci rispetto ai tracker tradizionali [20] [21]. In contrasto con l'apprendimento superficiale, le caratteristiche necessarie per la discriminazione vengono estratte autonomamente dal modello. Diverse reti neurali possono essere impiegate per svariati processi, inclusi il rilevamento degli oggetti, la misura della similarità e la stima dello stato.

Questo tipo di tracker offre la flessibilità di separare le fasi di rilevamento e tracciamento. È possibile delegare il tracciamento a un metodo classico, sfruttando il solo apprendimento profondo per l'estrazione delle caratteristiche. In alternativa, entrambe le fasi possono essere gestite dallo stesso network, permettendo l'utilizzo di un tracker basato su approcci end-to-end.

Tra i metodi end-to-end, vi sono i tracker siamesi che confrontano due frame di input per determinare se l'oggetto è presente in entrambi, i tracker basati sull'apprendimento di patch, in cui il modello viene addestrato su campioni positivi e negativi, e i tracker basati sui grafi che forniscono una rappresentazione delle relazioni spaziali in un'immagine.

Collaborativi

Algoritmi di questo tipo sfruttano le caratteristiche sia dei tracker generativi che dei tracker discriminativi per incorporarne i vantaggi e costruire un tracker più robusto e accurato. Ad esempio, possono utilizzare le capacità dei tracker generativi di mantenere informazioni attraverso i frame nel tempo e sfruttare le capacità dei tracker discriminativi per separare con precisione il bersaglio dallo sfondo [22].

L'obiettivo è combinare le forze di entrambi i tipi di tracker per migliorare le prestazioni complessive.

2.5 CNN e YOLO

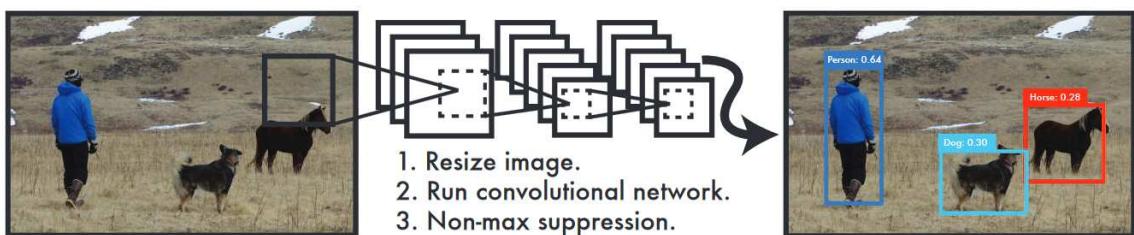


Figura 2.9: Elaborazione dell'immagine utilizzando YOLO [23]

A oggi le reti neurali, in particolare le reti neurali convoluzionali(CNN), sono considerate il metodo predefinito nell'elaborazione di immagini, grazie all'utilizzo di più livelli computazionali in cui si alternano tra convoluzione² e aggregazione³ [25], permettendo di scoprire le caratteristiche semantiche dei pixel dell'immagine meglio delle tecniche che utilizzano i metodi convenzionali [26].

Il modello scelto è YOLO, un modello di Video Tracking basato su apprendimento profondo end-to-end, che tratta il problema di rilevamento degli oggetti come uno di regressione per identificare la classe e la posizioni degli oggetti rilevati da una singola rete neurale convoluzionale [23].

²Hanno il ruolo di scansionare l'immagine e identificare le caratteristiche specifiche [24]

³Viene fatto sottocampionamento, ovvero ridotta la dimensione spaziale dell'input, eliminando le informazioni meno utili

YOLO permette un bilanciamento notevole fra accuratezza e velocità, data la rapida e affidabile identificazione di oggetti nell’immagine [27].

Capitolo 3

Componenti del Sistema

In questo capitolo verranno analizzate nel dettaglio le due componenti fondamentali del sistema: il network su cui si basa il sistema proposto, KNX, e l'algoritmo di Video Tracking utilizzato, YOLOv8.

3.1 KNX

Come precedentemente introdotto, KNX è la tecnologia che permette la comunicazione di vari dispositivi di produttori diversi al fine di avere un ecosistema di smart home in grado di comunicare agevolmente.

KNX è uno standard aperto Europeo, approvato come standard internazionale nel 2006. Il sistema KNX è un sistema decentralizzato e distribuito in cui i tutti i dispositivi danno vita ad applicazioni distribuite, in cui la loro iterazione è possibile, grazie a tipi di dati standardizzati(Datapoint) e a blocchi funzionali, che rappresentano i canali logici dei dispositivi. Il collegamento avviene tramite doppino, cavo della corrente o anche radio frequenza.

Grazie al software disponibile per Windows, indipendente dai produttori, Engineering Tool Software(ETS), è possibile collegare una serie di dispositivi individuali in un'installazione funzionante e integrare diversi mezzi trasmissivi e modalità di configurazione KNX.

Il sistema è inoltre estremamente adattabile, non avendo necessità di un microprocessore o architettura specifica [28].

3.1.1 Architettura KNX

Gli elementi principali del sistema sono:

- **Modelli di applicazione distribuiti e l'interoperabilità:** Rappresentano le possibili operazioni rese disponibili dai dispositivi attraverso Datapoint.
- **Schemi di configurazione e di gestione:** Tutte le risorse del network sono rese interoperabili e disponibili nel sistema distribuito grazie a questi schemi.
- **Sistema di comunicazione:** Oltre al livello di comunicazione fisico, esiste anche un protocollo di messaggi e modelli di comunicazione in ogni nodo, tutto gestito dal KNX

Common Kernel.

Questo sistema supporta tutti i requisiti di comunicazione di rete per la configurazione e gestione dell'installazione, oltre a fare da host alle applicazioni distribuite.

- **Modelli di dispositivi concreti:** I dispositivi sono conformi a profili forniti da KNX per la realizzazione, al fine di creare prodotti che possono essere inseriti in un'installazione, realizzando e combinando più efficacemente i passaggi precedenti.

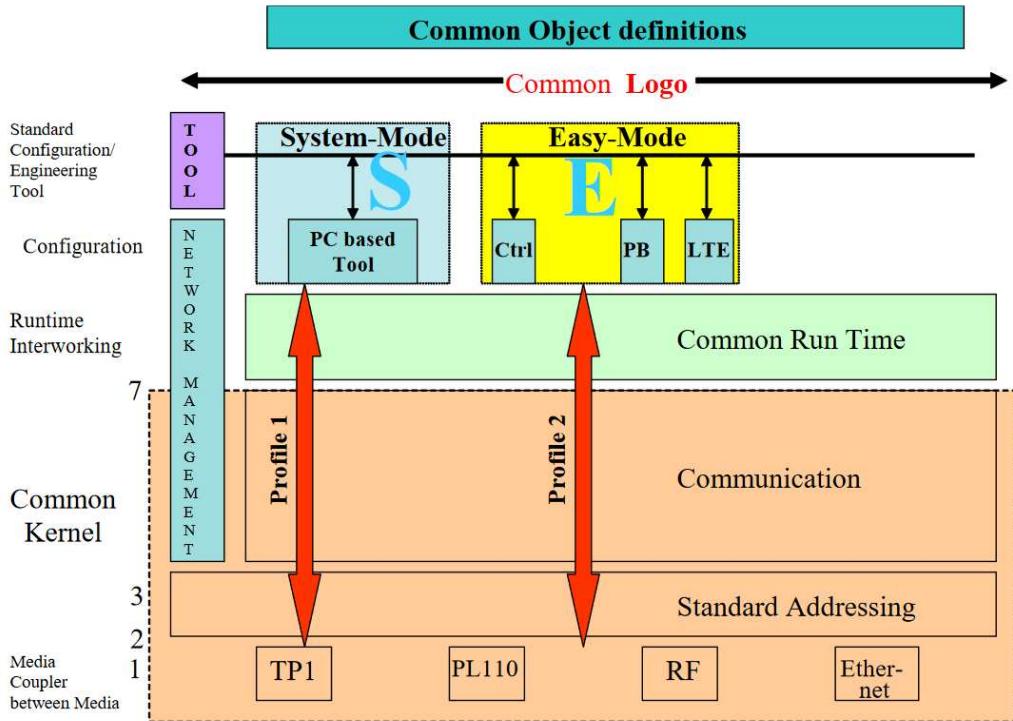


Figura 3.1: Architettura KNX, immagine ottenuta dal manuale KNX Specifications [28]

Topologia e Indirizzamento

Prima di approfondire i dettagli degli elementi precedentemente descritti, è essenziale discutere della topologia di una rete KNX. KNX consente l'installazione di 65536 dispositivi. Come illustrato nella Figura 3.2, ogni sottorete comprende 256 dispositivi, di cui alcuni fungono da accoppiatori che facilitano la connessione tra linee o segmenti all'interno di ogni sottorete. 16 linee sono raggruppate in un'area, e 15 aree formano un dominio, collegandosi attraverso un backbone.

Il sistema KNX utilizza una combinazione di comunicazione broadcast e punto a punto per la gestione della rete e delle risorse dei dispositivi. Durante l'installazione, a ciascun dispositivo viene assegnato un indirizzo univoco di 16 bit tramite broadcast, e le comunicazioni successive avvengono punto a punto utilizzando questi indirizzi.

Ciascun dispositivo mette a disposizione diversi oggetti di gruppo, ciascuno dei quali è asso-

ciato a un Datapoint, ossia un tipo di dato standardizzato. Questi oggetti di gruppo possono essere aggregati indipendentemente dalla topologia della rete attraverso variabili condivise, note come indirizzi di gruppo, creando un disaccoppiamento.

Le variabili di gruppo sono identificate da un indirizzo di 16 bit e possono essere lette e scritte bidirezionalmente. Ad esempio, una variabile che gestisce l'accensione delle luci con indirizzo 0/0/1 può essere associata agli attuatori con indirizzo fisico 1/1/1 e all'interruttore con indirizzo 2/1/1.

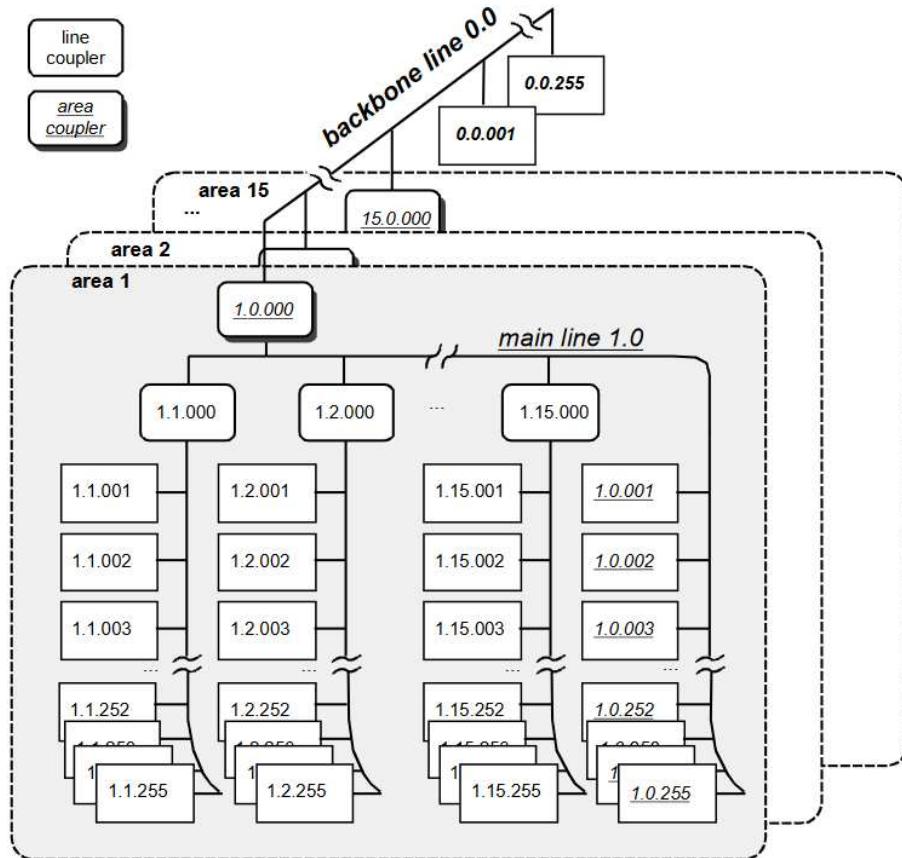


Figura 3.2: Topologia KNX, immagine ottenuta dal manuale KNX Specifications [28]

Frame KNX

octet 0	1	2	3	4	5	6	7	8	..	N - 1	N ≤ 22
Control Field	Source Address	Destination Address		Address Type; NPCI; length	TPCI	APCI	data/APCI	data			FrameCheck

Figura 3.3: Frame KNX, immagine ottenuta dal manuale KNX Specifications [28]

Il telegrammi mandati sul bus della rete possono trasportare fino a 1984 bit, divisi in 248 ottetti. Con riferimento alla figura 3.3, le parti in cui è diviso il frame includono:

- Campo di Controllo: determina la priorità del frame
- Indirizzo del mittente.
- Indirizzo del destinatario: può essere sia individuale che di gruppo dato poichè KNX permette il multicast.
- Tipo dell'indirizzo di gruppo.
- TPCI: gestisce le relazioni comunicative a livello di trasporto, come il mantenimento delle connessioni punto a punto.
- APCI: gestisce i meccanismi del livello di applicazione.
- Dati: contiene fino a 14 ottetti di dati, equivalenti a 112 bit, conformemente alle specifiche del protocollo applicativo e allo schema di indirizzamento.
- Controllo del frame: contribuisce alla consistenza e l'affidabilità della trasmissione.

3.1.2 Modelli di applicazione

Un modello di applicazione consiste nell'insieme degli oggetti di gruppo resi disponibili da un dispositivo. Il sistema opera grazie all'interazione dei Datapoint provenienti da dispositivi diversi, connettendosi attraverso un identificatore comune, come gli indirizzi di gruppo.

Quando un'applicazione imposta un valore su un Datapoint mittente, il dispositivo invia successivamente un messaggio di scrittura che include l'indirizzo di gruppo corrispondente e il nuovo valore. Un Datapoint ricevente con lo stesso indirizzo di gruppo riceve e comunica questo valore all'applicazione locale. Quest'ultima può quindi intraprendere azioni in risposta alla modifica di questo valore, come cambiare uno stato interno, uno stato di output fisico, o entrambi.

Un altro modo per interagire con i Datapoint è attraverso gli oggetti di interfaccia che raggruppano le proprietà di un determinato oggetto di gruppo in un oggetto separato.

