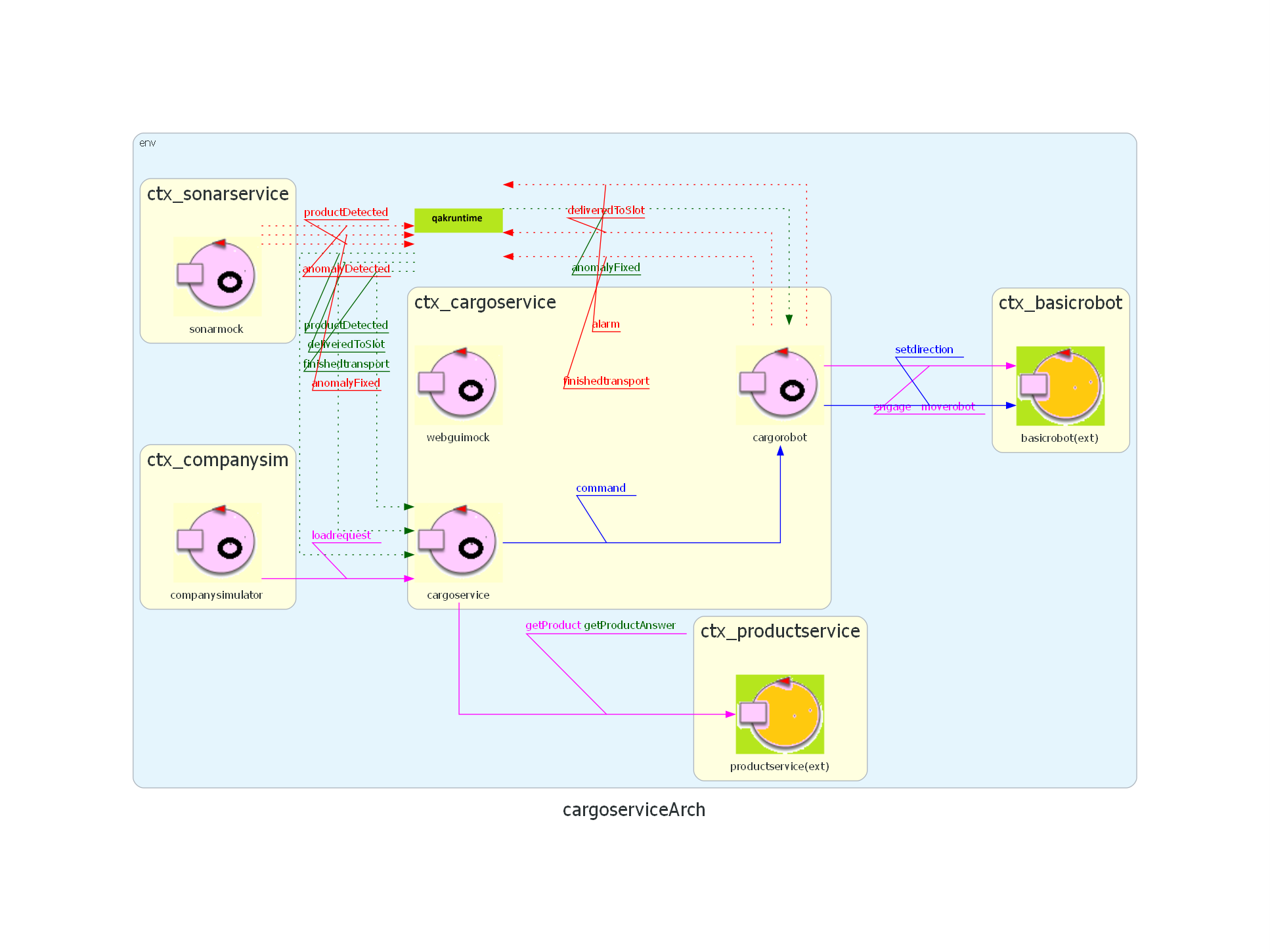
**SPRINT 2:**

**Punto di Partenza:**

Nello sprint1 il team ha implementato i componenti cargoservice e cargorobot, ovvero, il nucleo del sistema stesso, ottenendo la seguente architettura:



