Relazione Progetto ISS – Seconda Parte – Guiducci Daniele

GitHub: https://github.com/DanieleG27/ISS_2025

Obiettivi del progetto

Durante la seconda parte del corso, l'obiettivo principale è stato progettare e realizzare un sistema distribuito sfruttando l'architettura a microservizi e il paradigma ad attori. Il progetto si è sviluppato a partire dal classico "Game of Life" di Conway, con l'obiettivo di trasformarlo in un'applicazione distribuita, dove ogni cella rappresenta un attore indipendente, idealmente eseguito su un nodo fisico separato (es. Raspberry Pi).

L'intento non era solo far comunicare celle diverse su una griglia, ma anche esplorare concetti chiave del mondo IoT, tra cui l'interazione con l'ambiente fisico, l'uso di sensori e attuatori, e la gestione coordinata del comportamento distribuito.

Sfide dei sistemi distribuiti

Realizzare un sistema distribuito comporta vantaggi significativi, come la scalabilità e la tolleranza ai guasti, ma introduce anche una complessità maggiore nella gestione delle comunicazioni e del coordinamento tra nodi.

Nel caso specifico del progetto Conway distribuito, sono emersi alcuni problemi critici:

- Come assegnare dinamicamente la posizione di ogni cella nella griglia?
- Come determinare in modo efficace i vicini di ciascuna cella?
- Come sincronizzare il ciclo di aggiornamento dello stato tra celle diverse?

Per affrontare queste problematiche, si è adottato un modello **orchestrato**, dove un'entità centrale gestisce la disposizione delle celle, la comunicazione iniziale e la sincronizzazione dei cicli di vita. Ogni cella, pur restando indipendente, riceve informazioni fondamentali dall'orchestratore per funzionare correttamente. Per le comunicazioni, è stato utilizzato **MQTT**, un protocollo leggero ed efficace nei contesti IoT, che consente ai nodi di interagire tramite topic specifici senza conoscere l'identità fisica degli interlocutori.

Analisi e modellazione con QAK

Uno degli strumenti principali utilizzati per l'analisi e la modellazione del sistema è stato il linguaggio **QAK**, un DSL (Domain Specific Language) progettato specificamente per la programmazione ad attori.

QAK consente di descrivere i componenti del sistema come automi a stati finiti, facilitando così la definizione del comportamento reattivo di ogni attore. Il linguaggio mette in evidenza concetti chiave come il **contesto**, gli **attori**, i tipi di **messaggi** e la semantica delle interazioni. Questo approccio ha permesso di creare prototipi eseguibili rapidi, fornendo una chiara mappa delle interazioni tra i vari componenti del sistema.

La differenza fondamentale tra un DSL come QAK e una semplice libreria risiede nella **sintassi dedicata**, che astrae ulteriormente la complessità e rende il modello più leggibile e manutenibile.

Sperimentazione con Raspberry Pi e sensori

Un elemento chiave del progetto è stata l'integrazione del sistema con dispositivi fisici. Utilizzando **Raspberry Pi**, si è potuto testare in maniera concreta l'interazione tra software e hardware.

Due esperimenti principali sono stati affrontati:

1. **Stato visivo della cella** – Ogni Raspberry comandava un **LED**, acceso o spento in base allo stato della cella (viva o morta), trasformando così il comportamento astratto in una manifestazione fisica.
2. **Navigazione robotica e percezione ambientale** – Un altro modulo ha previsto l'uso di **sonar** per permettere a un robot di percepire la distanza da ostacoli, decidendo dinamicamente se avanzare, fermarsi o ruotare. Questo scenario ha messo in luce l'importanza dei **sensori (input)** e degli **attuatori (output)** in un sistema IoT distribuito.

Anche in questi contesti, l'approccio ad attori si è rivelato particolarmente utile, permettendo di modellare ogni componente fisico come un'entità autonoma e reattiva agli eventi esterni.

Conclusione

La seconda parte del progetto ISS ha permesso di esplorare a fondo la progettazione di sistemi distribuiti reali, dove la teoria degli attori e dei microservizi si fonde con la concretezza dell'hardware IoT. Il linguaggio QAK ha rappresentato uno strumento potente per modellare e testare rapidamente le architetture distribuite, mentre l'integrazione con Raspberry Pi ha evidenziato le potenzialità e le sfide del mondo embedded.

Il progetto ha mostrato come una buona astrazione software e un corretto disegno architetturale possano permettere a sistemi complessi di funzionare in modo coordinato, robusto e scalabile.