Per permettere lo scambio di informazioni tra Datapoint è necessario collegarli a un indirizzo di gruppo comune, a seconda della configurazione dei dispositivi questo può avvenire in tre modi:

- **Binding libero:** non c'è nessuna restrizione.
- **Binding strutturato:** il modello di applicazione stipula uno schema su come collegare vari Datapoint, il valore dell'indirizzo è libero.
- **Binding tramite Tag:** il valore dell'indirizzo deve rispettare la zona specificata dal dispositivo.

3.1.3 Interoperabilità

Il concetto di interoperabilità è uno dei numerosi vantaggi offerti da un sistema KNX, esso garantisce la possibilità di ottenere l'integrazione più completa possibile tra dispositivi all'interno di qualsiasi applicazione.

I principi di interoperabilità sono rappresentati dai blocchi funzionali che descrivono l'aspetto di ciascuna applicazione locale quando vista dalla rete. Questi blocchi funzionali sono costituiti dall'interfaccia dei Datapoint, dal comportamento interno previsto, dai messaggi e dall'I/O fisico.

All'interno della specifica dei blocchi funzionali, a ogni Datapoint è assegnato un nome esplicativo e il tipo richiesto. Questo tipo può essere generico, come un booleano o una percentuale, ma possono anche essere definiti parametri, ovvero tipi più specializzati.

Al fine di garantire una corretta interoperabilità, è necessario che ogni dispositivo sia in grado di fornire informazioni essenziali alla rete, ad esempio il profilo che implementa.

Grazie a ETS è possibile consentire un'integrazione immediata tra diverse modalità di configurazione all'interno di un'unica installazione.

3.1.4 Modalità di configurazione

I dispositivi che fanno parte di un sistema KNX, hanno la possibilità di scegliere tra diverse modalità di configurazione, dando sempre la possibilità di estendere l'installazione. Questa specifica assicura una certa libertà per il produttore, garantendo al contempo coerenza e interoperabilità, anche in presenza di fornitori multipli.

ETS rende possibile l'interoperabilità tra dispositivi che hanno configurazioni differenti, attraverso l'uso di oggetti, indirizzi di gruppo e procedure di gestione coerenti, tramite l'utilizzo dei descrittori forniti dai dispositivi.

3.1.5 Gestione della rete

La gestione della rete in KNX prevede un insieme di meccanismi per scoprire, impostare o recuperare dati di configurazione attivamente attraverso la rete. Il sistema propone procedure, ovvero sequenze di messaggi, per accedere ai valori delle diverse risorse di rete all'interno dei dispositivi. Vengono inoltre definiti identificatori e formati per queste risorse al fine di garantire una corretta interoperabilità tra tutti i dispositivi di rete KNX.

Tali risorse possono includere indirizzi, parametri di comunicazione, parametri dell'applicazione o insiemi complessi di dati come tabelle di binding o addirittura l'intero programma applicativo eseguibile.

Ogni dispositivo che implementa una specifica modalità di configurazione deve aderire ai servizi e alle risorse specificati nel relativo profilo. Questi vengono utilizzati all'interno delle procedure identificate dalla modalità stessa.

3.1.6 Sistema di comunicazione

Un modello di kernel comune è condiviso da tutti i dispositivi. Questo modello svolge un ruolo chiave nel soddisfare le esigenze dei modelli di applicazione, della configurazione e della gestione di rete.

Il kernel include un sistema di comunicazione conforme al modello OSI a 7 strati:

- Livello Data Link: fornisce il controllo di accesso al mezzo trasmissivo e il controllo logico del collegamento.
- Livello di rete: provvede un telegramma con conferma segmentaria e controlla l'hop count del pacchetto¹.
- Livello di trasporto: rende disponibili 4 tipi di comunicazione fra i punti:multicast, broadcast, uno a uno con connessione e senza.
- Livello di sessione e presentazione: sono vuoti.
- Livello di applicazione: Offre vari servizi al processo applicativo in base al tipo di comunicazione utilizzata a livello di trasporto.

3.1.7 Dispositivi e Profili

Un'installazione KNX consiste sempre in un insieme di dispositivi collegati al bus o alla rete e identificati da un indirizzo univoco. Tutti questi dispositivi devono aderire a un profilo, al fine di ottenere la certificazione KNX. Questi profili sintetizzano una serie di caratteristiche che il dispositivo deve avere.

3.1.8 ETS

ETS, fornito dalla KNX Association e progettato per il sistema operativo Windows, è un software dedicato alla progettazione e configurazione di installazioni KNX.

Ogni produttore è responsabile di mantenere aggiornate le informazioni relative ai propri dispositivi nel database di ETS, il quale offre un ampio supporto e gestione di tutti i dispositivi presenti nel catalogo. Questo consente a ETS di accedere ai programmi applicativi e ai parametri disponibili per ciascun dispositivo.

In aggiunta, come precedentemente menzionato, ETS agevola l'integrazione tra dispositivi con configurazioni diverse e offre la possibilità di ottenere dettagli sugli elementi dell'installazione attraverso una scansione.

3.2 YOLO: You Only Look Once

Per l'implementazione del sistema di Video Tracking verrà utilizzato YOLO (You Only Look Once), un modello di rilevamento di oggetti end-to-end. Il modello è stato presentato per la

¹L'hop count di un frame viene decrementato dai router per evitare il looping dei messaggi; quando diventa zero, il frame viene scartato dalla rete.

prima volta da Joseph Redmon e il suo team nel 2016 durante la CVPR e successivamente è stata oggetto di modifiche e miglioramenti da parte di Ultralytics, portando alla sua versione attuale YOLOv8.

3.2.1 Metodi a una fase e due fasi

YOLO è un metodo a una fase ovvero esegue la selezione e la predizione in un singolo passaggio, grazie al trattamento del problema di rilevamento come regressione, la fase iniziale di selezione delle regioni è così eliminata, il che rende il modello computazionalmente meno esigente garantendo velocità di rilevamento più elevate e FPS maggiore, nonostante abbia una precisione inferiore rispetto ai rilevatori a due fasi [26].

Quest'ultimi dividono il processo in due fasi, prima effettuano l'estrazione delle caratteristiche, proponendo aree dell'immagine dove potrebbero esserci oggetti, dopodiché utilizzano un classificatore sulle aree proposte.

Questo richiede molto potere computazionale dato che ogni componente va addestrato separatamente, nonostante garantiscano precisione maggiore.

3.2.2 Funzionamento del modello

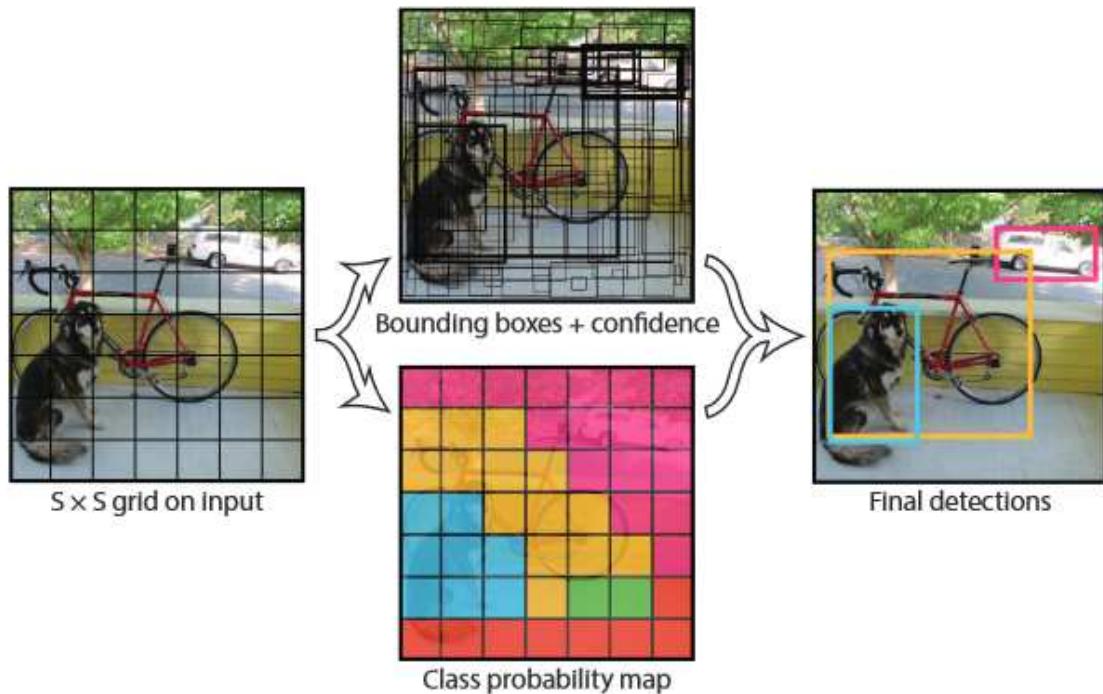


Figura 3.4: Immagine che mostra la divisione in griglia e le bounding box [23]

YOLO unisce le componenti della rilevazione di oggetti in una singola rete neurale. Questa rete utilizza le caratteristiche dell'intera immagine per predire simultaneamente le classi e le bounding box, ovvero i rettangoli che racchiudono gli oggetti rilevati.

Inoltre il modello ha la capacità di avere una conoscenza globale dell'immagine. Questo significa che durante l'addestramento e il testing, la rete è in grado di considerare l'intera immagine, consentendo di comprendere le informazioni contestuali sulle classi e sul loro aspetto.

Il sistema divide l'immagine in una griglia SxS, se il centro di un oggetto è in una cella, questa sarà responsabile del rilevamento, rilevando B bounding box, formate da quattro valori che indicano la sua posizione nello spazio: le coordinate x,y e la larghezza e l'altezza rispetto all'immagine, e il relativo punteggio di fiducia, valore rappresenta quanto il modello è fiducioso che ci sia un oggetto nella bounding box.

Ogni cella calcola inoltre C probabilità condizionate di classe, che moltiplicate al punteggio di confidenza della bounding box, stabiliscono quanto sia probabile che una classe compaia. Questi parametri sono utilizzati per costruire l'output.

3.2.3 Punti di forza e problemi di YOLOv1

Nonostante l'elevata velocità abbia permesso l'utilizzo del modello in tempo reale, la prima versione presentava un errore di localizzazione maggiore rispetto ai tracker a due fasi. Questo era dovuto alla difficoltà nel predire oggetti troppo vicini o con rapporti di aspetto non conformi ai dati forniti durante l'addestramento.

Le versioni successive hanno introdotto meccanismi con l'obbiettivo di migliorare il modello.

3.2.4 Evoluzione negli anni

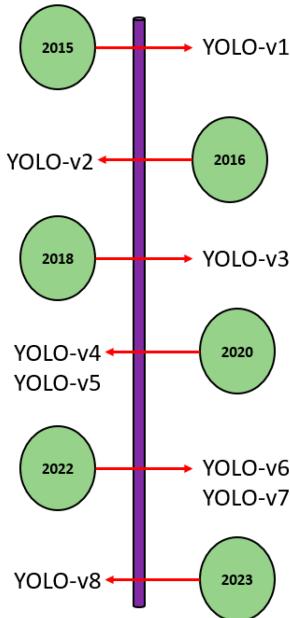


Figura 3.5: Linea temporale dell'evoluzione di YOLO [26]

YOLO-v2: Rilasciato l'anno dopo la prima versione, ha introdotto miglioramenti all'architettura come l'utilizzo di una risoluzione più alta, della batch normalization² e di anchor boxes,

²Tecnica utilizzata nell'apprendimento profondo con lo scopo di velocizzare la fase di addestramento e migliorare la precisione [29]

ovvero bounding box di dimensioni predeterminate, ottenute tramite un algoritmo di clustering k-means effettuato sul set di addestramento.

YOLO-v3: Rilasciato nel 2018, introdusse la possibilità di fare predizione a tre livelli di granularità al fine di migliorare le performance del modello nella rilevazione di oggetti più piccoli.

YOLO-v4: Dopo due anni dal rilascio del modello precedente, questa versione, oltre a portare ulteriori modifiche architettoniche introdusse una migliorata aggregazione delle caratteristiche e una serie di ottimizzazioni con costo computazionale molto basso.

YOLO-v5: A distanza di pochi mesi, questa divenne la prima versione a essere accessibile a un pubblico più ampio, grazie all'essere basata sul framework Pytorch e a una repository GitHub.

YOLO-v6 e YOLO-v7: Queste versione ha significativamente migliorato le prestazioni e la velocità del modello attraverso ottimizzazioni dell'architettura e delle funzioni di perdita. L'eliminazione delle anchor box, inoltre, rese l'esecuzione del modello più efficiente anche su dispositivi meno potenti.

3.2.5 YOLO v8

La versione più recente di YOLO, rilasciata nel 2023, offre supporto per il rilevamento di oggetti, la classificazione e la segmentazione, ed è facilmente installabile tramite il pacchetto Python [30]. Questa nuova iterazione, oltre a introdurre ulteriori miglioramenti nell'architettura per mantenere un'alta velocità ed efficienza, ha introdotto la separazione tra i processi di classificazione e regressione. Questa divisione contribuisce a garantire una maggiore precisione complessiva al modello.

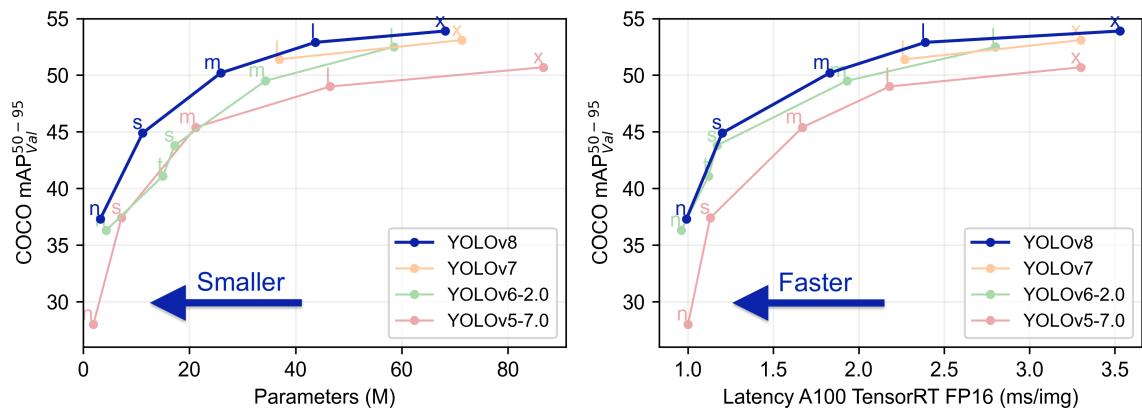


Figura 3.6: Prestazioni della nuova versione di YOLO rispetto alle precedenti [30]

Grazie alle sue prestazioni eccellenti e alla facilità di utilizzo, è stata scelta questa versione del modello per l'implementazione del sistema. Suddividendo l'area rilevata dalla telecamera in zone specifiche, siamo in grado di sfruttare le informazioni ottenute dal rilevamento del modello per implementare un sistema di Video Tracking.

