

# ColdStorageSprint0V2

## Introduction

### Goal dello sprint0

- individuare un architettura logica iniziale che definisca le macro-entità del sistema e le loro interazioni
- definire un piano di lavoro iniziale

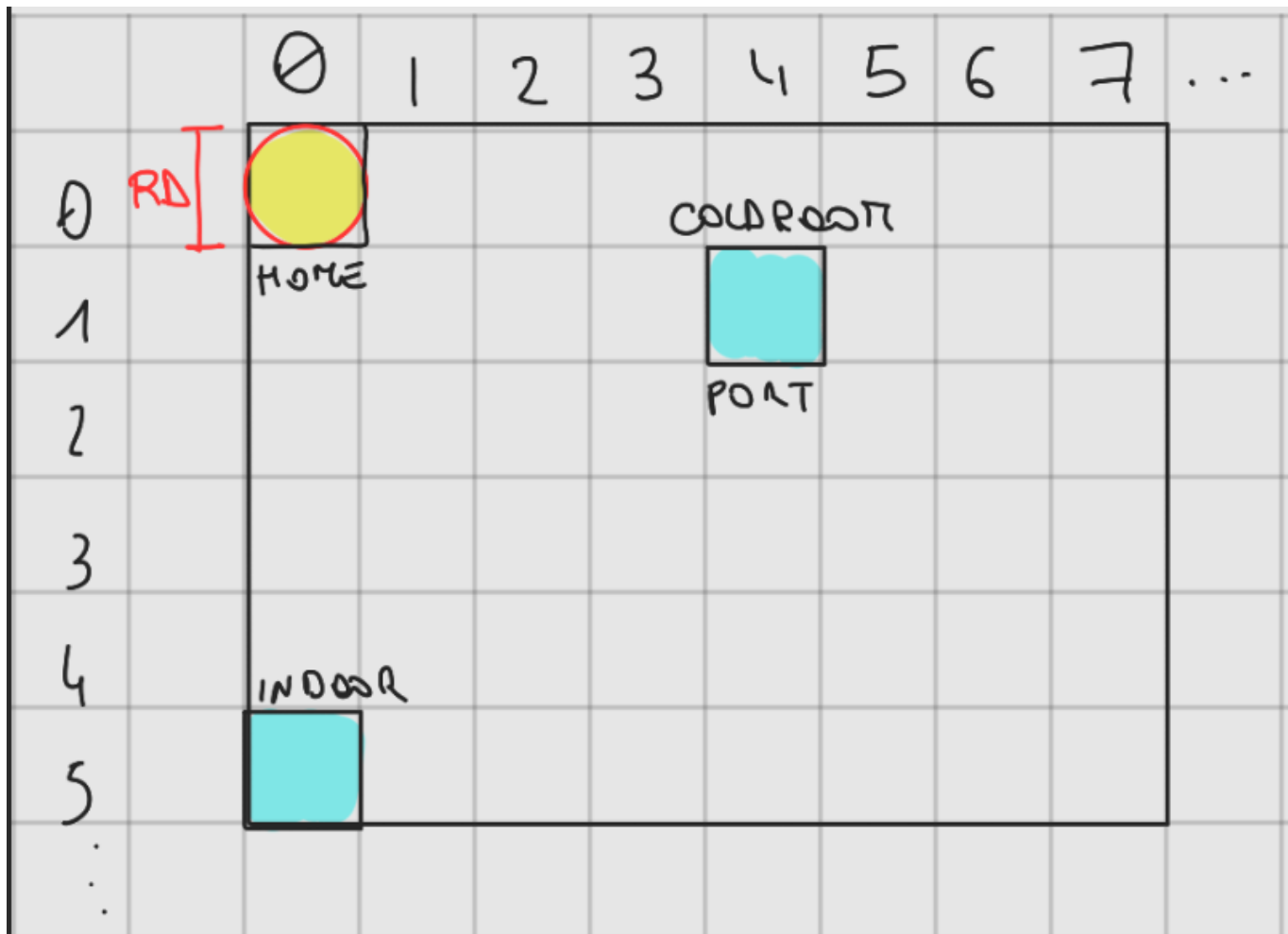
## Requirements

I requisiti sono scritti dal committente TemaFinale23.

