

# ColdStorageSprint3V2

## Introduzione

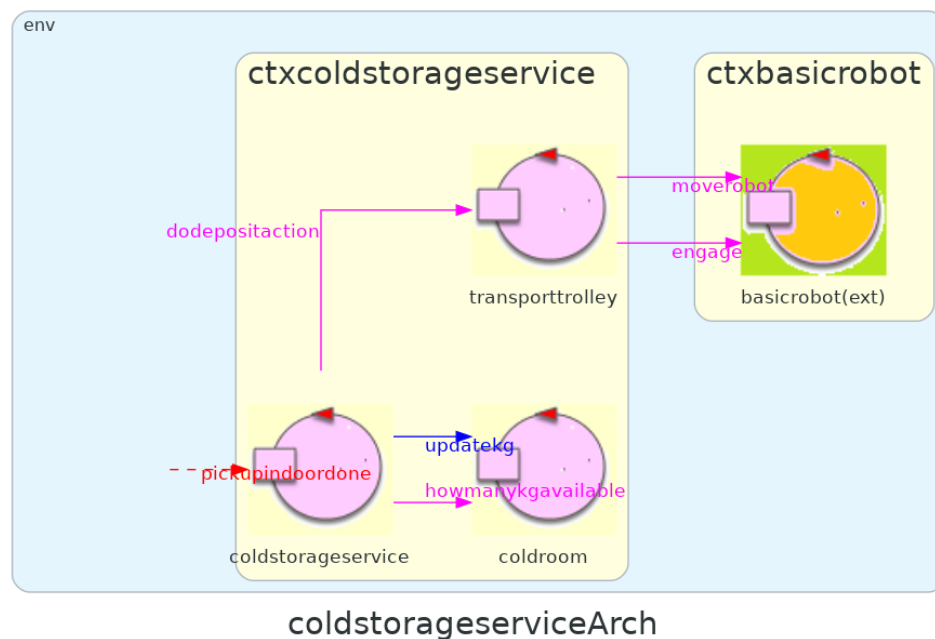
### Goal conseguiti nello sprint1

- prototipazione del core business ColdStorageService + TransportTrolley

### architettura logica sprint1

Codice dell'architettura:

<https://github.com/RootLeo00/robot-coldstorage/blob/sprint1/src/coldstorage-service-analisi.qak>



### Goal dello sprint3

- introduzione del alarm requirement nel prototipo sviluppato nella progettazione dello sprint1

## Requirements

The system includes a Sonar and a Led connected to a RaspberryPi.

The Sonar is used as an 'alarm device': when it measures a distance less than a prefixed value **DLIMIT**, the transport trolley must be stopped; it will be resumed when Sonar detects again a distance higher than **DLIMIT**.

The Led is used as a **warning devices**, according to the following scheme:

- the Led is **off** when the transport trolley is at HOME
- the Led **blinks** while the transport trolley is moving
- the Led is **on** when transport trolley is stopped.