Capitolo 4

Analisi del Sistema Implementato

In questo capitolo conclusivo, saranno esaminate le fasi di implementazione del sistema KNX, concentrando in modo specifico sul progetto ETS che gestisce il laboratorio. Successivamente, sarà approfondita la configurazione delle diverse telecamere, evidenziando i vantaggi e le potenziali criticità di ognuna, motivando la scelta della telecamera utilizzata per l’interazione con il sistema. Infine, il sistema creato sarà dettagliato attraverso porzioni di codice, diagrammi di classe e spiegazioni sul suo funzionamento.

4.1 Topologia del sistema

Dalla figura 4.1, è evidente che il sistema di Smart Lighting opererà su un computer connesso alla stessa LAN dell’interfaccia IP di KNX e della telecamera IP.

Attraverso un dispositivo EIBPORT, è possibile interconnettere tutti i dispositivi KNX nell’installazione, con particolare rilievo per i due DALI Gateway. Questi ultimi rivestono un’importanza significativa poiché consentono la connessione delle luci al sistema sfruttando un bus DALI.

4.2 Installazione KNX CASALab

4.2.1 EIBPORT

L’intera configurazione KNX del laboratorio CASALab si anima grazie al dispositivo EIBPORT, il quale svolge una duplice funzione fondamentale. Innanzitutto, offre la possibilità di connettersi a un’applicazione web dedicata, consentendo la gestione completa di tutta l’installazione. In secondo luogo, agisce come interfaccia IP, agevolando l’interazione con tutte le componenti del sistema KNX.

Questo dispositivo funge da gateway tra la rete LAN e il sistema KNX, consentendo l’accesso agli oggetti di gruppo e la modifica degli indirizzi di gruppo attraverso ETS, a condizione che il computer su cui è in esecuzione il programma e l’interfaccia, siano collegati alla stessa LAN. Attraverso l’utilizzo del software BABStarter 4.3, reso disponibile da BAB Technologie, è possibile effettuare regolazioni e personalizzazioni delle impostazioni del dispositivo. Questo

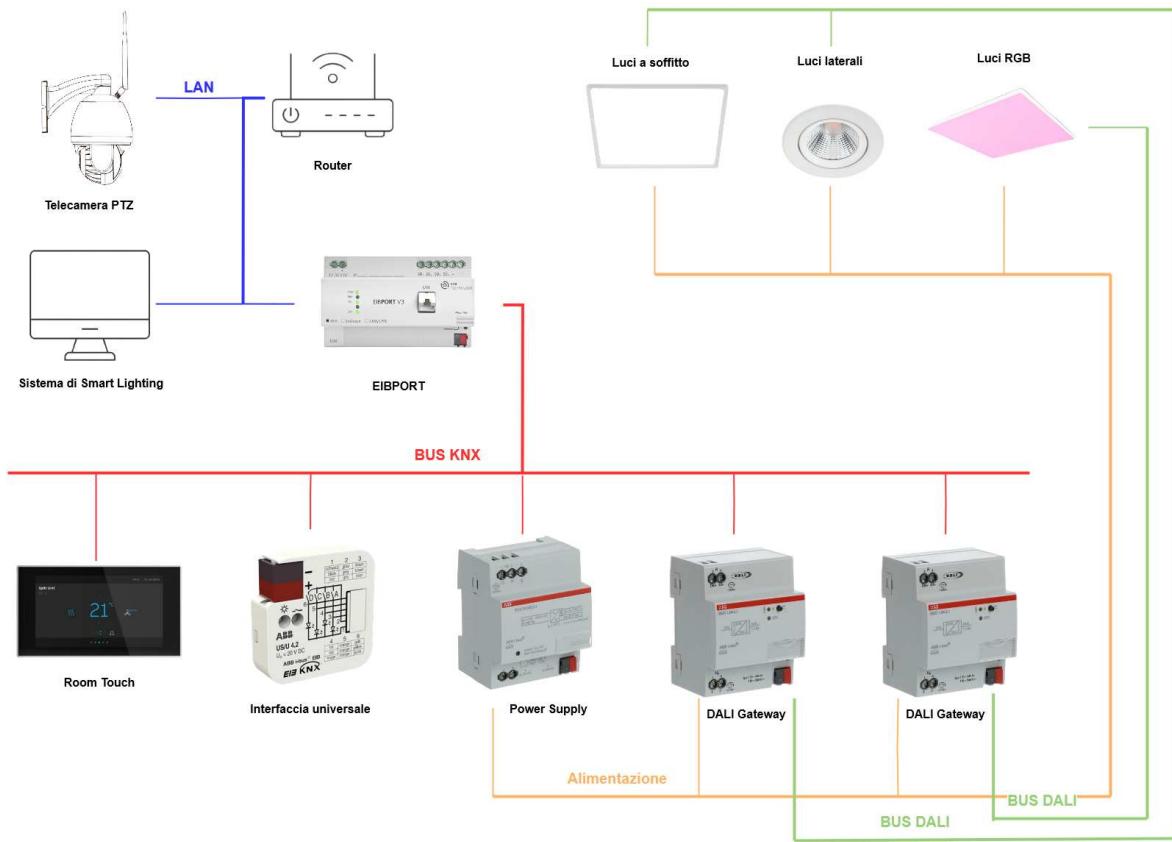


Figura 4.1: Topologia delle connessioni del sistema

include modifiche come la configurazione dell'indirizzo IP, delle credenziali e dell'indirizzo fisico del gateway sulla rete KNX, fornendo una maggiore flessibilità e controllo sulle funzionalità del sistema KNX.

4.2.2 Dispositivi dell'installazione

Grazie al progetto ETS è possibile configurare i dispositivi installati nel laboratorio. Il sistema, oltre all'interfaccia fornita dall'EIBPORT, la cui posizione predefinita è la linea principale 4, è composto da cinque dispositivi, come evidenziato nell'immagine 4.2

- **Power Supply:** Questo dispositivo è responsabile unicamente dell'alimentazione del sistema e non dispone di oggetti di gruppo.
- **Dali-Gateway:** Ne sono presenti due, uno per ogni ambiente. Questi dispositivi sono responsabili del collegamento delle luci al sistema KNX.
- **Interfaccia universale:** Consente il collegamento di una pulsantiera tradizionale al sistema.
- **ABB RoomTouch:** Questo dispositivo offre molteplici funzioni altamente personalizzabili tramite l'estensione di ABB per ETS. Attualmente, lo schermo consente il controllo completo delle luci nel laboratorio, sia in modalità singola che attraverso scenari. Tra le

sue funzionalità includono la gestione della luminosità e della temperatura colore, oltre alla regolazione del colore RGB delle luci attraverso una ruota dei colori.



Figura 4.2: Schermata dei dispositivi del progetto ETS



Figura 4.3: Software BABStarter

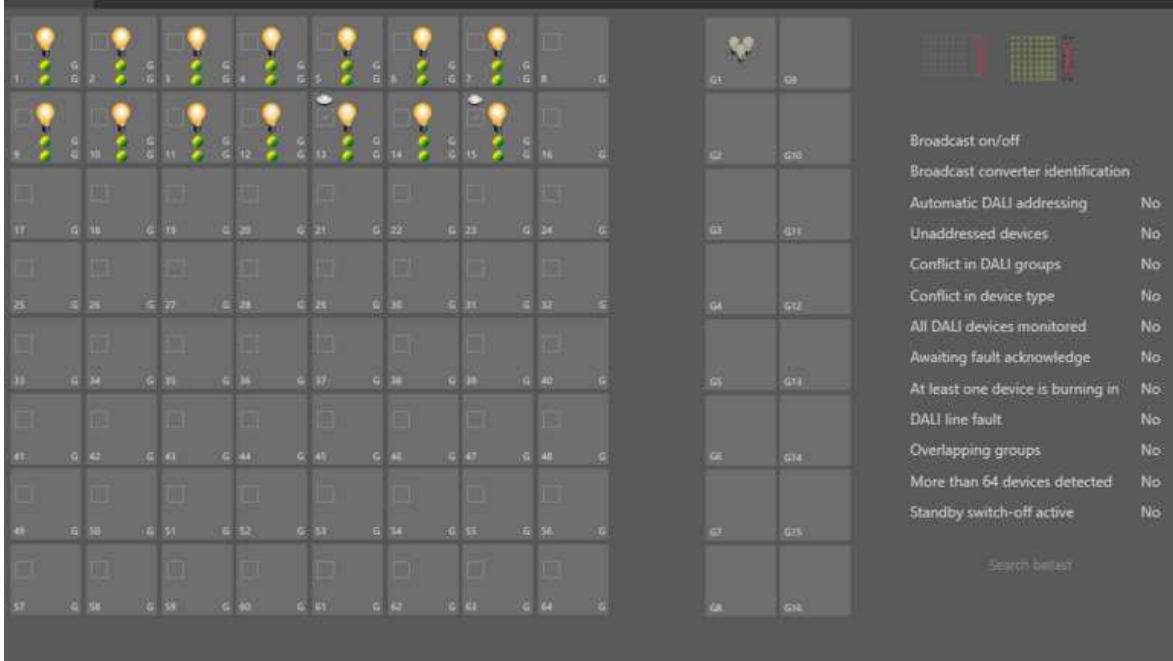


Figura 4.4: Configurazione dei gruppi di luci nel software I-bus Tool

4.2.3 Dali-Gateway

I Dali-Gateway di ABB rivestono un ruolo fondamentale come interfaccia tra l'installazione KNX e il sistema di controllo dell'illuminazione DALI, consentendo la connessione di fino a 64 dispositivi, suddivisi in 16 gruppi [31]. Utilizzando il software I-bus Tool fornito da ABB, è possibile installare e configurare le luci. Questo strumento offre la flessibilità di personalizzare l'installazione, consentendo la creazione o la modifica di gruppi di luci.

È possibile, inoltre, configurare le luci in modo che siano singolarmente associabili a un indirizzo di gruppo, permettendo un controllo altamente personalizzato.

Come illustrato nell'immagine 4.4, le luci principali della sala grande sono state aggregate in un unico gruppo per agevolare un controllo globale dell'illuminazione. Analogamente, le luci laterali della sala principale sono state raggruppate insieme, mentre un'interfaccia DALI separata gestisce le luci RGB dell'ufficio.

Gli oggetti di gruppo forniti dall'interfaccia DALI della sala principale comprendono funzionalità di accensione/spegnimento, controllo della luminosità e gestione della temperatura colore, ciascuna associata ad un controllo di stato corrispondente.

Per quanto riguarda l'interfaccia DALI per le luci RGB, sono disponibili oggetti di gruppo per la commutazione, il controllo della luminosità e la gestione dei colori. Ogni colore è rappresentato da tre percentuali RGB, gestite separatamente tramite un oggetto di gruppo dedicato, con un gruppo aggiuntivo per il controllo del bianco (4.5). Tutti questi oggetti di gruppo sono accessibili attraverso parametri su ETS resi disponibili dall'interfaccia, mentre quest'ultima si occupa del raggruppamento.

L'interfaccia DALI rende disponibile anche una funzionalità scenario per il controllo di tutti i gruppi, attraverso una sola iterazione.

83	Uscita A - G1: Rosso -RGB	Commutazione	0/0/3 luce A3 - s... 0/0/3	1 bit	C - W -	-	switch	Basso
84	Uscita A - G1: Rosso -RGB	Stato commutazione		1 bit	C R - T -	-	switch	Basso
85	Uscita A - G1: Rosso -RGB	Dimmer relativo		4 bit	C - W -	-	dimming c...Basso	
86	Uscita A - G1: Rosso -RGB	Valore luminosità	0/3/2 luce A3 - R...0/3/2	1 byte	C - W -	-	percentag... Basso	
87	Uscita A - G1: Rosso -RGB	Stato valore luminosità		1 byte	C R - T -	-	percentag... Basso	
91	Uscita A - G1: Rosso -RGB	Dimmer temperatura colore		4 bit	C - W -	-	dimming c...Basso	
92	Uscita A - G1: Rosso -RGB	Impostare temp. colore (Kelvin)		2 bytes	C - W -	-	absolute c... Basso	
105	Uscita A - G2: Verde - RGB	Commutazione	0/0/3 luce A3 - s... 0/0/3	1 bit	C - W -	-	switch	Basso
106	Uscita A - G2: Verde - RGB	Stato commutazione		1 bit	C R - T -	-	switch	Basso
107	Uscita A - G2: Verde - RGB	Dimmer relativo		4 bit	C - W -	-	dimming c...Basso	
108	Uscita A - G2: Verde - RGB	Valore luminosità	0/3/3 luce A3 - ... 0/3/3	1 byte	C - W -	-	percentag... Basso	
109	Uscita A - G2: Verde - RGB	Stato valore luminosità	1/0/1 Stato valor... 1/0/1	1 byte	C R - T -	-	percentag... Basso	
113	Uscita A - G2: Verde - RGB	Dimmer temperatura colore		4 bit	C - W -	-	dimming c...Basso	
114	Uscita A - G2: Verde - RGB	Impostare temp. colore (Kelvin)		2 bytes	C - W -	-	absolute c... Basso	
127	Uscita A - G3: Blu - RGB	Commutazione	0/0/3 luce A3 - s... 0/0/3	1 bit	C - W -	-	switch	Basso
128	Uscita A - G3: Blu - RGB	Stato commutazione		1 bit	C R - T -	-	switch	Basso
129	Uscita A - G3: Blu - RGB	Dimmer relativo		4 bit	C - W -	-	dimming c...Basso	
130	Uscita A - G3: Blu - RGB	Valore luminosità	0/3/4 luce A3 - B...0/3/4	1 byte	C - W -	-	percentag... Basso	
131	Uscita A - G3: Blu - RGB	Stato valore luminosità		1 byte	C R - T -	-	percentag... Basso	
135	Uscita A - G3: Blu - RGB	Dimmer temperatura colore		4 bit	C - W -	-	dimming c...Basso	
136	Uscita A - G3: Blu - RGB	Impostare temp. colore (Kelvin)		2 bytes	C - W -	-	absolute c... Basso	
149	Uscita A - G4: Bianco - RGB	Commutazione	0/0/8 luce A3 - s... 0/0/8	1 bit	C - W -	-	switch	Basso
150	Uscita A - G4: Bianco - RGB	Stato commutazione		1 bit	C R - T -	-	switch	Basso
151	Uscita A - G4: Bianco - RGB	Dimmer relativo		4 bit	C - W -	-	dimming c...Basso	
152	Uscita A - G4: Bianco - RGB	Valore luminosità	0/3/5 Bianco1 - ... 0/3/5	1 byte	C - W -	-	percentag... Basso	
153	Uscita A - G4: Bianco - RGB	Stato valore luminosità		1 byte	C R - T -	-	percentag... Basso	
157	Uscita A - G4: Bianco - RGB	Dimmer temperatura colore		4 bit	C - W -	-	dimming c...Basso	
158	Uscita A - G4: Bianco - RGB	Impostare temp. colore (Kelvin)		2 bytes	C - W -	-	absolute c... Basso	

Figura 4.5: Le luci RGB sono gestite dai gruppi G1, G2 e G3, associate al rosso, al verde e al blu e dal gruppo G4 è associato al controllo del bianco

4.2.4 Indirizzi di gruppo

Nel progetto ETS, a ogni gruppo di luci è associato un nome identificativo:

- **A1:** Luci principali della sala grande
- **A2:** Luci laterali della sala grande
- **A3:** Canale RGB delle luci dell'ufficio
- **A4:** Canale bianco delle luci dell'ufficio

In seguito sarà illustrata la configurazione degli indirizzi di gruppo dell'installazione con l'aiuto di immagini ottenute dal progetto ETS.