### Requirements analysis

dai requisiti si evince che la stanza sia modellabile tramite una mappa che rappresenta una suddivisione in celle. La dimensione di ogni cella è legata alla dimensione del transport trolley. La mappa quindi è modellata come una griglia di quadrati di lato  $RD$ , dato che nei requisiti è specificato che "The transport trolley has the form of a square of side length  $RD$ ."

- Il robot viene considerato un oggetto inscrivibile in un cerchio di raggio  $RD$



- Il concetto di INDOOR e di PORT vengono modellate come posizioni nella mappa, ovvero come coppie di coordinate. In particolare INDOOR è formalizzata con la cella di coordinate la coordinata **(!!!!!!)**, mentre PORT è formalizzata con la cella di coordinate **(!!!!)**
- Il committente ha predisposto il software per modellare il ddr robot. Il ddr robot è modellato tramite l'entità robot astratta BasicRobot. Il ddr robot è modellato come un attore. I comandi che possono essere inviati al ddr robot sono:
  - **turnLeft** : il robot ruota a sinistra di 90°
  - **turnRight** : il robot ruota a destra di 90°
  - **moveForward** : il robot si muove in avanti di uno step
  - **moveBackward** : il robot si muove indietro di uno step
  - **alarm** : il robot si ferma
- Il transport trolley è quell'entità della applicazione che ha la responsabilità di controllare il basicrobot, il quale ci è stato già consegnato dal committente.
- Fridge Truck è un'entità che interagisce con il sistema ma non ne fa parte, dunque non verrà modellato.
- ServiceAccessGUI è un'entità formalizzata come un attore che ha la responsabilità di inviare messaggi alla ColdStorageService e di interagire con l'utente

- ColdStorageService è l'entità core business del sistema e siccome il sistema è distribuito, allora ColdStorageService è modellata come un attore
- Cold Room viene modellata come un attore di modo da interagire con ColdStorageService.
- KEYPOINT: essa potrebbe essere modellata come un POJO all'interno di ColdStorageService stesso, ma riteniamo che modellando Cold Room come un attore porti ai seguenti vantaggi:
  - si alleggerisce ColdStorageService di un altro compito, dato che ColdStorageService è l'entità core business
  - si segue il principio di singola responsabilità
  - se verranno previste in futuro una entità di "Scarico della Cold Room", allora questa entità potrà interagire direttamente con la Cold Room, senza dover interagire con ColdStorageService

Dai requisiti possiamo asserire che:

- il sistema è **distribuito** su più nodi di elaborazione;

