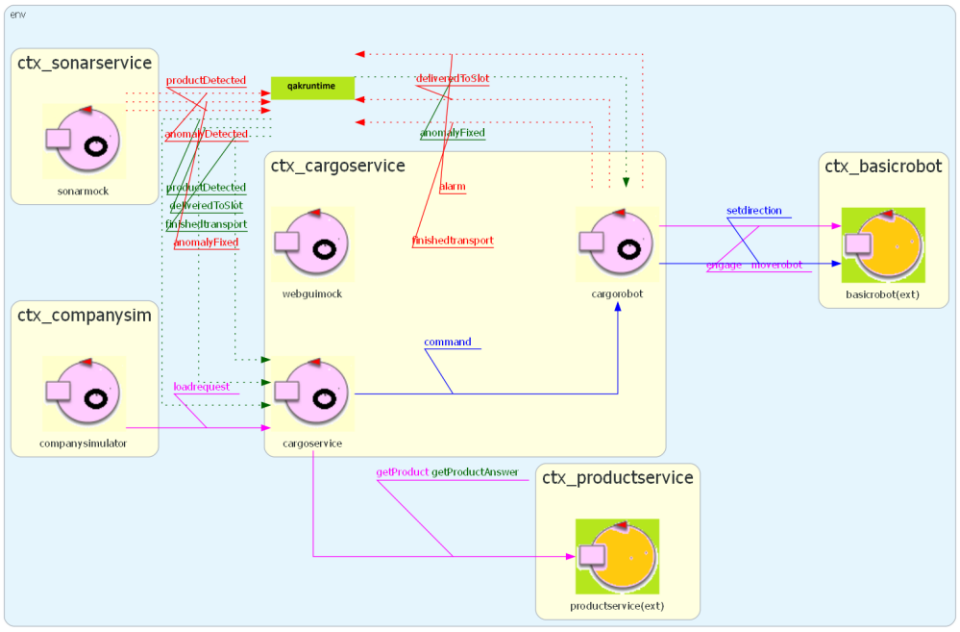
SPRINT 2:

Punto di Partenza:

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



cargoserviceArch

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, connessi però allo stesso dispositivo ***raspberry*** che li controlla, 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.

Problematiche affrontate:

L'hardware:

Il sistema è stato implementato su un **Raspberry Pi 4 Model B**, un microcomputer a basso costo e dalle dimensioni ridotte, molto utilizzato in ambito didattico e prototipale per progetti di elettronica e IoT.

Il Raspberry Pi 4B integra:

- una CPU quad-core ARM,
- porte USB, HDMI e GPIO (General Purpose Input/Output),
- supporto a Linux e a numerosi linguaggi di programmazione (Python, Java, C/C++).

In questo progetto, le porte GPIO vengono utilizzate per collegare e controllare i dispositivi esterni:

- **Sonar** (sensore a ultrasuoni) per la misurazione delle distanze, utile al rilevamento di prodotti e anomalie;
- **LED** che funge da dispositivo di segnalazione visiva, attivato o lampeggiante in base agli eventi generati dal SonarService.

Grazie a questa configurazione, il sonar e il LED costituiscono un **sistema unico e compatto**, gestito direttamente dal Raspberry Pi.

Come avviene la comunicazione con l'hardware:

La comunicazione avviene tramite l'esecuzione di script Python forniti dal committente.

Gli script comandano i pin GPIO di un raspberry PI dotato di un sonar e di un led, in questo modo può ottenere una misurazione al secondo e controllare l'accensione di un led.

Si indicano di seguito i link ai singoli script con allegata una descrizione del loro funzionamento:

sonar.py	Ottiene la distanza misurata dal sonar
ledPython25Blink.py	Fa lampeggiare il led per controllare che funzioni
ledPython25Off.py	Spegne il led
ledPython25On.py	Accende il led

Criterio di rilevamento prodotto nel sonarservice:

Il **SonarService** non si limita a segnalare immediatamente la presenza di un prodotto sulla base di una singola misurazione, perché ciò potrebbe introdurre falsi positivi dovuti a rumore o a rilevamenti sporadici.