### SERVICE USER STORY

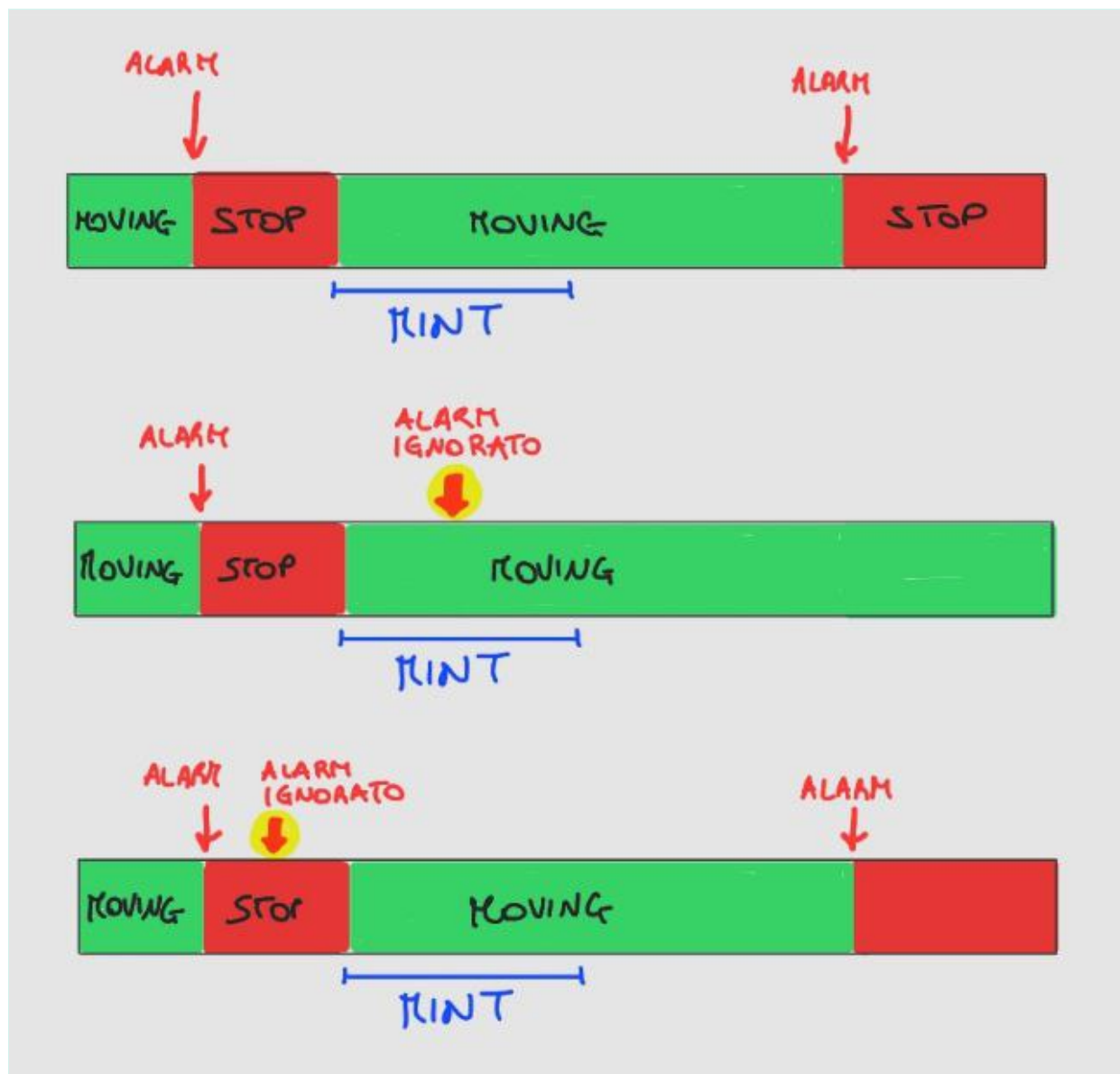
5) While the transport trolley is moving, the Alarm requirements should be satisfied. However, the transport trolley should not be stopped if some prefixed amount of time (**MINT** msec) is not passed from the previous stop.

## Requirements analysis

- Per Sonar si intende il dispositivo a ultrasuoni HCSR04 e viene formalizzato a livello logico con l'entità Sonar.
- Per Led si intende un dispositivo che emette energia sotto forma di luce e viene formalizzato a livello logico dall'entità Led.

Dopo opportuni colloqui con il committente, possiamo affermare che :