Questa configurazione rivestirà un ruolo fondamentale nella progettazione del sistema di Smart Lighting. Grazie all'utilizzo della libreria Python xknx, sarà possibile stabilire una connessione all'interfaccia fornita dall'EIBPORT. Sfruttando gli indirizzi di gruppo definiti nella configurazione, sarà quindi possibile inviare pacchetti di scrittura o lettura direttamente alla rete KNX.

4.2.5 Progetto ETS

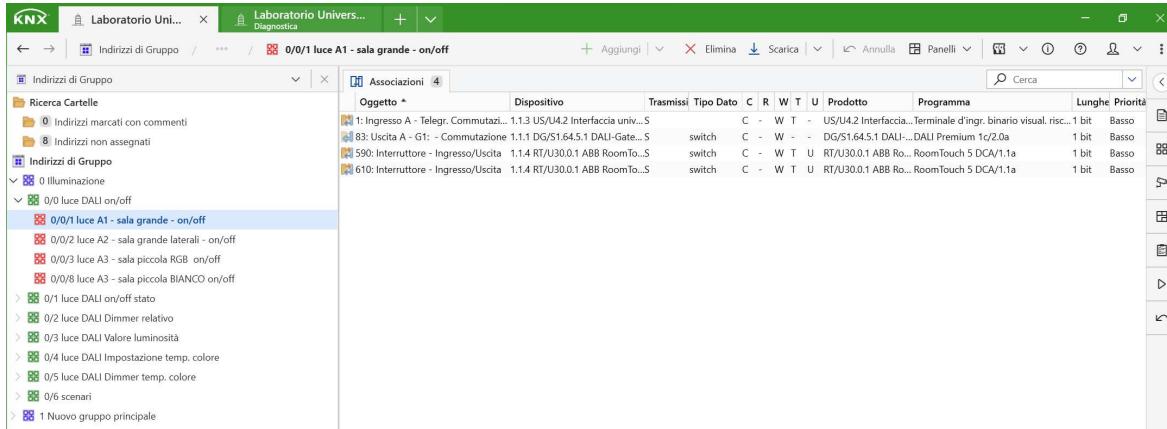


Figura 4.6: Indirizzo di gruppo utilizzato per la commutazione delle luci A1. Così come per le luci A2, un indirizzo di gruppo permette il collegamento con il RoomTouch e con una pulsantiera tradizionale, resa disponibile al sistema dall’interfaccia universale

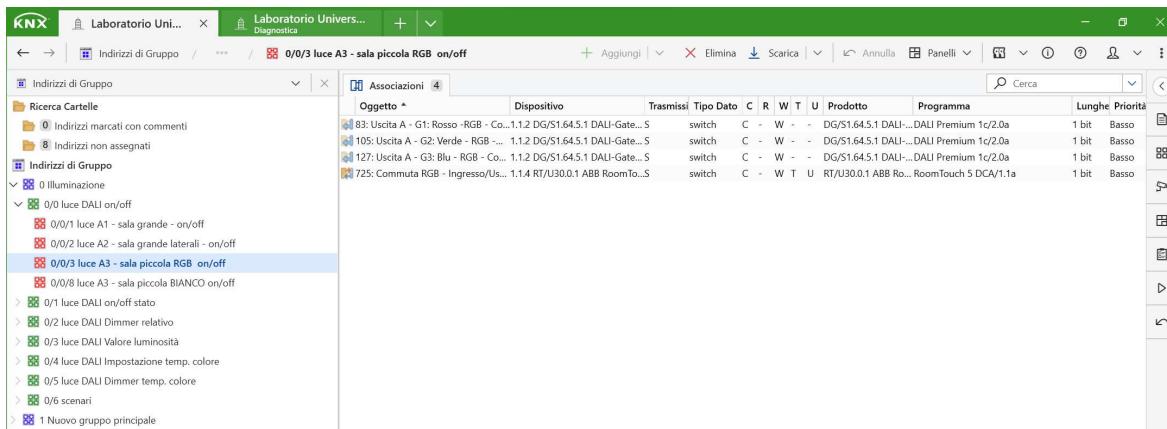


Figura 4.7: Indirizzo di gruppo utilizzato per la commutazione delle luci A3. La commutazione delle luci RGB avviene tramite l’associazione tra il RoomTouch e i tre gruppi responsabili dei colori

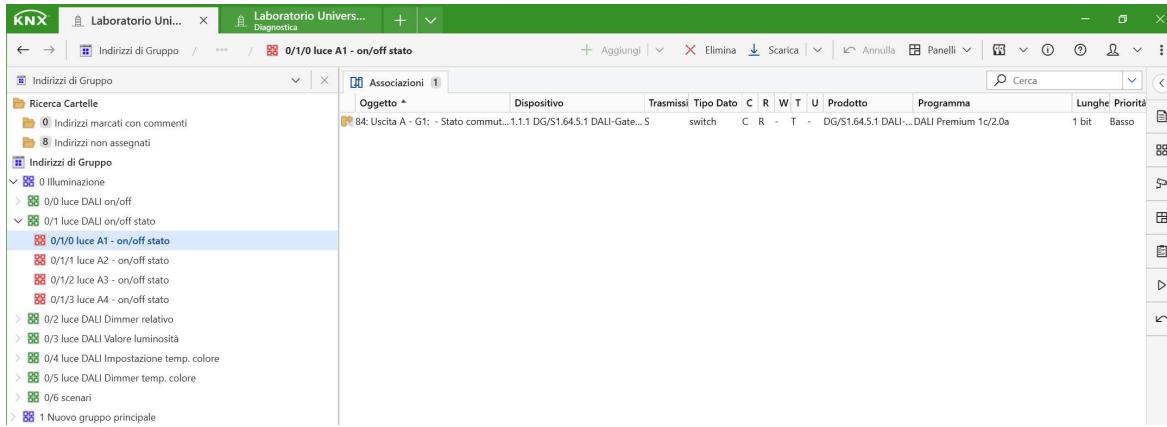


Figura 4.8: Indirizzo di gruppo utilizzato per la lettura dello stato della commutazione della luce A1, esemplificativo per tutti gli indirizzi di gruppo di questo tipo. Per ottenere lo stato delle luci, l'indirizzo di gruppo corrispondente è stato associato solo all'oggetto di gruppo dello stato, essendo utilizzato solo per la lettura del valore

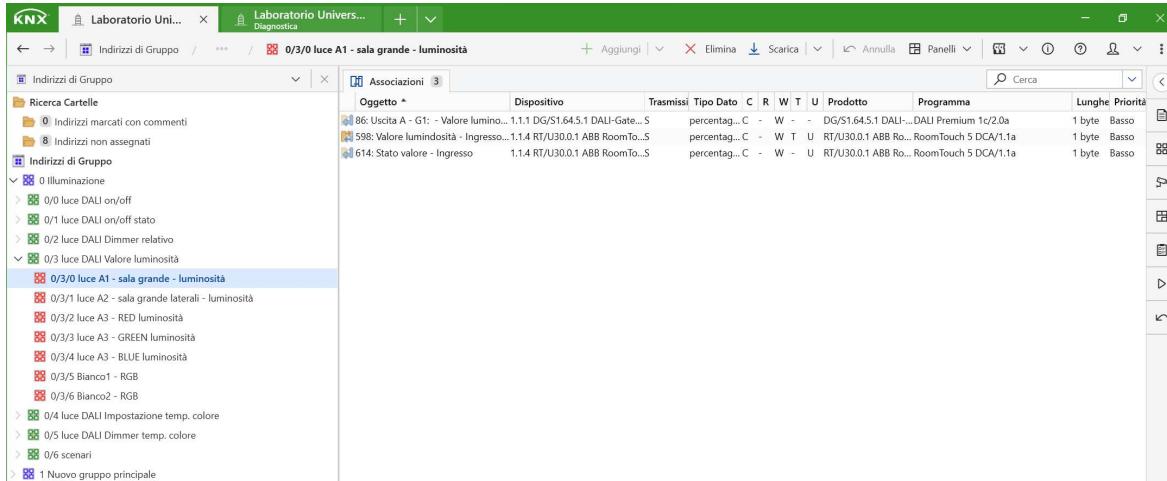


Figura 4.9: Indirizzo di gruppo per il controllo della luminosità, esemplificativo anche per l'indirizzo di gruppo usato per il controllo della temperatura colore. Per il controllo luminosità (e della temperatura colore), di tutte le luci, è associato un indirizzo di gruppo al RoomTouch permettendo al dispositivo un controllo delle luci in tutte le loro funzioni.

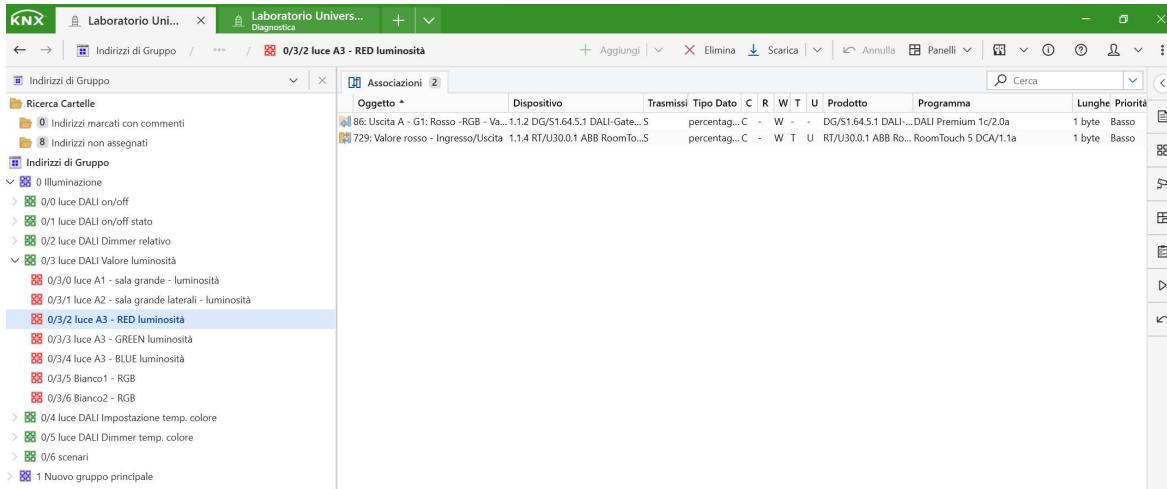


Figura 4.10: Indirizzo di gruppo utilizzato per la selezioni di un valore percentuale del colore rosso per le luci A3, esemplificativo per gli indirizzi di gruppo associati al verde e al blu. Dal RoomTouch è possibile selezionare un colore da una ruota, questa operazione utilizza i tre indirizzi di gruppo che controllano il valore percentuale del rosso, del blu e del verde.

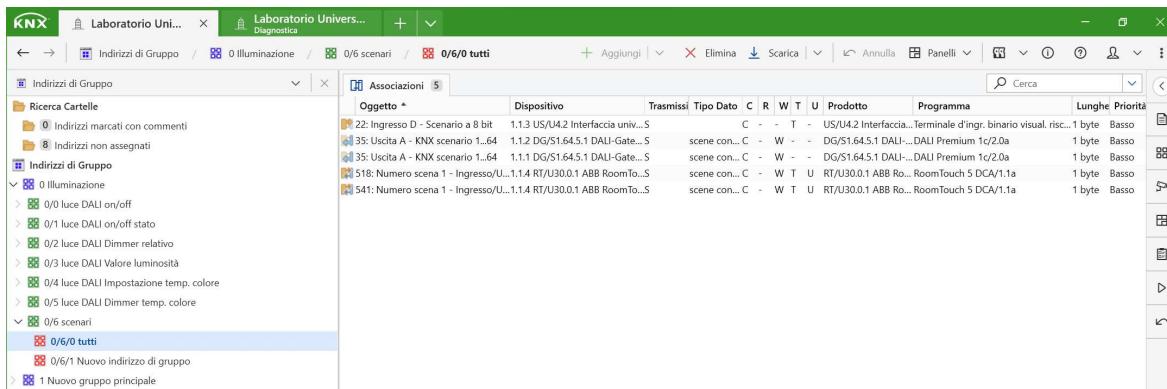


Figura 4.11: Indirizzo di gruppo associato al controllo degli scenari. Tramite il RoomTouch è possibile accendere o spegnere tutte le luci del laboratorio attraverso il collegamento con oggetti di gruppo scenario, resi disponibili dalle interfacce DALI, ovvero oggetti che permettono la commutazione di tutti i gruppi dell’interfaccia. Questo collegamento agisce su entrambe le interfacce, dando la possibilità di accendere o spegnere le luci di entrambi gli ambienti.

4.3 Telecamere

Come precedentemente introdotto, nel laboratorio sono disponibili 3 diverse telecamere: una bullet, una a 360° e una PTZ. Ciascuna di esse ha delle caratteristiche peculiari che verranno esaminate, evidenziato anche le diverse tipologie di configurazione e i software disponibili. Saranno anche presentati i pro e i contro di ciascuna opzione al fine di effettuare una scelta ponderata che si adeguai ai requisiti richiesti dal sistema.

4.3.1 Dahua IPC-HFW4830E-S

Questa telecamera IP appartiene alla categoria delle telecamere bullet, caratterizzate da dimensioni contenute e un campo visivo limitato, solitamente impiegate per finalità di sicurezza. La telecamera è dotata di un cavo Ethernet che, oltre a fornire la connessione di rete, agisce anche come alimentazione attraverso la tecnologia PoE (Power over Ethernet).