Per garantire **affidabilità** è stato introdotto un **meccanismo di validazione su misurazioni consecutive**, applicato sia al rilevamento del prodotto che alle anomalie:

1. Ogni volta che viene acquisita una distanza, si verifica se la misurazione è **consistente** con la presenza di un prodotto (ovvero, se la distanza è inferiore a una soglia prestabilita, variazione improvvisa per un'anomalia).
2. Viene mantenuto un **contatore di misurazioni consistenti** (*productCounter*, *anomalyCounter*):
    - Se la misurazione corrente è consistente → il contatore viene incrementato.
    - Se la misurazione è non consistente → il contatore viene azzerato.
3. Quando il contatore raggiunge almeno **3 misurazioni consecutive** (effettuate a distanza di un secondo l'una dall'altra), il sistema considera il prodotto effettivamente rilevato.
4. Dopo la segnalazione, il contatore viene nuovamente riportato a zero per prepararsi a un nuovo ciclo di rilevamento.

In questo modo, il SonarService riduce drasticamente falsi positivi, dovuti a rilevamenti isolati o fluttuazioni momentanee del sensore.

Sebbene la richiesta del committente riguardasse solo il rilevamento dei prodotti, lo stesso approccio è stato esteso anche alla gestione delle anomalie, così da rendere il comportamento del sistema più uniforme e robusto.

Progetto e realizzazione:

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 tre attori collaboranti.

Sonar_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).

Sonar_Reactor: riceve i dati dal listener e applica la logica di rilevamento degli eventi: rileva anomalie (anomalyDetected / anomalyFixed) e prodotti (productDetected). Notifica tutto il sistema degli eventi rilevati e richiede al led_device di accendere o spegnere il led.

Led_device: innesca gli script necessari per accendere o spegnere il led quando richiesto dal sonar_reactor.

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

Messaggi:

il sonar_reactor emette i seguenti eventi, già definiti durante lo sprint1, che vengono rilevati dal cargorobot e dal led_device :

productDetected	Segnalazione del rilevamento di un prodotto
anomalyDetected	Segnalazione del rilevamento di un'anomalia
anomalyFixed	Segnalazione della riparazione di un'anomalia rilevata in precedenza

il sonar_listener manda sotto forma di evento su un canale locale la misura della distanza rilevata, che viene letto dal sonar_reactor:

distance(D)	Valore della distanza rilevata

Led_device:

Il led_device non emette alcun messaggio, ma rimane in attesa degli eventi generati dal sonar_reactor.

Il suo compito è puramente attuativo, ovvero, reagire agli eventi ricevuti pilotando il LED tramite script Python dedicati:

- **anomalyDetected**: accende il LED segnalando una condizione anomala.
- **anomalyFixed**: spegne il LED indicando che la condizione anomala è stata risolta
- **productDetected**: fa lampeggiare il LED per segnalare la rilevazione di un prodotto

In questo modo il LED funge da **interfaccia fisica di notifica**: non elabora logica, ma traduce lo stato del sistema in un segnale visivo chiaro e immediato.

Comunicazione:

QAK, come già anticipato, permette la comunicazione tramite **TCP**, **CoAP** e **MQTT**.
In questo caso è stato aggiunto un **broker MQTT** per la pubblicazione degli eventi (productDetected, anomalyDetected, anomalyFixed) verso eventuali attori o sistemi esterni interessati.

Motivazione:

- MQTT viene scelto perché permette una comunicazione **publisher/subscriber efficiente e asincrona**, adatta per notificare più sistemi in tempo reale senza richiedere una connessione diretta con ognuno.

Modello eseguibile ottenuto: sonarservice.qak

Piano di test: TestSonarService.java

È stato inoltre aggiunto un attore che emettesse distanze diverse per osservare il comportamento del sistema:

```
QActor sonarsimul context ctx_sonarservice{
	State s0 initial {
		println("$name starts") color cyan

	}Goto work

	State work{
		delay 1000 // attendo che sonarlistener entri in attesa

		// misurazioni non consistenti
		[# var M = 30 #]
		println("emitting 30 ...") color magenta
		emitlocalstream distance	: distance($M)
```

```
            delay 1000
           // codice simile ….
            [# M = 0 #]
            println("emitting 0 ...") color magenta
                  emitlocalstream distance     : distance($M)
        }
}
```

Modello ottenuto: