VIPRA

V.I.P.R.A.

Veicoli Interconnessi per la Prevenzione dei Rischi su Asfalto

Project details and budget

Presented to

Project-Day Archimede

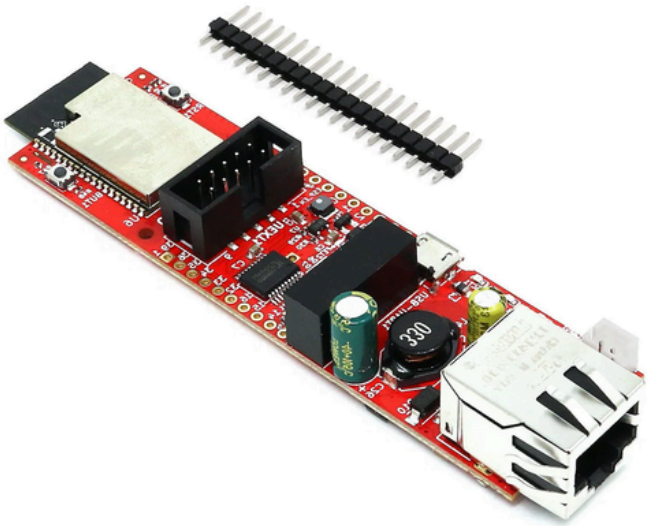
Presented by

Biesto Edoardo, Dilecce Riccardo, Ido Natan, Jendoubi Francesco, Navoni Federico e Patti Alessandro

ABOUT VIPRA

Il progetto propone una soluzione tecnologica avanzata per la sicurezza stradale, basata su un sistema hardware integrato che unisce elaborazione edge, connettività a lungo raggio e intelligenza artificiale. Garantisce rapidità di risposta, affidabilità, scalabilità ed efficienza energetica, mantenendo bassi i costi grazie all'uso di componenti open-source e protocolli ottimizzati

Il gateway ESP32-POE è il componente centrale del sistema, fungendo da interfaccia intelligente tra la rete MQTT e i dispositivi LoRa. Collegato via Ethernet alla Raspberry Pi, gestisce efficientemente il flusso bidirezionale dei dati, evitando i limiti delle connessioni wireless. L'alimentazione PoE semplifica il cablaggio e l'architettura a doppio core consente la gestione parallela dei messaggi.



ABOUT VIPRA

La comunicazione a lungo raggio è gestita da due nodi LoRa. Il modulo IN riceve i dati da veicoli e sensori, applica un filtro per eliminare duplicati e invia solo informazioni valide al gateway. Il modulo OUT assegna un ID univoco ai messaggi e ne gestisce la priorità, garantendo l'invio immediato degli alert più critici. Il sistema copre fino a 10 km, consuma poca energia ed è protetto da crittografia AES-128.

La Raspberry Pi è il nucleo analitico del sistema, dotata di capacità di visione artificiale ed elaborazione edge per analizzare in tempo reale i flussi video. Rileva ostacoli e altri rischi grazie a modelli di intelligenza artificiale ottimizzati, che assicurano rapidità e precisione. Gli alert generati vengono trasmessi via MQTT con metadati contestuali, rendendo le segnalazioni complete e strutturate. Il sistema supporta aggiornamenti over-the-air, sia per il firmware che per i modelli AI, garantendo adattabilità e manutenzione semplificata.

Contesto applicativo e funzioni primarie

Contesto Applicativo

Il sistema VIPRA è progettato per aumentare la sicurezza stradale attraverso un modulo di rilevamento e classificazione visiva installato su veicoli privati. Utilizzando una telecamera intelligente e algoritmi di intelligenza artificiale, il sistema identifica potenziali pericoli (es. ostacoli, pedoni) e li comunica al conducente o ad altri veicoli tramite un protocollo broadcast dinamico. L'obiettivo è mitigare rischi in scenari di guida quotidiana, con un focus specifico su strade asfaltate.

