Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Sistema di rilevazione persone con Frigate, MQTT e notifiche Telegram su Raspberry Pi 4 per il laboratorio IOT dell'Icar Cnr di Cosenza

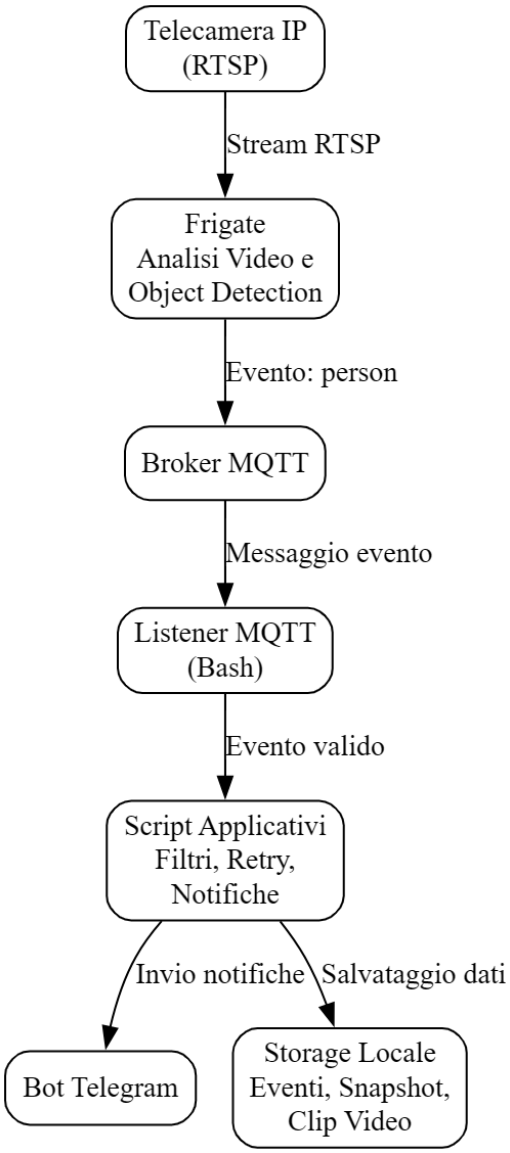
Antonio Francesco Gentile[1], Davide Macrì[2], Emilio Greco[3]

RT-ICAR-CS-26-01 **Gennaio 2026**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 8-9C, 87036 Rende, Italy, URL: www.icar.cnr.it
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: www.icar.cnr.it
– Sezione di Palermo, Via Ugo La Malfa, 153, 90146 Palermo, URL: www.icar.cnr.it

Contesto, obiettivi e principi di progetto

Il progetto descritto in questa relazione riguarda la realizzazione di un sistema di videosorveglianza intelligente basato su analisi video automatica, pensato per funzionare su hardware embedded a basso consumo, in particolare un Raspberry Pi 4. L'obiettivo principale è quello di rilevare la presenza di persone tramite una telecamera IP, generare eventi affidabili e notificare l'utente in modo tempestivo e documentato, riducendo al minimo falsi positivi e rumore informativo.
Il sistema nasce dall'esigenza di avere una soluzione autonoma, controllabile e trasparente, che non dipenda da servizi cloud proprietari per l'elaborazione delle immagini e che consenta all'utente di mantenere il pieno controllo sui dati generati.



Obiettivi da rispettare:

- Frigate + MQTT integrati
- Listener eventi → Telegram
- 5 snapshot + clip video
- crop persona
- fasce orarie / giorni / festività
- retry automatico

- fallback su file log
- backup automatico
- secrets in .env
- avvio tramite systemd

STRUTTURA DEL PROGETTO

```
frigate-project/
├── docker-compose.yml
├── .env
├── frigate/
│   └── config.yml
├── mqtt/
│   └── mosquitto.conf
├── scripts/
│   ├── frigate_telegram_guard.sh
│   ├── frigate_mqtt_listener.sh
│   ├── retry_failed.sh
│   └── backup_frigate.sh
├── holidays/
│   └── holidays.txt
├── data/
│   ├── events/
│   ├── sent_events.db
│   └── telegram_fail.log
└── systemd/
    └── frigate-mqtt-listener.service
```

Analisi video e architettura di base

La base del sistema è Frigate, un software open source specializzato nella video-analisi tramite reti neurali. Frigate riceve lo stream RTSP proveniente da una telecamera IP (nel caso specifico una Tapo C200) e analizza i fotogrammi per individuare oggetti di interesse, in particolare le persone. La scelta di Frigate è motivata dalla sua maturità, dalla buona qualità dell'object detection e dalla capacità di funzionare anche in assenza di acceleratori hardware dedicati, seppur con prestazioni limitate. Su Raspberry Pi 4, Frigate è configurato per operare a bassa risoluzione e con un numero ridotto di frame al secondo, in modo da mantenere il carico della CPU entro limiti accettabili. Questa scelta comporta un compromesso tra reattività e consumo di risorse, ma risulta adeguata per scenari di sorveglianza non critici in tempo reale.

Gestione eventi, logica applicativa e notifiche

Per la gestione degli eventi generati da Frigate si è scelto di utilizzare MQTT come meccanismo di comunicazione asincrona. Quando Frigate conclude un evento di rilevazione di una persona, pubblica un messaggio su un topic MQTT. Questo approccio consente di disaccoppiare completamente la fase di analisi video dalla logica di notifica, rendendo il sistema più modulare e facilmente estendibile. La logica applicativa vera e propria è implementata tramite script Bash, una scelta volutamente minimalista finalizzata a ridurre le dipendenze software e a facilitare il debugging diretto sul sistema. Gli script ricevono gli eventi MQTT, applicano regole temporali e contestuali, recuperano snapshot e clip video da Frigate e inviano le notifiche all'utente tramite Telegram. Il sistema applica filtri

temporali stringenti, limitando il monitoraggio alle ore notturne e ai giorni sensibili come weekend e festività, al fine di ridurre notifiche inutili e migliorare la qualità degli avvisi.

Affidabilità, sicurezza, limiti ed evoluzione

Quando un evento supera i filtri configurati, il sistema invia una notifica Telegram composta da cinque snapshot consecutivi, automaticamente ritagliati sull'area della persona rilevata, seguiti da un breve clip video. In caso di fallimento dell'invio, l'evento viene registrato localmente e riprocessato automaticamente, garantendo che nessuna informazione venga persa. Tutti i materiali multimediali vengono inoltre archiviati su storage locale.
La gestione dei segreti è demandata a un file .env separato dal codice, mentre l'avvio e la persistenza dei servizi sono affidati a Docker Compose e systemd, assicurando affidabilità all'accensione del sistema. È inoltre presente un meccanismo di backup automatico notturno. Sebbene l'architettura presenti limiti intrinseci, come l'assenza di acceleratori hardware e l'uso di un singolo nodo, il progetto è stato concepito per evolvere facilmente verso soluzioni più performanti e robuste, mantenendo semplicità, controllo e assenza di lock-in tecnologico.