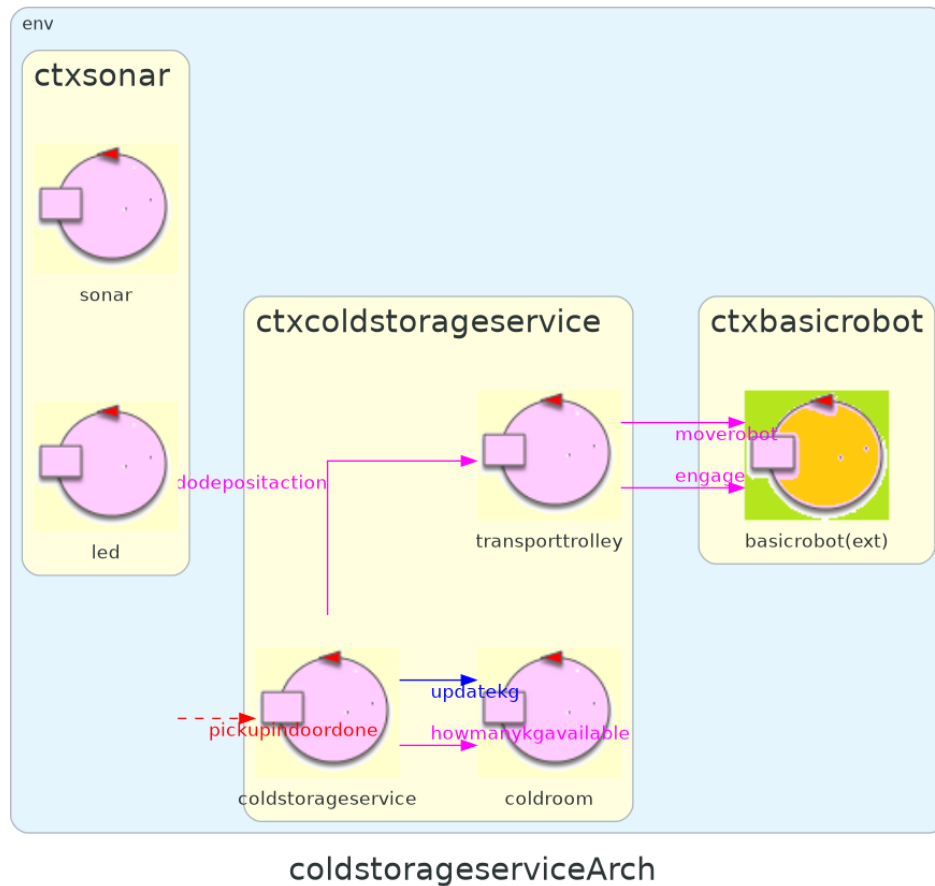
- il sistema **NON** è un sistema in tempo reale quindi non ci sono dei vincoli temporali. Questo vuol dire che a seguito dell'alarm requirement il robot deve fermarsi, ma è permesso che ci sia un ritardo di tempo.
- il Sonar e il Led sono su un raspberry che non dipende dalla service area.
- Il calcolo del delay di MINT parte da quando parte la endalarm dello stop precedente



## Architettura logica dopo analisi dei requisiti

Codice dell'architettura:

<https://github.com/RootLeo00/robot-coldstorage/blob/sprint3.5/src/coldstorage-service-analisi.qak>



## Problem Analysis

### Misurazioni corrette Sonar

Si predispongono due CodedQActor usabili per costruire una pipe che ha sonar come sorgente-dati e che provvede a eliminare dati spuri ([dataCleaner](#)) e a generare ([distancefilter](#)) eventi significativi per il livello applicativo.

### Alarm Basic Robot

Per fermare il robot si fa affidamento al comando "alarm", già formalizzato dai messaggi presenti in questa documentazione: [BasicRobot23.html](#)

## Il messaggio endalarm

Quando il Transport Trolley riceve un evento di alarm, il Basic Robot poi gli invia **moverobotfailed** come risposta finale dell'ultimo comando. Dunque si deve tener traccia dell'ultima mossa finale del Transport Trolley, dovendo così estendere il Transport Trolley sviluppato allo sprint1.

## Lo stato del Basic Robot

Il comportamento del led è profondamente legato allo stato del Basic Robot. In particolare, il led ha bisogno di sapere se il Basic Robot è in home, in movimento oppure fermo a causa di un alarm.

## Progettazione

### Misurazioni corrette Sonar

Si predispongono due CodedQActor usabili per costruire una pipe che ha sonar come sorgente-dati e che provvede a eliminare dati spuri ([dataCleaner](#)) e a generare ([distancefilter](#)) eventi significativi per il livello applicativo.

## Il messaggio stopobstacle

Se l'attore Sonar emettesse direttamente l'evento **alarm**, fermando così il basicrobot il Transport Trolley dovrebbe gestire i casi di fallimento dovuti alla risposta **moverobotfailed** data dal basic robot. Questo è il problema che si è riscontrato nello sprint3 originale.

In questo sprint3.4, si è deciso di inserire tutta la logica dell'alarm requirement all'interno dell'entità Sonar, ma non sarà questa entità a fermare direttamente il basicrobot, bensì sarà solo il Transport Trolley ad avere la responsabilità di fermare il basicrobot.

