Differential Drive Robot

Introduction

Realizzazione di un sistema software che comanda un Differential Drive Robot (DDR).

Requirements

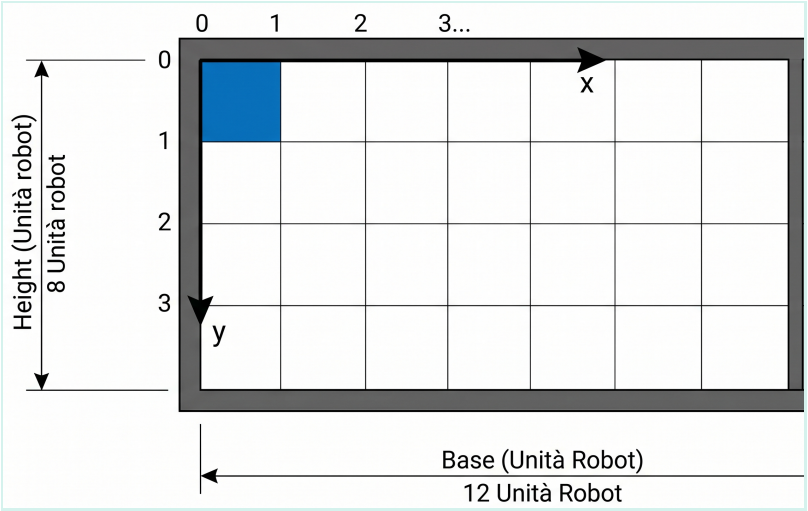
1. Costruire un sistema software che comanda un Differential Drive Robot (DDR) in modo che, partendo dalla posizione iniziale HOME, il robot si sposti lungo il perimetro di una stanza rettangolare vuota.
2. Il DDR si deve fermare quando tornato in HOME.
3. Il DDR è anche sensibile ai dati rilevati da un Radar.
4. Quando il Radar rileva un 'intruso' a una distanza minore di una distanza prefissata 'DMIN':
   - se il DDR si sta muovendo , il DDR si ferma fino a che il Radar non rileva più un 'intruso' così vicino
   - se il DDR è fermo in HOME, il DDR ruota su sè stesso fino a che un 'intruso' così vicino scompare

Il sistema integra un sonar in grado di misurare la distanza da eventuali ostacoli

Requirement analysis

Con ddr si intende un robot inscrivibile in un qadrato di lato R. La dimensione di un ddr viene definita come una unità robotica.
La stanza in cui si muove il ddr presenta le seguenti caratteristiche: Una altezza H e una base B, numero reale misurato in unità robotiche.
Il modello della stanza è pertanto formalizzato come una griglia:

```
val room = Array(H) { IntArray(B) }
```



Il ddr deve partire da una posizione iniziale HOME. Si definisce pertanto la HOME come l'area (0,0)-(1,1) in unità robot facendo riferimento al sistema stanza, ovvero la cella della griglia di indice (0,0).

- Il modello qak del sistema risponde al nome di ddrsystem.
  Il sistema deve comandare un DDR, pertanto deve essere un attuatore di comandi verso il ddr.
  Per modellare il sistema è necessario definire un QActor che controlla il ddr, che prende il nome di **mind**.
  Questo attore deve pertanto impostare il comportamento che specifica lo spostamento lungo il perimetro della stanza.
- Il ddr è un servizio che risponde ai comandi del sistema e pertanto al QActor mind.
  La distanza DMIN viene definita come una variabile reale, anch'essa in unità robotiche.

  Il sistema pertanto dovrà comunicare con il servizio ddr, sia per governare il movimento lungo il perimetro della stanza, oppure per gestire la risposta agli eventi scatenati dal radar.
  Il modello qak finora delineato è il seguente:

```
System ddrsystem
```

```
Context ctxddrsys ip [ host="localhost" port=8040 ]

QActor mind context ctxddrsys {

    [#
        var H //altezza della stanza in unità robotiche
        var B //base della stanza in unità robotiche
        val room = Array(H) { IntArray(B) }
        val DMIN  // distanza minima in unità robotiche
    #]

    State s0 initial{
        //gestione del movimento del ddr
    }

}
```

- Il Radar viene modellato come un attore, che ha il compito di rilevare gli oggetti nel suo campo d'azione, fornendo la distanza rilevata rispetto ad essi.
  Si definisce la distanza come una variabile reale D in metri.
  Il radar può essere rappresentato secondo 2 modelli di funzionamento:

A. Il radar emette un evento periodico, per avvertire della distanza attuale da ciò che rileva.

B. Il radar risponde ad una richiesta di distanza, fornendo la risposta attuale

Da cui il modello qak seguente:

```
//modello A: ad evento periodico
event radar : distanza (D) //distanza in metri

//modello B: a request/response
Request getsonardistance : getdistance(X) //richiesta distanza attuale
```

```
Reply   currentdistance : currentdistance(D)   //risposta radar, distanza


// QActor ddrsystem

QActor radar context ctxddrsys {

    [# var D #] //distanza rilevata in metri

    State s0 initial{

    }Goto handleDistance

    //gestione della comunicazione con qactor ddr, segnalazione distanza
    State handleDistance{
        emit radar : distance($D)
    }Transition t0
        whenRequest getsonardistance -> requestedDistance

    State requestedDistance{
        //elab request
        replyTo getsonardistance with currentdistance : currentdistance($D)
    }
}
```

Problem analysis

Test plans

Project

Testing

Deployment

Maintenance

By studentName email: gregorio.bussolari@studio.unibo.it,  GIT repo: https://github.com/GregorioBussolari/iss2026Unibo.git