

Final task ISS-2021 Bologna: Automated Car-Parking

Automated Car-Parking

A company intends to build an *automating parking service* composed of a set of elements:

- A software system, named **ParkManagerService**, that implements the required automation functions.
- A **DDR** robot working as a **transport trolley**, that is initially situated in its **home** location. The **transport trolley** has the form of a square of side length **RD**.
- A **parking-area** is an empty room that includes;
 - an **INDOOR** to enter the car in the area. Facing the **INDOOR**, there is a **INDOOR-area** equipped with a **weighsensor** that measures the **weight** of the car;
 - an **OUTDOOR** to exit from the **parking-area**. Just after the **OUTDOOR**, there is **OUTDOOR-area** equipped with a **outsonar**, used to detect the presence of a car. The **OUTDOOR-area**, once engaged by a car, should be freed within a prefixed interval of time **DTFREE**;
 - a number **N (N=6)** of **parking-slots**;
 - a **thermometer** that measures the temperature **TA** of the area;
 - a **fan** that should be activated when $TA > TMAX$, where **TMAX** is a prefixed value (e.g. 35)

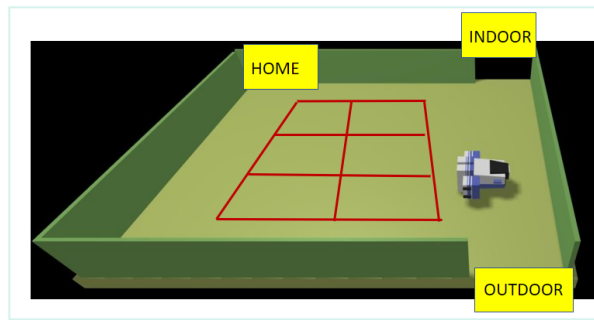
A **map** of the parking area, represented as a grid of squares of side length **RD**, is available in the file `parkingMap.txt`:

```
| r, 0, 0, 0, 0, 0, 0, X,
| 0, 0, X, X, 0, 0, 0, X,
| 0, 0, X, X, 0, 0, 0, X,
| 0, 0, X, X, 0, 0, 0, X,
| 0, 0, 0, 0, 0, 0, 0, X,
| X, X, X, X, X, X, X, X,
```

The map includes the positions of the **parking-slots** (marked above with the symbol **X**) and of the **fixed obstacles** in the area (the walls marked with the symbol **X**).

The area marked with **X** is a sort of 'equipped area' upon which the **transport trolley** cannot walk. Thus, to get the car in the **parking-slot (2,2)**, the **transport trolley** must go in cell **(1,2)**.

The proper scene for the WEnv is reported in: `parkingAreaConfig.js`



- a **parking-manager** (an human being) which supervises the state of the **parking-area** and handles critical situations.

The job of our company is to design, build and deploy the **ParkManagerService**.

User stories

As a **client - parking phase** :

- I intend to use a **ParkServiceGUI** provided by the **ParkManagerService** to notify my interest in *entering* my auto in the **parking-area** and to receive as answer the number **SLOTNUM** of a free parking-slot ($1 \leq \text{SLOTNUM} \leq 6$). **SLOTNUM == 0** means that no free slot is available.
- If **SLOTNUM > 0**, I move my car in front to the **INDOOR**, get out of the car and afterwards press a **CARENTER** button on the **ParkServiceGUI**. Afterwards, the **transport trolley** takes over my car and moves it from the **INDOOR** to the selected **parking-slot**. The **ParkServiceGUI** will show to me a receipt that includes a (unique) **TOKENID**, to be used in the *car pick up* phase.

As a **client - car pick up phase** :

- I intend to use the **ParkServiceGUI** to submit the request to pick up my car, by sending the **TOKENID** previously received.
- Afterwards, the **transport trolley** takes over my car and moves it from its **parking-slot** to the **OUTDOOR-area**.
- I move the car, so to free the **OUTDOOR-area**.

As a **parking-manager**:

- I intend to use the **ParkServiceStatusGUI** provided by the **ParkManagerService** to observe the **current state** of the **parking area**, including the value **TA** of the temperature, the state of the **fan** and the state of the **transport trolley** (**idle, working or stopped**).
- I intend to **stop** the **transport trolley** when **TA > TMAX**, activate the **fan** and wait until **TA < TMAX**. At this time, I stop the **fan** and resume the behavior of the **transport trolley**. Hopefully, the **start/stop of the fan** could also be automated by the **ParkManagerService**, while the **start/stop of the transport trolley** is always up to me.
- I expect that the **ParkManagerService** sends to me an **alarm** if it detects that the **OUTDOOR-area** has not been cleaned within the **DTFREE** interval of time.