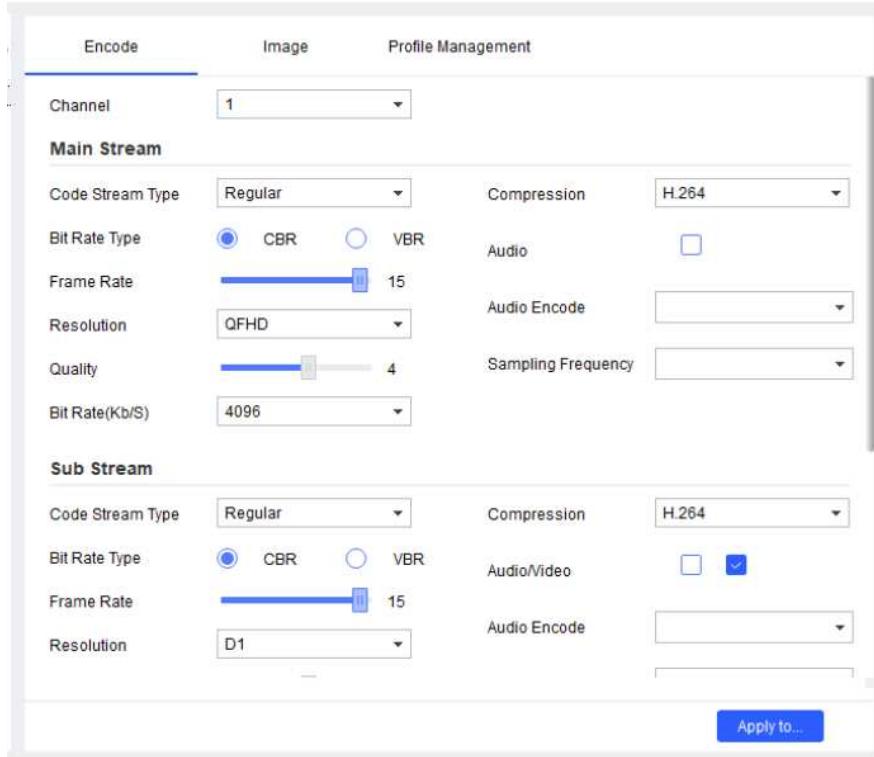


Figura 4.12: Possibili impostazioni del flusso video della telecamera sul software ConfigTool

Per configurare la telecamera dopo averla collegata all’Ethernet, è sufficiente connettere il computer alla stessa LAN e utilizzare il software fornito da Dahua: ConfigTool. L’installazione della telecamera è un processo rapido: dopo aver collegato il cavo e atteso qualche secondo, ConfigTool dovrebbe aver individuato la telecamera tramite la funzione di discovery, effettuata su un segmento specifico della rete.

Una volta individuata la telecamera, è possibile, se necessario, modificare l’IP sia in modo dinamico che statico. Tuttavia, si consiglia di associare un IP statico riservato nella rete.

A questo punto, è possibile procedere con diverse configurazioni. Un aspetto cruciale è assicurarsi che il flusso secondario abbia una risoluzione sufficientemente elevata.

Sebbene la telecamera supporti lo streaming video in 4k, va tenuto presente che tale funzione richiede notevoli risorse computazionali nel sistema. Pertanto, per ottenere una performance ottimale, dovrebbe esser utilizzato il flusso secondario, che offre una risoluzione inferiore ma più gestibile dal punto di vista delle risorse.

Una volta completata la configurazione, le immagini catturate possono essere visualizzate collegandosi direttamente all’IP della telecamera. È possibile farlo tramite una web application resa disponibile dalla telecamera stessa. Alternativamente, è possibile utilizzare la

porta RTSP e connettersi utilizzando software compatibili con questo tipo di connessione, come VLC o, nel sistema proposto, sfruttando le funzionalità offerte dalla libreria Python OpenCV. Nonostante la facilità di installazione e l'alta risoluzione, la telecamera analizzata presenta un campo visivo troppo limitato per essere impiegata efficacemente nella ripresa del laboratorio. Questa restrizione la rende inadatta al monitoraggio dell'area di interesse. Tuttavia, la facilità nell'ottenere il flusso video e il collegamento la rendono essenziale per una fase preliminare di progettazione del sistema.

4.3.2 Kandao Obsidian R

La telecamera fornita da Kandao sfrutta sei telecamere per offrire una visione a 360° dell'ambiente, con una risoluzione che raggiunge i 8k.

Principalmente utilizzata per registrazioni video di alta qualità, offre la possibilità di utilizzare un software di editing specifico per i video a 360°, Kandao Studio, disponibile per Windows e fornito dalla stessa azienda. La telecamera consente anche di catturare foto da ciascuna delle telecamere componenti e di salvarle su microSD.

Questa telecamera può essere collegata tramite Ethernet, necessario per il collegamento con il software, ma non per l'utilizzo in sé. Infatti, la camera può essere alimentata sia tramite un'entrata di 12 volt, oppure mediante l'uso di batterie.

Kandao Live è il software fornito da Kandao che offre la possibilità di configurare i parametri relativi al flusso video e di visualizzare le riprese individuali delle telecamere.

Il collegamento avviene tramite IP, con un computer connesso alla stessa LAN della telecamera. Inoltre, il software consente di avviare una trasmissione live; solo attraverso questa funzione è possibile accedere al flusso video composito a 360° dall'indirizzo RTSP fornito.

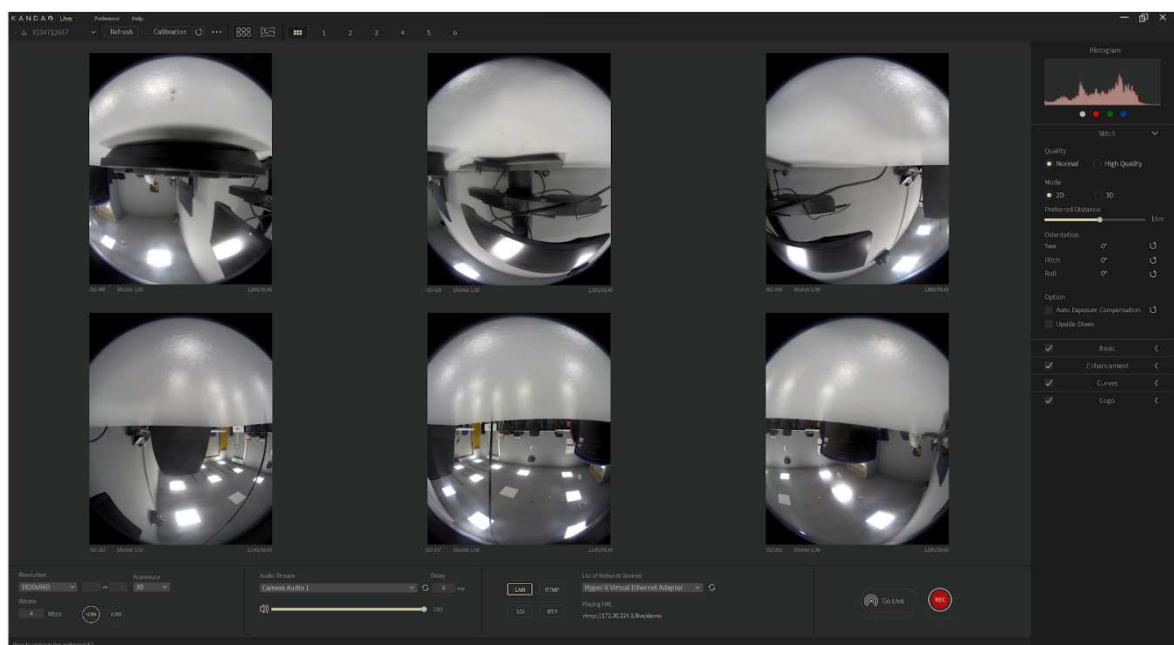


Figura 4.13: Schermata principale del software KandaoLive

Sebbene la capacità di visualizzare il flusso video a 360° e 8k sia un notevole vantaggio per

questa telecamera, è importante sottolineare che la sua concezione mira principalmente alla registrazione video, e non alla sicurezza. Questa differenza di scopo presenta alcune problematiche. L'eccessivo surriscaldamento del dispositivo, insieme all'obbligo di avviare una trasmissione live su KandaoLive per ottenere il flusso video e la non trascurabile latenza causata dalla post-elaborazione dell'immagine, rendono questa telecamera inutilizzabile nel sistema.

4.3.3 PTZIP2M4XWS

Questa telecamera PTZ offre la possibilità di movimento orizzontale, verticale e zoom, rendendola ideale per applicazioni di sicurezza.

L'installazione avviene tramite connessione Ethernet e alimentazione. Per consentire a un dispositivo sulla stessa LAN della telecamera di connettersi, è possibile farlo attraverso un'applicazione mobile o una web application a cui è possibile connettersi tramite l'IP della telecamera. Per la modifica delle impostazioni di rete e relative al flusso video e la configurazione, è disponibile il software HIP2PClient, che consente inoltre di controllare il movimento.

Questa telecamera consente la condivisione sia del flusso primario che secondario su porte RTSP, rispettivamente 11 e 12.

Per permettere alla telecamera di muoversi tramite script Python, è necessario inviare richieste con parametri specifici al suo IP, assicurandosi di eseguire l'autenticazione con le credenziali fornite durante la configurazione iniziale. Queste richieste saranno esaminate in dettaglio quando si discuterà del sistema sviluppato.

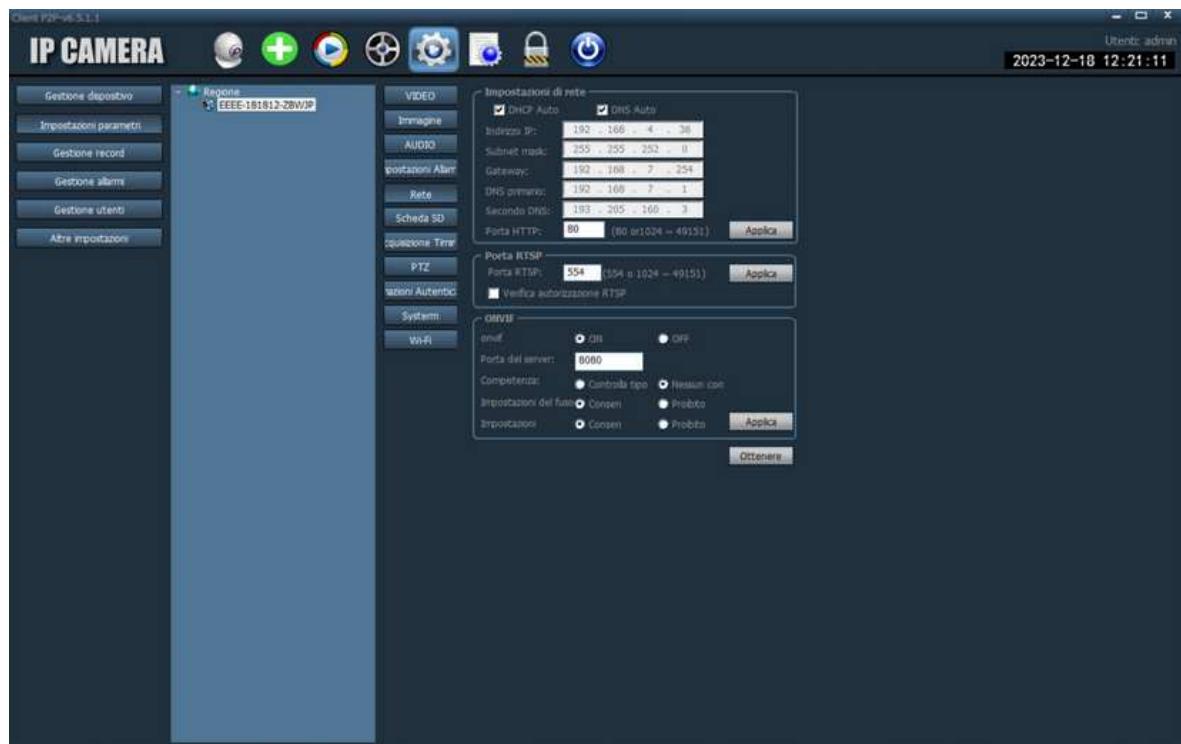


Figura 4.14: Impostazioni della telecamera sul software HiP2P

Questa telecamera, pur essendo più complessa da installare rispetto alla telecamera bullet,

offre un'esplorazione dell'ambiente molto più completa, specialmente quando posizionata in un angolo della stanza.

La sua capacità di connessione senza necessità di software aggiuntivi e l'offerta di un flusso secondario a risoluzione adeguata, la rendono la scelta ideale per il sistema.

È importante sottolineare che, pur necessitando dell'implementazione dei comandi per il movimento, questi operano attraverso richieste asincrone, evitando un impatto significativo sul sistema.

4.3.4 Telecamera scelta per l'integrazione con il sistema

Dopo una fase iniziale di implementazione, durante la quale la telecamera Dahua è stata impiegata per acquisire il flusso video senza richiedere procedure di installazione complesse, si è proceduto con l'integrazione della telecamera PTZ nel sistema. Una volta implementate la maggior parte delle funzionalità legate al flusso video e valutate le caratteristiche precedentemente descritte, questa telecamera è stata selezionata come soluzione ottimale per il monitoraggio dell'ambiente.

4.4 Sistema proposto

Il sistema proposto per il laboratorio è implementato in Python e consente diverse funzionalità chiave:

- **Ottenimento del flusso video:** La telecamera PTZ fornisce il flusso video, consentendo anche il controllo di movimenti e zoom tramite input da tastiera.
- **Interazione con l'illuminazione KNX:** Utilizzando lo scambio di pacchetti, il sistema interagisce con l'illuminazione tramite il network KNX.
- **Suddivisione in Zone:** L'utente può suddividere dinamicamente l'ambiente in diverse zone, conferendo flessibilità e adattabilità al sistema.
- **Rilevamento e Tracciamento delle Persone:** Il sistema rileva le persone nei frame del flusso video, ed esegue il tracciamento grazie all'ausilio delle zone definite.
- **Interfaccia Grafica Intuitiva:** L'interfaccia grafica offre la visualizzazione del video, della rilevazione e delle zone. Inoltre, fornisce pulsanti interattivi per interagire direttamente con il sistema.

4.5 RTSP

Tutte le telecamere utilizzano il protocollo RTSP (Real Time Streaming Protocol) per condividere il proprio flusso video sulla rete. A livello di applicazione, questo protocollo consente il controllo dei dati multimediali in tempo reale, come audio e video. RTSP offre la possibilità di gestire diverse sessioni di trasferimento dati e di selezionare i meccanismi di consegna e il

protocollo di trasferimento desiderato [32].

Per ottenere il flusso video, è necessario che il client si connetta alla porta RTSP del server, nel nostro caso la telecamera tramite un URL così costruito:

```
URL=rtsp://username:password@indirizzo_IP:porta_rtsp
/cam/realmonitor?channel=1&subtype=0&unicast=true.
```

Gli elementi che lo compongono sono:

- **Username e Password:** Tipicamente definiti con valori predefiniti durante la fase iniziale di installazione.
- **Indirizzo IP:** La telecamera potrebbe disporre di un indirizzo IP predefinito fornito dal produttore, modificabile tramite il software di configurazione fornito.
- **Porta RTSP:** Di default è 554, tuttavia il produttore potrebbe assegnare una specifica porta.
- **Channel:** Indica il canale della telecamera.
- **Subtype:** Specifica il flusso video trasmesso, con opzioni per il flusso principale (subtype=0) o il flusso secondario (subtype=1).