Si veda l'entità Sonar nella progettazione per maggiori dettagli, al seguente link:

<https://github.com/RootLeo00/robot-coldstorage/blob/sprint3.5/src/coldstorageservice.qak>

## Il timer

Dai requisiti si deve predisporre un timer che parte dall'ultima endalarm effettuata. Per fare ciò si adopera la classe Java System con il metodo:

```
System.currentTimeMillis()
```

## Controllo del Led tramite RobotState

Dai requisiti il Led deve modificare il proprio stato a seconda dello stato del robot (se in movimento oppure se è in home oppure se è in stato di allarme). Questo è possibile se il Transport Trolley

emette degli eventi sullo stato del robot ricevuti dal ControllerLed.

E' stato necessario inserire un evento nell'attore Transport Trolley della progettazione dello sprint1:

```
Event robotstate : robotstate(ATHOME,MOVING,STOPPED)
```

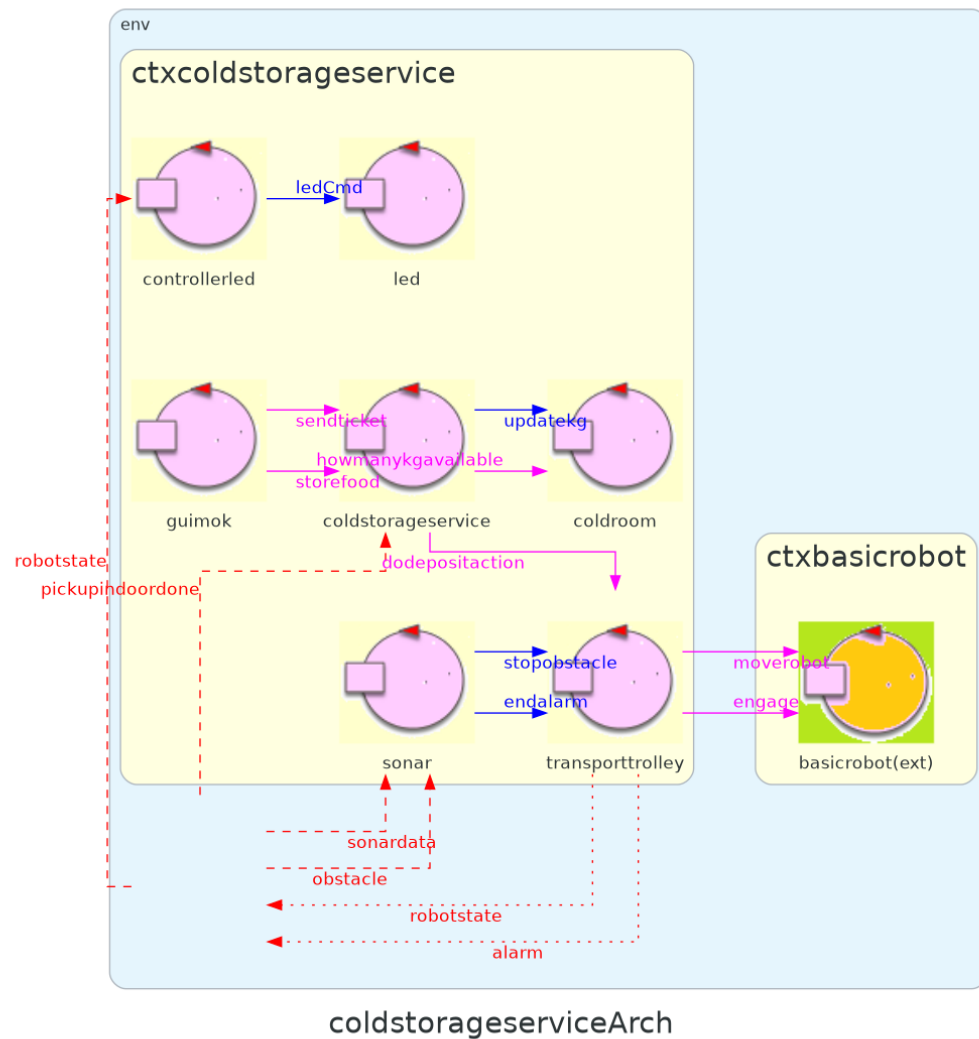
dove ATHOME, MOVING e STOPPED possono essere "true" o "false" a seconda dello stato del robot.