Requirements

The **ParkManagerService** should create the **ParkServiceGUI** (for the client) and the **ParkServiceStatusGUI** (for the manager) and then perform the following tasks:

- **acceptIN**: accept the request of a client to park the car if there is at least one **parking-slot** available, select a free slot identified with a unique **SLOTNUM**.
A request of this type can be elaborated only when the **INDOOR-area** is **free**, and the **transport trolley** is at **home** or working (**not stopped** by the manager). If the **INDOOR-area** is already engaged by a car, the request is not immediately processed (the client could simply wait or could - optionally - receive a proper notice).
- **informIN**: inform the client about the value of the **SLOTNUM**.

If **SLOTNUM>0**:

1. **moveToIn**: move the **transport trolley** from its current location to the **INDOOR** ;
2. **receipt**: send to the client a receipt including the value of the **TOKENID** ;
3. **moveToSlotIn**: move the **transport trolley** from the **INDOOR** to the selected **parking-slot**;
4. **backToHome**: if no other request is present, move the **transport trolley** to its **home** location, else **acceptIN** or **acceptOUT**.

If **SLOTNUM==0**:

- **moveToHome**: if not already at home, move the **transport trolley** to its **home** location.

- **acceptOUT**: accept the request of a client to get out the car with **TOKENID**. A request of this type can be elaborated only when the **OUTDOOR-area** is **free** and the **transport trolley** is at **home** or working (**not stopped** by the manager). If the **OUTDOOR-area** is still engaged by a car, the request is not immediately processed (the client could simply wait or could - optionally - receive a proper notice).
 1. **findSlot**: deduce the number of the parking slot (**CARSLOTNUM**) from the **TOKENID**;
 2. **moveToSlotOut**: move the **transport trolley** from its current location to the **CARSLOTNUM/parking-slot** ;
 3. **moveToOut**: move the **transport trolley** to the **OUTDOOR** ;
 4. **moveToHome**: if no other request is present move the **transport trolley** to its **home** location;
else **acceptIN** or **acceptOUT**

- **monitor**: update the **ParkServiceStatusGUI** with the required information about the state of the system.

- **manage**: accept the request of the manager to stop/resume the behavior of the **transport trolley**.

About the devices

All the sensors (**weightsensor**, **outsonar**, **thermometer**) and the **fan** should be properly simulated by mock-objects or mock-actors.

When using a real robot

No further requirement.

When available a Raspberry and a sonar

The **outsonar** could be a real device. We can simulate the presence/absence of a car.

When using **only** the virtual robot or **no real sonar** available

Consider the new requirement:

- **authorize**: allow a manager to use the **ParkServiceStatusGUI** only if she/he owns **proper permissions**.

Non functional requirements

1. The ideal work team is composed of **3 persons**. Teams of 1 or 2 persons (**NOT** 4 or more) are also allowed.
2. The team must present a **workplan** as the result of the requirement/problem analysis, including some significant **TestPlan**.
3. The team must present the sequence of **SPRINT** performed, with appropriate motivations.
4. Each **SPRINT** must be associated with its own 'chronicle' (see [templateToFill.html](#)) that presents, in concise way, the key-points related to each phases of development. Hopefully, the team could also deploy the system using docker.
5. Each team must publish and maintain a GIT-repository (referred in the [templateToFill.html](#)) with the code and the related documents.
6. The team must present (in synthetic, schematic way) the **specific activity of each team-component**.

Guidance

- Oltre al codice sviluppato durante il corso, il progetto [it.unibo.qakDemo](#) include codice che potrebbe risultare utile per l'applicazione finale.
- Il numero e le finalità degli SPRINT sono definiti dal Team di sviluppo dopo opportune interazioni con il committente.
- Il committente (e/o il product-owner) è disponibile ONLINE in linea di massima ogni **Giovedì dalle 15 alle 18** fino a fine Luglio, ma è sempre contattabile on-demand via email.
- Lo svolgimento del lavoro è auspicabile avvenga in diverse fasi:

1. *Fase di analisi*, che termina con la definizione di una architettura logica del sistema, di modelli eseguibili e alcuni, significativi piani di testing.

E' raccomandato che i risultati di questa fase vengano presentati al committente (con opportuno appuntamento) prima della consegna finale del prodotto.

2. *Fase di progetto e realizzazione*, che termina con il deployment del prodotto.
3. *Fase di discussione* del lavoro svolto, che potrebbe (auspicabilmente) svolgersi IN PRESENZA in LAb2. E' opportuno che ogni partecipante sia pronto a discutere anche sugli elaborati che ha prodotto durante il corso.

By AN Unibo-DISI