4.5.1 Librerie utilizzate

Il sistema sfrutta una serie di funzionalità fornite da diverse librerie, ognuna dedicata a compiti specifici per rispondere alle esigenze del sistema. Di seguito, sono forniti dettagli rilevanti sulle librerie chiave utilizzate

- **XKNX:** Utilizzata per la comunicazione con l'interfaccia IP del sistema KNX. Tramite un oggetto XKNX è possibile instaurare automaticamente una connessione tramite tunneling IP con l'interfaccia, dando la possibilità di eseguire lo scambio di telegrammi con il sistema.
- **Pygame:** L'interfaccia utente interattiva è implementata utilizzando Pygame. Questa libreria consente di integrare comandi dalla tastiera, ad esempio per controllare la telecamera, e di inserire pulsanti per eseguire operazioni più complesse. Pygame è anche responsabile della visualizzazione del flusso video e delle bounding box rilevate dal modello di Object Detection.
- **Opencv:** Utilizzata per il collegamento con le telecamere tramite il protocollo RTSP. OpenCV offre la possibilità di scansionare tutti i frame del flusso, fornendoli all'interfaccia utente per la visualizzazione e al modello di rilevamento per eseguire le operazioni necessarie.
- **Ultralytics:** Questa libreria abilita l'utilizzo del modello di Object Detection YOLOv8. Ricevendo il frame come input, consente di rilevare le bounding box relative alle persone che entrano nel campo visivo della telecamera.

4.6 Implementazione del sistema

Il sistema inizializza tutti i sottosistemi nel suo main. Ciò include la connessione alla telecamera e la creazione dei singleton per l'interfaccia pygame e il tracking.

Una volta stabilita la connessione alla porta RTSP, il sistema entra in un ciclo continuo in cui esegue periodicamente le operazioni di rilevamento. Se vengono individuate persone, viene aggiornato lo stato delle zone: la luce corrispondente viene accesa se vengono rilevate delle persone, altrimenti viene avviato un timer per programmare lo spegnimento se non vengono più rilevate.

Nel frattempo, il sistema rimane in ascolto degli eventi di input dell'utente, che possono includere l'interazione con l'interfaccia, il controllo del movimento della telecamera (nel caso sia PTZ) e la possibile chiusura del programma.

```

if __name__ == '__main__':
    # Started tracker
    tracker = VideoTracker()

    # Get screen resolution
    user32 = ctypes.windll.user32
    screen_width = user32.GetSystemMetrics(0)
    screen_height = user32.GetSystemMetrics(1)

    # OpenCV video capture
    camera_id, controller = get_camera_id()
    cap = cv2.VideoCapture(camera_id)

    # Created Pygame Interface
    interface = PygameInterface("Video_Tracking_System",
                                 screen_width, screen_height)

while True:
    # Read video frame from capture
    ret, frame = cap.read()
    frame = cv2.resize(frame, (screen_width, screen_height))
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Adjust frame to pygame screen
    interface.adjust_frame(frame_rgb)
    pygame_event_actions(interface, tracker, controller)

    # If there aren't people detected, after many frames

```

```
# sends an empty list to the zone control
if tracker.is_started():
    results = tracker.object_detection(frame, interface)
    Zone.update_zone(results)
```

4.6.1 Controllo della telecamera

Il sistema consente la scelta della telecamera attraverso un file di testo, consentendo così un adattamento facile al cambiamento della telecamera.

È possibile specificare l'URL RTSP della telecamera o inserire la stringa "0" per utilizzare la webcam, opzione che si è rivelata molto utile durante lo sviluppo del sistema. Nel caso in cui l'IP corrisponda a quello della telecamera PTZ, vengono creati un oggetto camera, contenente le impostazioni e le credenziali, e un oggetto controller che fornisce i metodi per il suo movimento.



Figura 4.15: Diagramma di classe dell'oggetto camera

Camera.py

L'oggetto Camera ha come variabili di istanza, oltre all'IP della telecamera, l'username e la password necessari per il collegamento a essa. Inoltre, ha i valori predefiniti di velocità di movimento specifici per il modello della telecamera.

Quando un nuovo oggetto è creato, prova a effettuare una richiesta HTTP per verificare se i parametri inseriti sono corretti:

```

except Exception as ex:
    print(ex)
    sys.exit()

```

CameraController.py

Per eseguire i comandi, viene utilizzato lo standard Common Gateway Interface (CGI), che consente a programmi esterni, noti come script CGI, di comunicare con un server web. Tutte le richieste effettuate dal controller, seguono lo standard CGI, che a sua volta fa uso del protocollo HTTP, richiedendo l'autenticazione come passo preliminare.

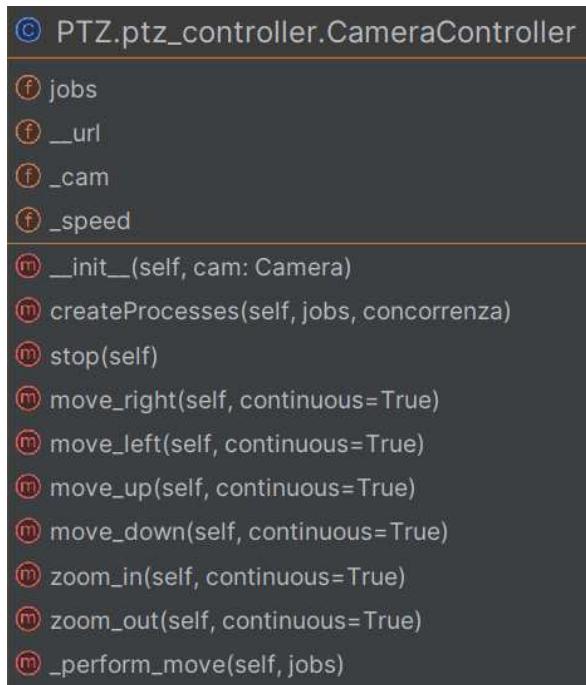


Figura 4.16: Diagramma di classe del controller

Alla creazione di un oggetto Controller 4.16, vengono avviati due sottoprocessi per gestire le richieste per effettuare i comandi della telecamera. Questo approccio consente al programma di eseguire le richieste per il movimento in parallelo anziché attendere in modo sequenziale. La classe mette a disposizione tutti i metodi per gli spostamenti, lo zoom e l'arresto attraverso funzioni dedicate. Quando uno di questi metodi viene chiamato, l'operazione viene inserita in una JoinableQueue, una struttura dati per la condivisione di dati in contesti paralleli. Il metodo utilizzato per il movimento a destra, di seguito illustrato, crea un dizionario di parametri che specifica il tipo di azione che la telecamera dovrà effettuare, questi sono:

- **Step:** Può assumere i valori 0 o 1, a seconda che il movimento debba essere un passo o continuo.
- **Act:** Definisce la direzione del movimento, in questo caso specifico, indica il movimento verso destra.

- **Speed:** Specifica la velocità del movimento, il cui valore è standardizzato dalla telecamera.

```
def move_right(self , continuous=True):
    step = 0
    if not continuous:
        step = 1

    params = {
        '_step': str(step),
        '_act': 'right',
        '_speed': str(self._speed)
    }
    self.jobs.put(params)
```

Successivamente, il dizionario viene inserito nella coda, da cui sarà recuperato da uno dei sottoprocessi per eseguire una richiesta HTTP, sfruttando lo standard CGI, utilizzando i parametri ottenuti.

```
def _perform_move(self , jobs):
    """perform movement specified by the parameters
    in the jobs queue"""
    while True:
        try:
            params = jobs.get()
            requests.get(self._url , params ,
                         auth=HTTPBasicAuth(self._cam._user ,
                                             self._cam._password))
        except Exception as e:
            print(e)
```

4.6.2 Tracciamento

La classe VideoTracker 4.17, che gestisce il tracciamento delle persone, è implementata come un singleton, assicurando che esista una sola istanza della classe nell'intero programma. Quando è istanziata per la prima volta, carica un modello di YOLO pre addestrato, pronto per la rilevazione e avvia 5 sottoprocessi che serviranno per l'esecuzione del rilevamento.

La rilevazione è avviata da un pulsante sull'interfaccia utente, quando questo è premuto è chiamata la funzione start che cambia lo stato del tracker, che permette al rilevamento di cominciare. Allo stesso modo il tracciamento può essere interrotto in qualsiasi momento dallo stesso pulsante.

Il metodo che implementa la logica per il rilevamento delle persone, avviene ogni tre frame per limitare l'impatto sulle prestazioni del sistema, garantendo la fluidità dello streaming video. In questo processo, ciascun frame viene inserito in una coda condivisa per essere elaborato in

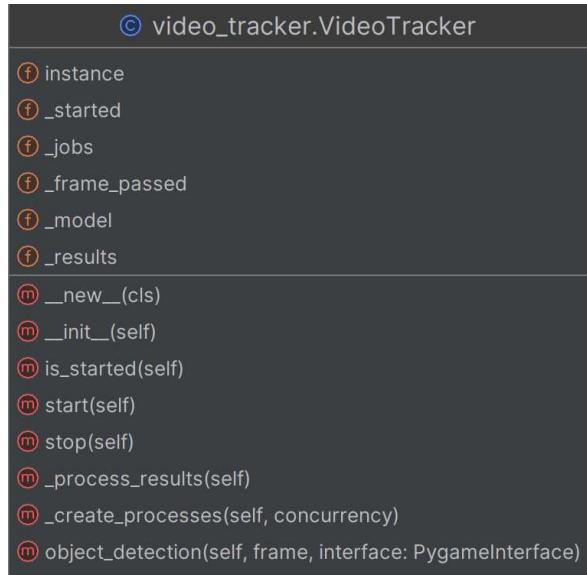


Figura 4.17: Diagramma di classe del tracker

parallelo dal modello YOLO. Una volta ottenuti i risultati, che comprendono le coordinate spaziali delle bounding box delle persone individuate, queste informazioni vengono utilizzate per visualizzarle dinamicamente sull'interfaccia grafica. Inoltre, le dimensioni delle bounding box vengono sfruttate per calcolare i rispettivi centri. La lista dei centri viene quindi restituita per aggiornare dinamicamente le zone definite nell'ambiente.

```

def object_detection(self , frame , interface: PygameInterface):
    """identifies the people that enters the frame,
    creates a box and put the center in the list"""

    # The detection is made one time each 3 frame
    self._frame_passed += 1

    if self._frame_passed == 2:
        self._frame_passed = 0
        self._jobs.put(frame)
        centers = []

    if not self._results.empty():
        points = self._results.get()

    for point in points:
        # Each person detected is represented
        as a point that is the center of a box
        x, y, w, h = point
        center = ((x + w / 2) , (y + h / 2))

```

```

        centers.append(center)
        interface.create_box(x, y, w, h)
    return centers
else:
    return []

```

Il modello, che esegue le rilevazioni sui frame in parallelo, restituisce una serie di bounding box, che, come visto precedentemente 3.2.2, oltre alle dimensioni spaziali, contengono anche il punteggio di fiducia e la classe degli oggetti rilevati. I parametri di classe e fiducia vengono utilizzati per filtrare le rilevazioni, mantenendo solo gli oggetti di classe 0 (persone) con un punteggio di fiducia superiore a 0.5. Nel contempo, le dimensioni delle bounding box vengono inserite in una coda dei risultati.

```

for result in model_results:

    for box in result.boxes.data.tolist():
        x, y, w, h, score, class_id = box

        # class_id 0 is the person id,
        # score is the accuracy of the prediction
        if class_id == 0 and score >= 0.5:
            results_box.append((x, y, w, h))
            self._results.put(results_box)

```

4.6.3 Interfaccia utente

L’interfaccia grafica del sistema è gestita da un singleton che sfrutta le funzionalità offerte dalla libreria Pygame. L’interfaccia da la possibilità di visualizzare l’output della telecamera, aggiornando di frame in frame l’interfaccia, per mostrare ad esempio le zone create dinamicamente o le persone rilevate dal modello. Inoltre, gestisce la rilevazione di tutti gli eventi da tastiera.

Durante la prima istanziazione, vengono specificate le dimensioni e il nome della finestra in cui saranno visualizzati i pulsanti e lo streaming video. Vengono creati anche tutti gli oggetti dell’interfaccia, i quali verranno successivamente disegnati durante l’aggiornamento del frame. Ad esempio, il metodo che inserisce i pulsanti sull’interfaccia disegna dei rettangoli delle dimensioni definite dalla risoluzione dello streaming e colloca il testo del pulsante su questo rettangolo.

```

def draw_buttons(self, buttons_to_draw):
    """draws the buttons on the screen"""
    margin_x = self.margin_left
    margin_y = self.margin_top

    for button in buttons_to_draw:

```

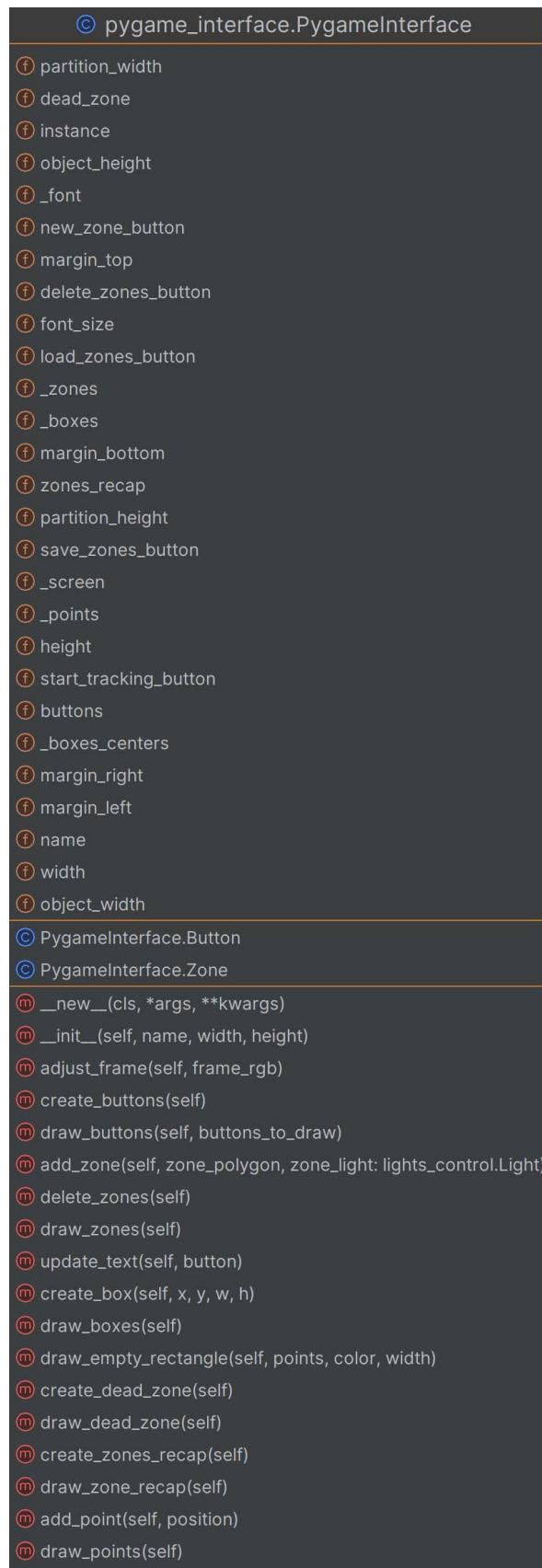


Figura 4.18: Diagramma di classe dell’interfaccia utente

```
pygame.draw.rect(self._screen, (255, 255, 255), button.rect)
self._screen.blit(button.rendered_text_default,
                  (button.rect.x + margin_x,
                   button.rect.y + margin_y * 1.5))
```

L’interfaccia è divisa in tre zone: sulla sinistra, sono presenti i pulsanti per le varie operazioni possibili, implementati come oggetti Button.