Si vuole fornire per comprensione un vocabolario riassuntivo di termini definiti nello sprint0 e nello sprint1. **Vocabolario:**

|  |  |
| --- | --- |
| Termine | Significato attribuito |
| Container | Contenitore in cui viene inserito il prodotto |
| loadrequest /richiesta di carico | Richiesta mandata dalla compagnia, specificando un PID |
| prodotto | Nel sistema è l’oggetto trasportato dal robot, la cui presenza può innescare diversi eventi |
| prodotto registrato | Prodotto conosciuto da ProductService a cui è associato un PID e un peso(Weight) |
| Microservizio | Componente software progettato per svolgere una specifica funzione del sistema. Ogni microservizio comunica con gli altri tramite messaggi, rendendo il sistema flessibile e scalabile. |
| GUI (Graphical User Interface) /WebGUI | Interfaccia grafica utente accessibile via web, che consente di visualizzare in tempo reale lo stato della stiva e interagire in modo intuitivo con il sistema. |
| Bounded Context | Il "bounded context" (**contesto limitato**) è un concetto fondamentale nel Domain-Driven Design (DDD) e si riferisce a un ambito applicativo ben definito e autonomo all'interno del quale vengono definite entità, regole e logiche di business in modo univoco e chiaro. All'interno di un bounded context, il significato di ogni entità o concetto è inequivocabile e specifico per quel contesto, evitando ambiguità e conflitti con altri contesti. |
| IOPort | Punto fisico (porta) attraverso il quale i contenitori dei prodotti entrano o escono dalla nave. È il punto in cui il sonar rileva la presenza di un prodotto. |
| Sonar | Sensore a ultrasuoni che misura la distanza tra sé e un oggetto. Nel nostro sistema serve per rilevare se un contenitore è presente all’IOPort. |
| DDRobot | è un robot che utilizza due motori indipendenti per muovere le ruote o i cingoli. È il supporto fisico che viene comandato da cargorobot. |
| PID (Product Identifier) | Numero intero univoco assegnato a ciascun prodotto registrato, usato per tracciarne l'identità all'interno del sistema. |
| Slot | Spazio fisico nella stiva della nave dove può essere posizionato un contenitore. Esistono 4 slot disponibili; uno è sempre occupato (slot5). |
| Cargorobot | Robot mobile autonomo (a guida differenziale) incaricato di trasportare i contenitori dall’IOPort fino allo slot assegnato e poi tornare alla posizione HOME. |
| Stiva | Area rettangolare della nave in cui i contenitori vengono caricati. Contiene gli slot e l’IOPort. |
| ProductService | Microservizio che gestisce la registrazione dei prodotti. Verifica i dati e assegna un PID univoco |
| CargoService | Microservizio che riceve richieste di carico, controlla i vincoli, assegna gli slot e coordina il caricamento tramite cargorobot. |
| SonarService | Microservizio che rileva la presenza di un contenitore all’IOPort tramite i dati forniti dal sonar. |
| DFREE | Distanza soglia usata dal sonar: se la distanza misurata è maggiore di DFREE per 3 secondi, si ipotizza un malfunzionamento del sensore. |
| MaxLoad | Peso massimo complessivo che la nave può sopportare. Il sistema rifiuta richieste che farebbero superare questo limite. |
| Worker | Persona che colloca fisicamente i contenitori sull’IOPort dopo che sono stati registrati. |
| Sistema logico di riferimento | Rappresentazione concettuale dell’intero sistema, con attori, componenti e interazioni, usata come base per l’architettura e la progettazione tecnica. |
| attore | Entità che svolge un ruolo attivo nel sistema, eseguendo azioni e comunicando con gli altri attori attraverso messaggi |
| Linguaggio QAK | Linguaggio modellistico usato per descrivere e simulare il comportamento dei componenti del sistema come “attori” |
| POJO | Plain Old Java Object: un oggetto di una classe in java |
| Anomalia | Nel documento è inteso come un comportamento inatteso di un componente hardware, tale da compromettere il normale funzionamento del sistema. |

**Goal dello Sprint2:**

* Enunciazione esplicita dei requisiti del sonarservice
* Analisi dei requisiti enunciati
* Definizione dell’architettura logica con modello eseguibile in qak e mockup dei servizi non ancora implementati (webgui)
* Progetto e realizzazione

**Enunciazione esplicita dei requisiti del sonarservice:**

Nel precedente sprint il componente sonarservice è stato sostituito da un mock che ne simulasse il comportamento. In questo ci proponiamo di progettarlo e implementarlo.

Il sonarservice coordina due dispositivi forniti dal committente, sonar e led, ed il suo scopo principale è quello di rilevare i prodotti posti all’IO-Port.

Esplicitiamo le funzionalità del sonarservice.

**RF1. Segnalazione rilevamento prodotto:** il sonarservice deve essere in grado di segnalare tempestivamente il rilevamento di un prodotto di fronte all’IO-Port. In questo caso verrà fatto lampeggiare il led fornito dal committente.