Obbiettivi del sistema e funzioni primarie

VIPRA è un sistema avanzato per la sicurezza stradale che combina rilevamento proattivo e allerta in tempo reale. Utilizzando una telecamera IMX500 con capacità di object detection integrata, il sistema analizza continuamente il flusso video, identificando potenziali situazioni di pericolo. I frame rilevanti vengono processati da modelli linguistici leggeri (Moondream2b e Gemma3: 1b) per una valutazione contestuale accurata, con solo i casi confermati trasmessi via protocollo MQTT.

Conclusioni

VIPRA rappresenta una soluzione innovativa per la sicurezza stradale, combinando hardware accessibile e software modulare. La scalabilità open-source ne permette l'adattamento a diversi scenari, pur richiedendo ottimizzazioni per garantire prestazioni in tempo reale.

ARCHITETTURA SOFTWARE PARTE AI

Panoramica

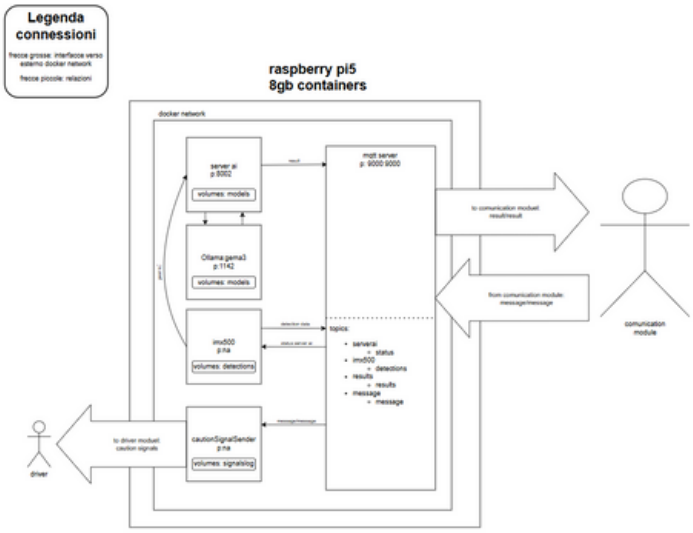
Il sistema è progettato come un insieme di container Docker orchestrati, che comunicano via MQTT. L'architettura è modulare per garantire:

- Isolamento: Ogni servizio gira in un container separato.
- Scalabilità: Possibilità di aggiornare singoli componenti.
- Efficienza: Uso ottimizzato delle risorse hardware .

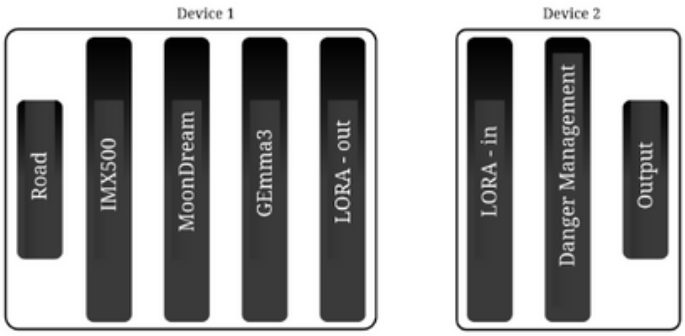
Gestione Dati

La gestione dati bilancia persistenza locale (per debug e modelli) e comunicazione real-time (per l'analisi immediata). L'uso di volumi Docker e MQTT ottimizza le risorse limitate del Raspberry Pi, mentre i log strutturati semplificano il debugging.

ARCHITETTURA RELAZIONE DOCKER:



ESEMPIO PIPELINE DI PROCESSO:



Componenti Principali

- Container imx500-camera:
  - Acquisisce e pre-processa il flusso video dalla telecamera IMX500.
- Container server_ai:
  - Analisi avanzata dei frame con pericolo sospetto
- Container Ollama:gemma3
  - Fornisce capacità di ragionamento semantico leggero (1B parametri).
- Broker MQTT (eclipse-mosquitto)
  - Cuore della comunicazione tra container e con l'esterno (porta 9000).