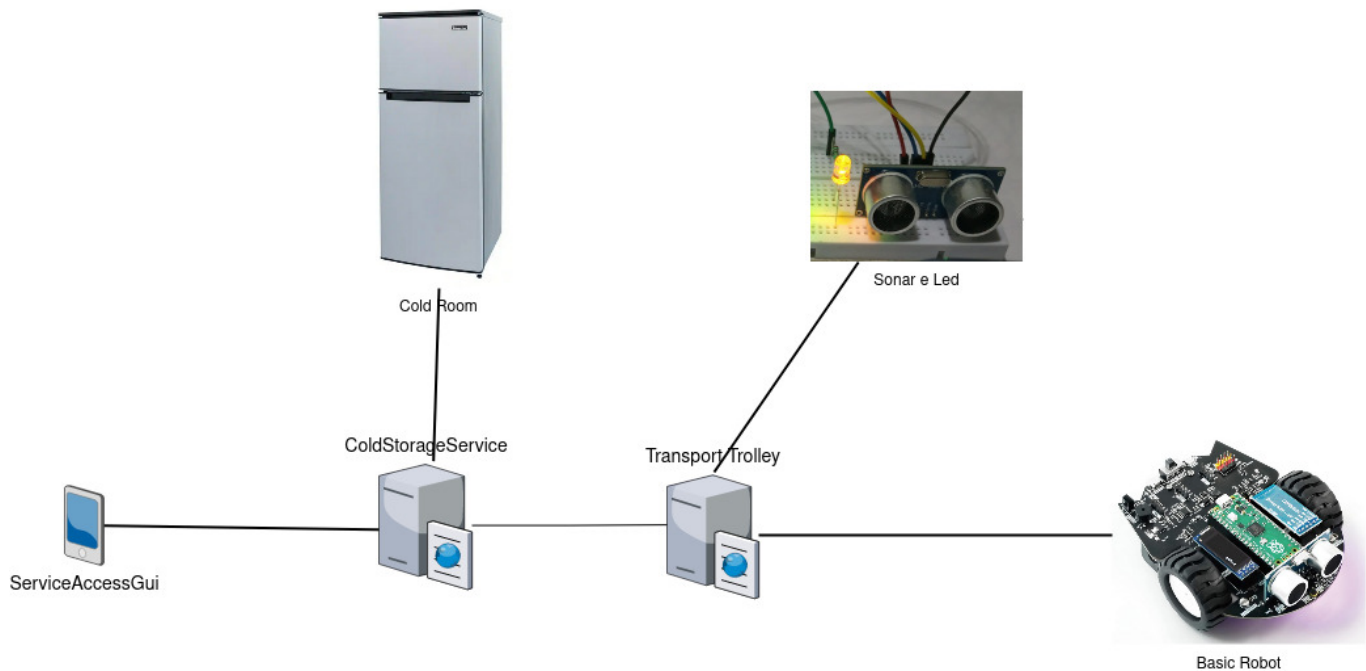
## Dialogo con il committente

Dopo opportuni colloqui con il committente, possiamo affermare che :

- le operazioni di carico e di scarico della ColdRoom potrebbero essere effettuate in parallelo oppure in maniera sequenziale. Per semplicità di realizzazione, dato che il committente non ha espresso riflessioni in materia, vengono effettuate in maniera sequenziale, ma nel caso realistico esse verrebbero fatte in parallelo.
- Non deve succedere che un camion, ricevuto il proprio ticket si veda rifiutata l'operazione di scarico una volta arrivato in INDOOR.
- E' possibile che il transport trolley non riesca a scaricare un intero truck tutto in un solo viaggio. Il committente ha affermato che non è una casistica da prendere in considerazione.
- la richiesta di un ticket può avvenire mentre sono ancora in corso operazioni di scarico precedenti

## Macrocomponenti

- Le macroentità sono:
  - *ColdStorageService*
  - *Transport Trolley*
  - *Sonar e Led*
  - *ServiceAccessGUI*
  - *ColdRoom*



## Architettura logica

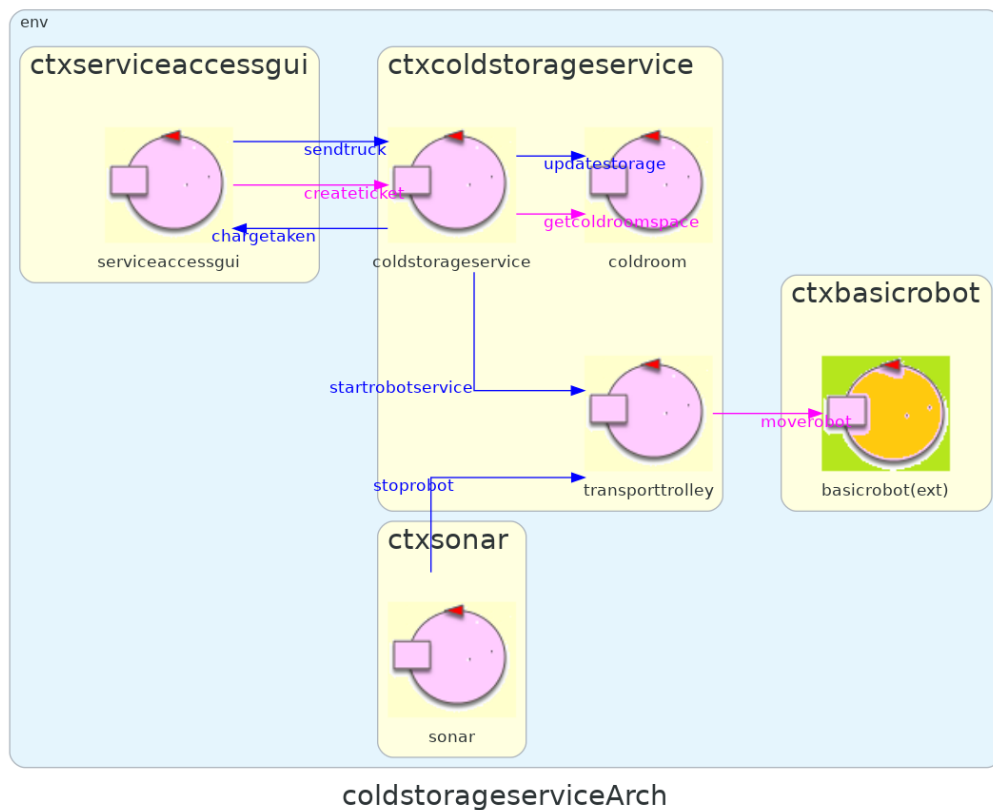
Il sistema è composto da:

- *ColdStorageService*: prende in carico richieste di generazione ticket e richieste di invio di un camion; si interfaccia con la ColdRoom per saperne lo stato; fa partire le deposit action; E' inserita in un suo contesto perchè è la core business.
- *Transport Trolley*: invia al Basic Robot la sequenza di comandi necessari per effettuare una deposit action
- *Basic Robot*: esegue i comandi del Transport Trolley. E' in un contesto esterno dato che è un servizio datoci dal committente.
- *Sonar*: effettua la misurazione e emette eventi. E' inserito in un contesto diverso perchè è specificato nei requisiti che è presente un Raspberry alla quale sarà collegato.
- *ServiceAccessGui*: si interfaccia con ColdStorageService per la richiesta di ticket. E' inserita in un suo contesto
- *Cold Room*: aggiorna lo stato della quantità di kg  
Come si legge nei requisiti, le interazioni tra le entità sono

Sender	Receiver	Description	Signature
ServiceAccessGui	ColdStorageService	send to the ColdStorageService a request to store new <b>FW</b> kg of food	createticket
ColdStorageService	ServiceAccessGui	If the request is accepted, the	acceptticket(TICKETNUMBER)

Sender	Receiver	Description	Signature
		services return a ticket [with the ticket number]	
ColdStorageService	ServiceAccessGui	Quando la Cold Room ha finito lo spazio disponibile, allora non può essere emesso il ticket	denyticket
ServiceAccessGui	ColdStorageService	the driver uses the ServiceAccessGUI to enter the ticket number and waits until the message <b>charge taken</b>	sendtruck
ColdStorageService	ServiceAccessGui	messaggio che permette al truck di presentarsi ad INDOOR	chargetaken
TransportTrolley	BasicRobot	messaggio di allarme del BasicRobot. Alla	alarm

Sender	Receiver	Description	Signature
		ricezione, il BasicRobot deve fermarsi	
Sonar	TransportTrolley	when it measures a distance less that a prefixed value <b>DLIMIT</b> , the transport trolley must be stopped	stoprobot



## Piano di lavoro

prevediamo di suddividere in sprint lo sviluppo del sistema secondo il seguente elenco

SPRINT	GOAL	TEMPO STIMATO	NOTE
SPRINT 1	a) prototipazione del core business ColdStorageService	2 giorni	si prevede di svolgere la fase a e b) in parallelo in quanto non fortemente dipendenti tra loro
	b) creazione di una infrastruttura containerizzata per facilitare il testing	3 ore	si prevede di svolgere la fase a e b) in parallelo in quanto non fortemente dipendenti tra loro
	c) testing	2 ore	
SPRINT 2	a) introduzione della la GUI nel sistema	1 giorno	
	b) test della gui	2 ore	
	c) test totale del sistema	2 ore	
SPRINT 3	a) introduzione del alarm requirement nel prototipo sviluppato allo sprint1	1 giorno	
	b) test alarm requirement	2 ore	
	c) test totale del sistema	2 ore	
SPRINT 4	a) deployment su robot fisico	3 ore	

Lo sprint 3 e lo sprint 4 possono essere realizzati in parallelo.

By Caterina Leonelli email:  
[caterina.leonelli2@studio.unibo.it](mailto:caterina.leonelli2@studio.unibo.it), GIT repo:  
<https://github.com/RootLeo00/sw-eng.git>



By Matteo Longhi email: [matteo.longhi5@studio.unibo.it](mailto:matteo.longhi5@studio.unibo.it)  
GIT repo:  
[https://github.com/carnivuth/iss\\_2023\\_matteo\\_longhi.git](https://github.com/carnivuth/iss_2023_matteo_longhi.git)