**RF2. Segnalazione rilevamento anomalia:** il sonarservice deve essere in grado di segnalare il rilevamento di un’anomalia nelle sue misurazioni nell’immediato, in modo che il sistema possa gestirla correttamente. Il sonarservice deve inoltre accendere il led fornito.

**RF2. Segnalazione risoluzione anomalia:** in seguito alla segnalazione del rilevamento di un’anomalia, il sonarservice deve segnalare al sistema la sua risoluzione. Il led viene spento.

**Analisi dei requisiti enunciati:**

Per poter essere ‘tempestivo’ nelle sue segnalazioni, il sonarservice deve adottare un comportamento reattivo e periodico, basato su misurazioni regolari effettuate tramite il sensore sonar.

**Comportamento del sonarservice:**

* Esegue misurazioni periodiche tramite il dispositivo device
* Valuta ogni misura e aggiorna uno stato interno tra i seguenti:
  + Detecting
  + ProductDetected
  + AnomalyDetected
  + WaitingForFix
* In base allo stato e al valore misurato, genera gli eventi richiesti e comanda il LED

**Misurazioni e soglie:**

La distanza misurata *d* può ricadere in tre intervalli:

|  |  |
| --- | --- |
| 0 < d < DFREE/2 | Se si mantiene entro questo intervallo per almeno tre secondi, si rileva la presenza di un container |
| DFREE/2 <= d <= DFREE | Non è presente alcun container |
| d > DFREE | anomalia |

**Logica del rilevamento:**

* Il rilevamento di un prodotto richiede che il valore d rientri stabilmente nel primo intervallo (0 < d < DFREE/2) per almeno 3 secondi consecutivi (ossia un numero N di misurazioni, dove N = 3s / periodo Campionamento).

Se, quindi, si registra la presenza di un container, il sonarservice emette l’evento **productDetected**

* Il rilevamento di un’anomalia deve essere istantaneo: appena una misura supera DFREE, viene generato un evento e acceso il LED.

In questo caso sonarservice emette l’evento **anomalyDetected**.

* La risoluzione dell’anomalia avviene quando viene registrato un valore d ≤ DFREE. In tal caso, il sonarservice:
  + emette un evento **anomalyFixed**
  + spegne il LED

In seguito alla registrazione di un’anomalia, il sonarservice deve attendere che il sonar registri un valore valido per poter mandare un segnale di risoluzione dell’anomalia.

**Definizione dell’architettura logica con modello eseguibile in qak:**

Il sonarservice è un microservizio indipendente che si occupa del rilevamento di oggetti tramite sonar e della gestione di anomalie.

Il sonarservice ha il compito di:

* misurare la distanza di oggetti tramite sensore sonar
* emettere eventi significativi:
  + productDetected
  + anomalyDetected
  + anomalyFixed
* segnalare lo stato del sistema tramite messaggi e controllo del LED.

**Suddivisione in componenti collaboranti:**

I compiti più importanti del sonar sono quello di effettuare la misurazione e quello di segnalare, se necessario, gli eventi descritti precedentemente.

Per migliorare la chiarezza e la manutenibilità, il sonarservice è stato suddiviso in due attori collaboranti.

**listener:** legge continuamente i dati dal sensore (tramite lo script Python sonar.py) e li trasmette al reactor. Gestisce anche un intervallo temporale fisso tra le letture (polling ogni secondo) per garantire un monitoraggio costante. Non interagisce direttamente con l’hardware, ma coordina la raccolta dei dati e l’emissione degli eventi (sonardata).

**Reactor:** riceve i dati dal listener e applica la logica di rilevamento degli eventi: rileva anomalie (anomalyDetected / anomalyFixed) e prodotti (productDetected). Interagisce con l’hardware, eseguendo gli script Python per accendere/spegnere il LED in base alle condizioni rilevate.

Questa suddivisione consente una chiara separazione tra la logica applicativa e il controllo hardware, facilitando l’estensione e la manutenzione del sistema.

**Comunicazione con l’hardware:**

La comunicazione avviene tramite l’esecuzione di script Python.

Gli script comandano i pin GPIO di un raspberry PI dotato di un sonar e di un led.

Si indicano di seguito i link ai singoli script con allegata una corta descrizione:

|  |  |
| --- | --- |
| getSonarDistance.py | Ottiene la distanza misurata dal sonar |
| ledCheck.py | Fa lampeggiare il led per controllare che funzioni |
| sonarStart.py | Accende il sonar |
| sonarStop.py | Spegne il sonar |
| turnLedOff.py | Spegne il led |
| turnLedOn.py | Accende il led |

**Progetto e realizzazione:**

**link al codice:**

**I dispositivi utilizzati:**