Il sistema è composto da:

- **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.
- **Led** : viene inserito nello stesso contesto di Sonar
- **ControllerLed** : entità che legge gli eventi del Transport Trolley per comandare il Led

Codice dell'architettura:

<https://github.com/RootLeo00/robot-coldstorage/blob/sprint3.5/src/coldstorageservice-progettazione.qak>



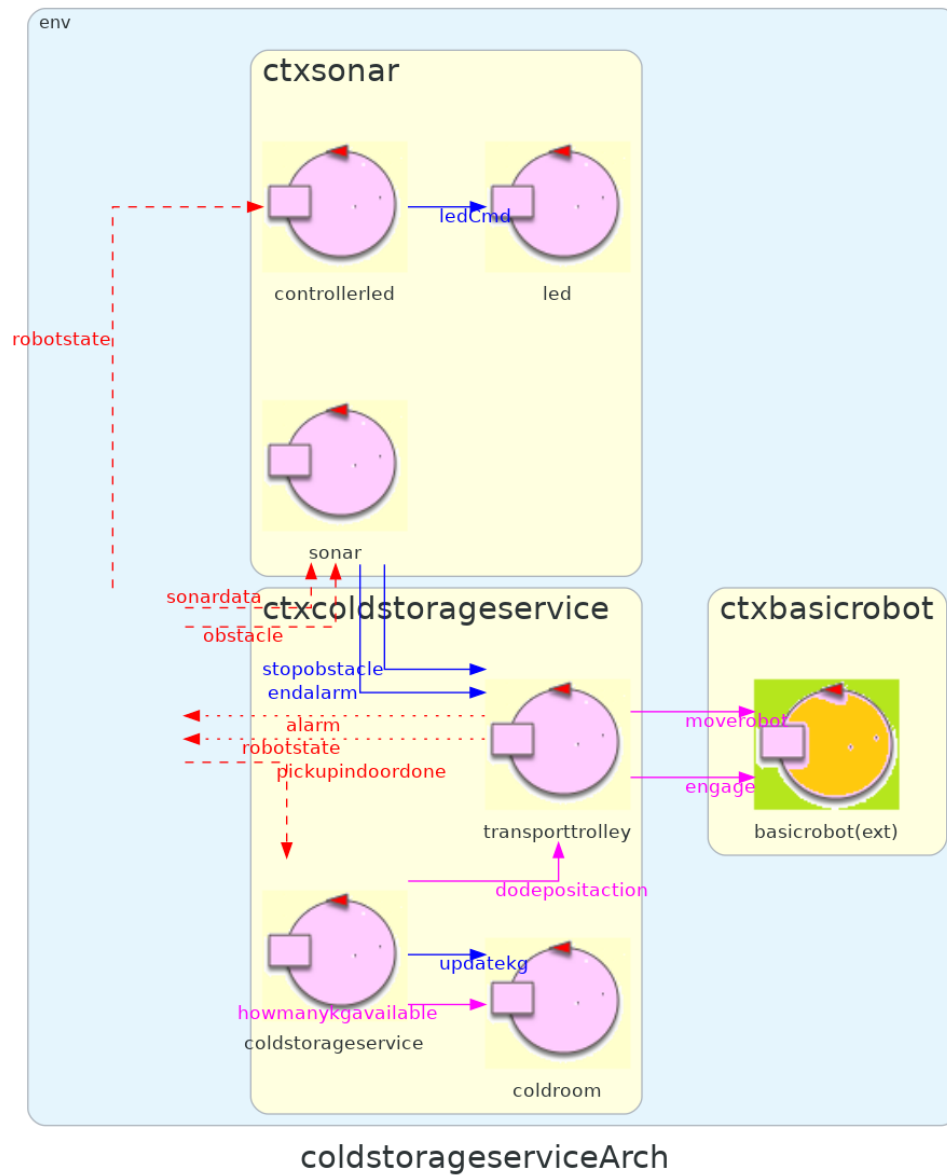
## Deployment

Come è descritto nei requisiti, Sonar e Led sono installati su un dispositivo (Raspberry) diverso da quello di Basic Robot o ColdStorageService.

L'architettura sottostante mostra che il Sonar e Led sono in un contesto esterno a quello della business logic.

Codice dell'architettura:

<https://github.com/RootLeo00/robot-coldstorage/blob/sprint3.5/src/coldstorageService-deployment.qak>



By Caterina Leonelli email:  
 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)