Sulla destra, c’è un riepilogo delle zone che tiene traccia degli oggetti Zone creati. Ogni zona ha come variabili di istanza il nome, la luce associata (con lo stato di commutazione mostrato), e un colore identificativo, assegnato casualmente.

Lo stato della luce presente nel riepilogo viene aggiornato al momento dell’aggiornamento della stessa, in modo da riflettere accuratamente lo stato di illuminazione attuale di tutte le zone.

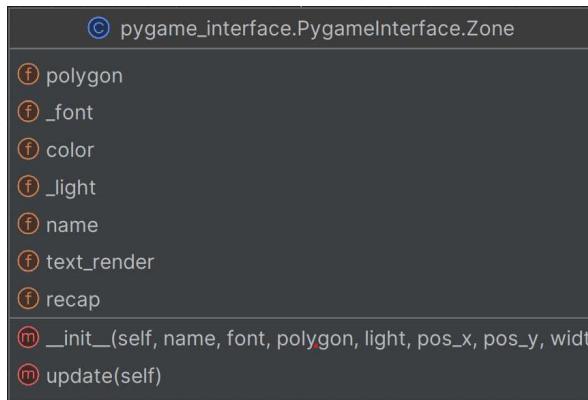


Figura 4.19: Diagramma di classe delle zone disegnate

Fra le due sezioni laterali è mostrato il flusso video, le zone disegnate e le bounding box delle persone rilevate. Quest’ultime sono disegnabili attraverso la creazione di poligoni, effettuata nel metodo seguente:

```
def draw_empty_rectangle(self, points, color, width):
    """draws a polygon using the points"""
    current_point = first_point = points[0]
    points.remove(first_point)
    for point in points:
        pygame.draw.line(self._screen, color, current_point, point, width)
        current_point = point
    pygame.draw.line(self._screen, color, current_point, first_point, width)
```

Nella stessa sezione, sono visualizzati i punti cliccati durante la creazione della zona come cerchi rossi. Questo miglioramento visivo facilita l’interazione con l’utente, consentendo una chiara rappresentazione dei vertici utilizzati per definire le zone.

```
def draw_points(self):
```

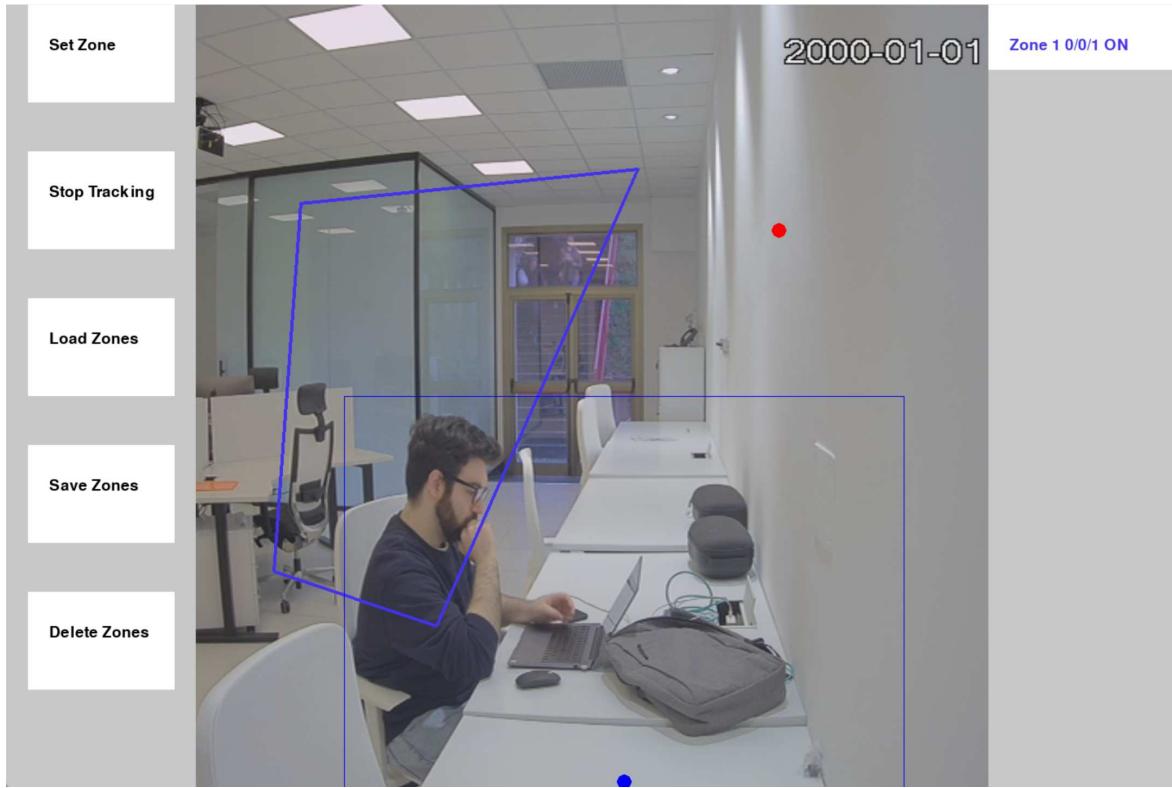


Figura 4.20: Interfaccia grafica del sistema

```
for point in self._points:
    pygame.draw.circle(self._screen, (255, 0, 0),
                       point, self.font_size/3)
```

Le dimensioni di tutti gli elementi dell’interfaccia sono calcolate in relazione alla risoluzione del flusso video, garantendo così che pulsanti, margini e dimensioni del testo e delle sezioni si adattino in modo responsivo.

Gestione degli eventi di input

La libreria Pygame permette di catturare tutti gli eventi di input e gestirli attraverso una coda. A ogni iterazione del ciclo del main, viene chiamato il metodo per la gestione degli eventi, che itera attraverso tutti gli eventi di input, associando diverse funzionalità a ciascun tipo di input atteso.

Gli eventi catturati possono essere di tre tipi:

- **pygame.KEYDOWN:** Rappresenta il momento in cui un tasto viene premuto. Se è utilizzata di la telecamera PTZ, questo evento è associato ai comandi relativi a essa. In alternativa, viene utilizzato esclusivamente per terminare l’applicazione.
- **pygame.KEYUP:** Rappresenta il momento in cui un tasto viene rilasciato, un’azione che viene considerata solo durante l’uso della telecamera PTZ. Questo evento è utilizzato per interrompere il movimento della telecamera, quando viene rilasciato il tasto precedentemente premuto per imparirlo.

- **pygame.MOUSEBUTTONDOWN:** Rappresenta il momento in cui viene premuto il tasto sinistro del mouse. Questo tipo di evento è sfruttato per le interazioni con l’interfaccia utente, come il click sui pulsanti.

Le azioni specifiche compiute quando viene cliccato un tasto sono dettagliate nella tabella 4.1.

La classe pygame.Event fornisce le coordinate del click effettuato, consentendo di determinare con quale pulsante dell’interfaccia l’utente sta interagendo.

Ciascun pulsante è implementato come un oggetto della classe Button 4.21. Ogni oggetto della classe ha come variabili di istanza sia il testo predefinito, associato al pulsante all’avvio, sia quello da visualizzare una volta cliccato.

Il metodo illustrato di seguito viene eseguito ogni volta che si interagisce con il pulsante, permettendo di scambiare i due testi, fornendo un feedback al click dell’utente. Ad esempio, questo viene utilizzato quando si interagisce con il pulsante che avvia il tracking, cambiando il testo da "Start Tracking" a "Stop Tracking".

```
def swap_texts(self):
    """swap the default text with the changed one if possible"""

    if self.rendered_text_changed:
        new_text = self.rendered_text_default
        self.rendered_text_default = self.rendered_text_changed
        self.rendered_text_changed = new_text
```

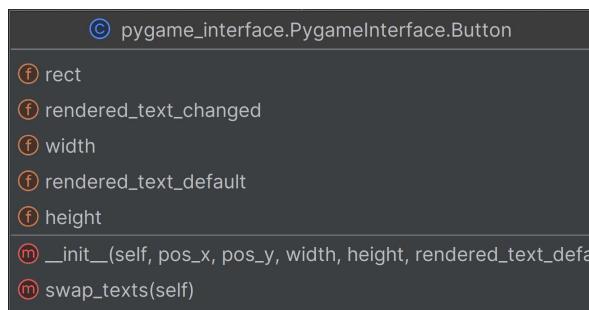


Figura 4.21: Diagramma di classe del pulsante

La gestione degli eventi legati all’interazione con l’interfaccia attraverso il click del mouse è mostrata più nel dettaglio nel diagramma di flusso 4.22.

4.6.4 Gestione delle zone

La gestione delle zone avviene attraverso la classe Zone 4.23, che tiene conto di quante istanze della classe sono state create e di quante persone ci sono in ogni zona creata. Ogni oggetto è composto da un poligono e una luce associata alla zona.

Per gestire tutte le zone istanziate, sono forniti metodi di classe che consentono di eseguire operazioni su di esse. Ad esempio, è possibile salvarle in un file di testo o cancellare tutte le istanze della classe, rimuovendo così le zone dall’interfaccia.

pygame.KEYDOWN	
Tasto premuto	Azione
Q	Uscita dal programma
W	Movimento su
A	Movimento a destra
S	Movimento a sinistra
D	Movimento giù
Freccia su	Zoom in
Freccia giù	Zoom out

pygame.KEYUP	
Tasto premuto	Azione
Qualsiasi	Il movimento è interrotto

Tabella 4.1: Tabella degli input inseribili da tastiera

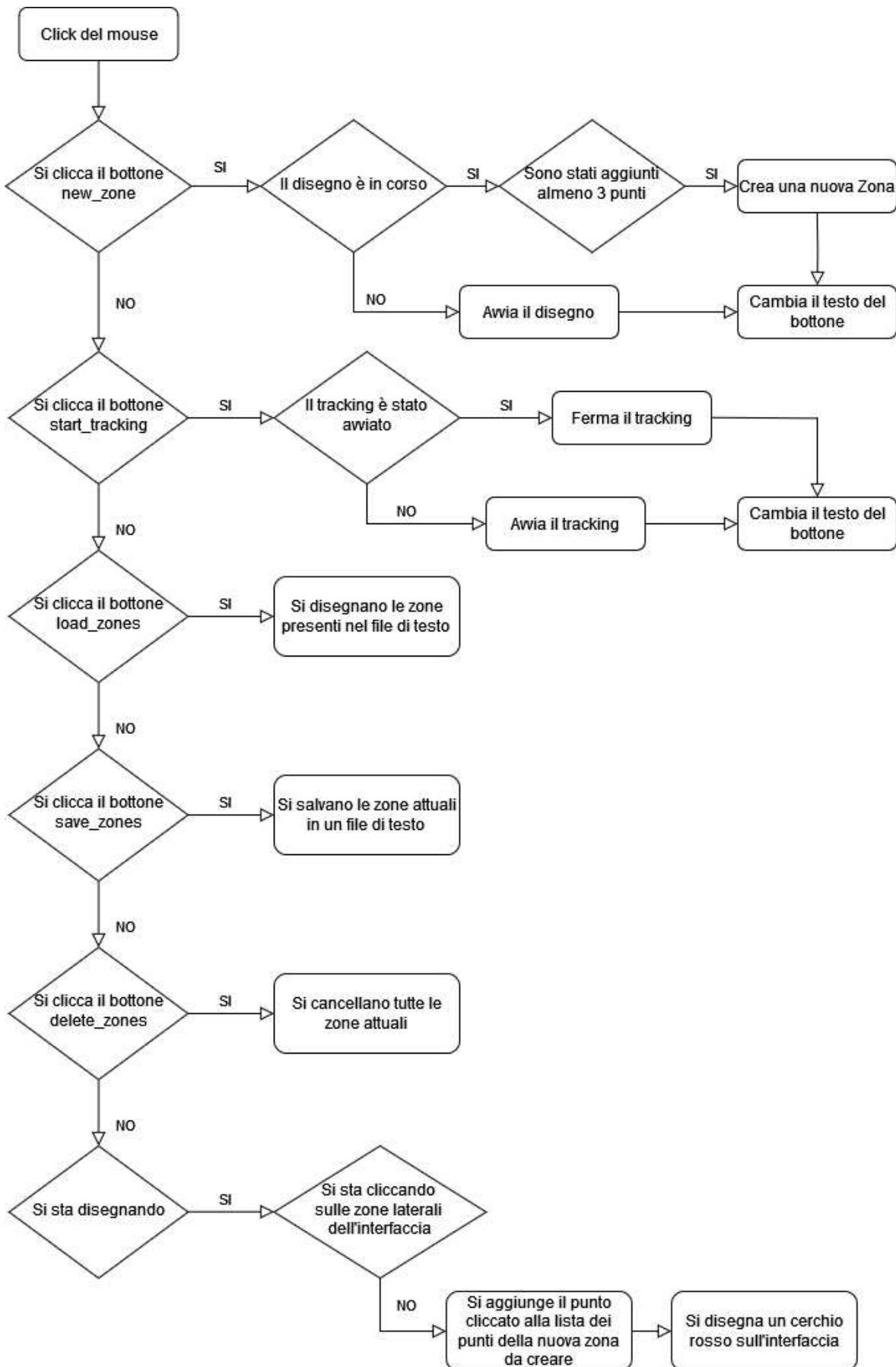


Figura 4.22: Diagramma di flusso per l'interazione da mouse con l'interfaccia grafica