Spiegazione codice scheda OUT

Il sistema realizzato implementa una trasmissione dati affidabile su tecnologia LoRa, basata su microcontrollore ESP32 e sistema operativo real-time FreeRTOS. L'architettura è stata progettata per essere modulare, efficiente e facilmente estendibile. I principali aspetti implementativi sono i seguenti:

- **Acquisizione dati seriali**

Il dispositivo riceve dati da un'interfaccia esterna attraverso la porta seriale Serial1. Questo consente la connessione con moduli o sensori che comunicano tramite protocollo UART.

- **Gestione concorrente tramite coda RTOS**

I dati ricevuti vengono gestiti tramite una coda (queue) di FreeRTOS. Questo meccanismo assicura la sincronizzazione tra i task e impedisce la perdita di messaggi anche in caso di ricezione ad alta frequenza.

- **Trasmissione dati via LoRa**

I messaggi, una volta prelevati dalla coda, vengono trasmessi tramite un modulo LoRa, configurato con parametri personalizzati (es. frequenza, potenza di trasmissione, larghezza di banda). Ciò garantisce una comunicazione wireless a lunga distanza e basso consumo.

- **Monitoraggio risorse di sistema**

Il sistema include un controllo periodico dell'utilizzo della memoria heap, allo scopo di rilevare in anticipo eventuali condizioni critiche che potrebbero portare a instabilità o crash del dispositivo.

- **Architettura modulare e scalabile**

Il codice è organizzato in maniera modulare, con task indipendenti e funzioni ben separate, facilitando l'estensione del sistema. In particolare, è predisposto per l'integrazione di nuove funzionalità, come la ricezione di pacchetti LoRa, la registrazione dati su memoria SD o la gestione remota tramite rete.

Spiegazione codice scheda IN

Obbiettivi e funzionalità principali

Il codice sviluppato rappresenta una soluzione solida e scalabile per applicazioni in ambito IoT che richiedono comunicazione wireless a lunga distanza, tramite tecnologia LoRa, e la capacità di interfacciarsi con dispositivi esterni attraverso porta seriale. Il progetto è realizzato su microcontrollore ESP32, sfruttando il sistema operativo FreeRTOS per una gestione concorrente, efficiente e reattiva delle attività.
L'architettura del sistema si basa su una suddivisione in task indipendenti, ognuno responsabile di una funzionalità specifica, come la lettura di dati dalla porta seriale, la gestione dei messaggi tramite code di sistema, e la trasmissione tramite modulo LoRa. Questa organizzazione consente una esecuzione parallela e non bloccante delle operazioni, aumentando l'affidabilità del sistema anche in presenza di flussi di dati discontinui o irregolari.
L'impiego di FreeRTOS garantisce un controllo preciso sulla pianificazione dei task, l'utilizzo della memoria, e la sincronizzazione tra le varie componenti del firmware.

Spiegazione codice scheda POE

Obbiettivi e funzionalità principali

Il codice sviluppato implementa un sistema embedded che dimostra l'integrazione efficace tra comunicazione seriale e comunicazione di rete tramite un microcontrollore, con particolare riferimento alla piattaforma ESP32. Grazie all'utilizzo del sistema operativo FreeRTOS, il progetto è strutturato in modo modulare e asincrono, consentendo una gestione ottimale delle risorse hardware e un'elevata affidabilità operativa anche in contesti ad alta intensità di dati.
Il sistema è progettato per ricevere dati da dispositivi legacy tramite porta seriale, elaborarli e inoltrarli verso una rete locale o remota tramite protocolli come MQTT o LoRa, in funzione della configurazione adottata. Tutte le operazioni sono suddivise in task indipendenti, coordinati tramite code di messaggi, per assicurare la separazione logica delle funzioni, la reattività del sistema e l'isolamento degli errori.