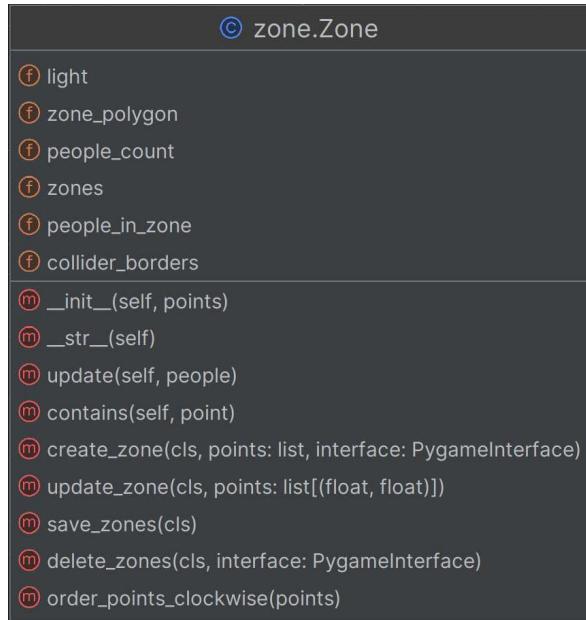


Figura 4.23: Diagramma di classe della zona

Il metodo per la creazione di nuove zone prende in input una serie di punti, li ordina in senso orario e utilizza tali punti per creare un oggetto shapely.Polygon, classe che offre diverse funzionalità per la gestione di poligoni. Una volta creato il poligono, viene effettuato un controllo su tutte le zone per verificare se quella che si sta cercando di creare è equivalente a una già esistente. Se si tratta di una nuova zona, questa è disegnata sull'interfaccia e l'istanza della classe viene creata.

```

@classmethod
def create_zone(cls, points: list, interface: PygameInterface):
    """ creates a new Zone instance """
    ordered_points = cls.order_points_clockwise(points)
    new_zone_polygon = Polygon(ordered_points)

    for zone in Zone.zones:
        if zone.zone_polygon == new_zone_polygon:
            print("Zone Already Created")
            return

    new_zone = Zone(points)
    new_zone.zone_polygon = new_zone_polygon
    new_zone.people_count = 0
    new_zone.light = interface.get_light()
    Zone.zones.append(new_zone)
    interface.add_zone(new_zone.zone_polygon, new_zone.light)

    new_zone = Zone(points)
    print("New Zone Created" + str(new_zone))
    cls.zones[new_zone] = new_zone.people_count
    interface.add_zone(new_zone.zone_polygon, new_zone.light)

```

Il metodo per aggiornare le zone è invocato nel main dopo aver ottenuto i risultati del rilevamento. In input, questo metodo riceve i punti corrispondenti alle persone rilevate.

Per ogni punto, viene verificato se esso si trova all'interno di una zona e viene ricalcolato il

numero di persone attualmente rilevate in ognuna di esse.

Se il risultato di questo calcolo differisce dallo stato attuale delle zone, viene eseguito un aggiornamento.

```
@classmethod
def update_zone(cls, points: list[(float, float)]):

    """check how many people are in each zone and manages changes"""
    cls.people_in_zone.clear()
    for zone in cls.zones.keys():
        cls.people_in_zone[zone] = 0
        if points:
            for point in points:
                if zone.contains(point):
                    cls.people_in_zone[zone] += 1
                else:
                    cls.people_in_zone[zone] = 0

    for zone in cls.people_in_zone.keys():
        if cls.people_in_zone[zone] != cls.zones[zone]:
            zone.update(cls.people_in_zone[zone])
```

La classe offre inoltre due metodi di istanza:

- **Update:** Questo metodo aggiorna il contatore locale delle persone e la luce associata. Se è rilevata la presenza di qualcuno, accende la luce. In caso contrario, se non sono più rilevate persone nella zona, avvia lo spegnimento della luce.
- **Contains:** Questo metodo verifica se una persona è all'interno della zona. Prende come input un punto, che rappresenta il centro della bounding box rilevata e crea un quadrato di dimensioni 100x100 con il punto fornito come centro. Utilizzando il metodo "intersect" offerto dalla classe shapely.Polygon, si verifica se il quadrato interseca la zona.

```
def contains(self, point):
    """return True if the polygon intersects
    the square created by the center given"""

    x, y = point
    borders = Zone.collider_borders
    collide = Polygon([(x - borders, y - borders),
                      (x - borders, y + borders),
                      (x + borders, y + borders),
                      (x + borders, y - borders)])
    return self.zone_polygon.intersects(collide)
```

4.6.5 Gestione dell'illuminazione

La gestione dell'illuminazione KNX è affidata alla classe Light 4.24, la quale tiene traccia di tutte le luci inizializzate, allo scopo di assegnare un indirizzo di gruppo a ciascuna di esse in modo sequenziale.

La classe utilizza una proprietà per lo stato, implementata come un evento, che restituisce "ON" se l'evento è impostato, "OFF" altrimenti.

La classe fornisce due metodi di istanza per la commutazione, entrambi utilizzano un metodo asincrono per l'aggiornamento che comunica con KNX. Inoltre, alla prima istanziazione di ogni oggetto, è avviato un sottoprocesso responsabile dello spegnimento della luce.



Figura 4.24: Diagramma di classe della luce

Il metodo chiamato per accendere una luce effettua un controllo sul suo stato. Se la luce è spenta, chiama il metodo per aggiornarla. Altrimenti, verifica se l'evento non è impostato. Questa condizione si verifica nel caso in cui è stato avviato uno spegnimento e, impostando l'evento, si permette la sua interruzione.

```

def light_on(self):
    """switch on the light if the light is off,
    if the timer is still running, set the state"""

    if self.state == "OFF":
        asyncio.run(self.lights_update(1))

```

```
        elif not self._stop_light_off_event.is_set():
            self._stop_light_off_event.set()
```

Il metodo chiamato per spegnere una luce, dopo aver verificato che la luce è accesa, rilascia l'evento, permettendo così al sottoprocesso che gestisce lo spegnimento di iniziare la sua esecuzione.

Quest'ultimo, una volta che l'evento è stato rilasciato, verifica ogni secondo, per un totale di 8 secondi, se l'evento è impostato, indicando un tentativo di accensione della luce. Se nessun tentativo è stato effettuato durante gli 8 secondi, il processo aggiorna lo stato della luce.

```
def light_off_timer(self):
    """8 seconds timer that starts if the event is cleared.
    After each second check if the light tried to be switched on"""

    while True:
        seconds_passed = 0
        # if the event was cleared

        while not self._stop_light_off_event.is_set():
            time.sleep(1)
            seconds_passed += 1

            # if the light wasn't requested to be switched on
            # and 8 seconds have passed
            if not self._stop_light_off_event.is_set()
                and seconds_passed == 8:
                asyncio.run(self.lights_update(0))
                # Becomes idle
                self._stop_light_off_event.set()
                seconds_passed = 0
```

Il metodo per l'aggiornamento delle luci sfrutta le funzionalità della libreria xknx. Mediante un oggetto XKNX, è possibile connettersi automaticamente all'interfaccia IP e inviare telegrammi utilizzando l'indirizzo di gruppo appropriato.

Si crea un oggetto Switch, che rappresenta un Datapoint booleano, e gli vengono assegnati gli indirizzi di gruppo per la commutazione della luce, oltre a un indirizzo di gruppo per leggere lo stato. In base al valore fornito in input al metodo, l'oggetto viene utilizzato per accendere o spegnere la luce, tramite un metodo asincrono, aggiornando di conseguenza il valore della proprietà.

```
async def lights_update(self, new_state):
    """turns the light according to the state and updates it"""
    try:
        async with XKNX() as xknx:
```

```
s = Switch(xknx, "switch", self.address, self.status_address)
if new_state:
    await s.set_on()
    print(f"{self.address}", "on")
    self.state = True
else:
    await s.set_off()
    print(f"{self.address}", "off")
    self.state = False

except Exception as e:
    print("xknx Error:", e)
    sys.exit()
```

Capitolo 5

Conclusione

L’implementazione di un sistema di Smart Lighting può apportare numerosi vantaggi, specialmente attraverso l’utilizzo del sistema KNX, che offre flessibilità nella selezione e nell’utilizzo dei dispositivi. In questo lavoro di tesi, è stato utilizzato il modello di Object Detection YOLO, in quanto progettato per garantire l’elaborazione in real-time dei frame video e allo stesso tempo un tracking preciso e affidabile dell’oggetto di interesse nella scena.

Il sistema proposto permette un’interazione efficace di queste tecnologie, e, in questo lavoro di tesi, è stata utilizzata una telecamera PTZ che offre una visione completa e la possibilità di esplorazione dell’area, grazie all’implementazione dei comandi per muoverla, che permette al suo punto di vista di comprendere tutto l’ambiente. La creazione dinamica delle zone mediante un’interfaccia utente appositamente progettata e realizzata, consente un’adattabilità a diverse configurazioni ambientali e orientamenti della telecamera.

Inoltre, grazie alla libreria xknx, il sistema può interfacciarsi con diverse installazioni tramite qualsiasi interfaccia IP, a condizione che gli indirizzi di gruppo delle luci seguano un formato specifico.

Un possibile miglioramento potrebbe consistere nell’implementare un meccanismo per ottenerne informazioni direttamente dal progetto ETS, consentendo al sistema di adattarsi in modo più dinamico a diverse configurazioni senza avere prerequisiti troppo stringenti.

Bibliografia

- [1] KNX Association. *KNX.org*. URL: <https://www.knx.org/knx-it/per-la-tua-casa/vantaggi/>.
- [2] Moe Soheilian, Géza Fischl e Myriam Aries. “Smart lighting application for energy saving and user well-being in the residential environment”. In: *Sustainability* 13.11 (2021), p. 6198.
- [3] Evangelos-Nikolaos D Madias et al. “The effect of artificial lighting on both biophilic and human-centric design”. In: *Journal of Building Engineering* 76 (2023), p. 107292.
- [4] Jorge Higuera, Aleix Llenas e Josep Carreras. “Trends in smart lighting for the Internet of Things”. In: *arXiv preprint arXiv:1809.00986* (2018).
- [5] Ivan Chew et al. “Smart lighting: The way forward? Reviewing the past to shape the future”. In: *Energy and Buildings* 149 (2017), pp. 180–191.
- [6] Man-Lin Wu, Che-Min Kung e Yi-Nan Lin. “DALI-2 Intelligent Lighting Control System”. In: *2020 International Symposium on Computer, Consumer and Control (IS3C)*. IEEE. 2020, pp. 158–161.
- [7] Renesas. *Digital Addressable Lighting Interface (DALI)*. 2014. URL: <https://www.eeweb.com/digital-addressable-lighting-interface-dali/> (visitato il 05/03/2024).
- [8] Jianfeng Wang. “Zigbee light link and its applications”. In: *IEEE Wireless Communications* 20.4 (2013), pp. 6–7.
- [9] Vladimir Tanasiev et al. “Energy-efficient solution for smart lighting through iot”. In: *2021 10th International Conference on ENERGY and ENVIRONMENT (CIEM)*. IEEE. 2021, pp. 1–4.
- [10] Susan Hilbolling et al. “Sustaining complement quality for digital product platforms: A case study of the Philips Hue ecosystem”. In: *Journal of Product Innovation Management* 38.1 (2021), pp. 21–48.
- [11] Alper Yilmaz, Omar Javed e Mubarak Shah. “Object tracking: A survey”. In: *Acm computing surveys (CSUR)* 38.4 (2006), 13–es.
- [12] Fei Chen et al. “Visual object tracking: A survey”. In: *Computer Vision and Image Understanding* 222 (2022), p. 103508.

- [13] Zahra Soleimanitaleb, Mohammad Ali Keyvanrad e Ali Jafari. “Object tracking methods: a review”. In: *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE. 2019, pp. 282–288.
- [14] Rachna Verma. “A review of object detection and tracking methods”. In: *International Journal of Advance Engineering and Research Development* 4.10 (2017), pp. 569–578.
- [15] Yong Jae Lee, Jaechul Kim e Kristen Grauman. “Key-segments for video object segmentation”. In: *2011 International Conference on Computer Vision* (2011), pp. 1995–2002. URL: <https://api.semanticscholar.org/CorpusID:3185202>.
- [16] Rens van de Schoot et al. “Bayesian statistics and modelling”. In: *Nature Reviews Methods Primers* 1.1 (2021), p. 1.
- [17] Bohyung Han et al. “Kernel-based bayesian filtering for object tracking”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. IEEE. 2005, pp. 227–234.
- [18] Carlos Henrique Costa Ribeiro. “A tutorial on reinforcement learning techniques”. In: *Supervised Learning Track Tutorials of the 1999 International Joint Conference on Neuronal Networks*. 1999.
- [19] Zhe Chen, Zhibin Hong e Dacheng Tao. “An experimental survey on correlation filter-based tracking”. In: *arXiv preprint arXiv:1509.05520* (2015).
- [20] Bo Li et al. “Siamrpn++: Evolution of siamese visual tracking with very deep networks”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4282–4291.
- [21] Goutam Bhat et al. “Learning discriminative model prediction for tracking”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6182–6191.
- [22] Yao Sui et al. “Visual tracking via subspace learning: A discriminative approach”. In: *International Journal of Computer Vision* 126 (2018), pp. 515–536.
- [23] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Giu. 2016.
- [24] Keiron O’Shea e Ryan Nash. “An introduction to convolutional neural networks”. In: *arXiv preprint arXiv:1511.08458* (2015).
- [25] Yanwei Pang et al. “Convolution in convolution for network in network”. In: *IEEE transactions on neural networks and learning systems* 29.5 (2017), pp. 1587–1597.
- [26] Muhammad Hussain. “YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection”. In: *Machines* 11.7 (2023), p. 677.

- [27] Juan Terven, Diana-Margarita Córdova-Esparza e Julio-Alejandro Romero-González. “A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas”. In: *Machine Learning and Knowledge Extraction* 5.4 (2023), pp. 1680–1716.
- [28] KNX Association. *KNX Specifications*. 2013. URL: https://my.knx.org/it/shop/knx-specifications?product_type=knx-specifications.
- [29] Nils Bjorck et al. “Understanding batch normalization”. In: *Advances in neural information processing systems* 31 (2018).
- [30] Ultralytics. *YOLOv8: A New State-of-the-Art Computer Vision Model*. 2023. URL: <https://yolov8.com/> (visitato il 24/02/2024).
- [31] ABB. *DG/S1.64.5.1 / ABB*. 2024. URL: <https://new.abb.com/products/it/2CDG110273R0011/dg-s1-64-5-1> (visitato il 01/03/2024).
- [32] Henning Schulzrinne, Anup Rao e Rob Lanphier. *RFC2326: Real time streaming protocol (RTSP)*. 1998.

