UNIVERSITÀ DELLA CALABRIA

Dipartimento di Matematica e Informatica

Corso di Laurea in Informatica

A.A 2016/2017

RoboSpider

Progettazione ed implementazione di un robot Arduino e di un'Applicazione per telecomandarlo

Studenti

Domenico Violi 132165

Carlo Cristarella Orestano 170015

Docente

F.Ricca

Indice

Progetto RoboSpider

Tematica Progettuale

La progettazione in esame riguarda la realizzazione di un **RoboSpider** (robot) e di una **Mobile App** telecomando per guidarlo a distanza. Lo sviluppo per realizzare il robot sfrutterà la tecnologia Arduino.
Arduino è una piattaforma hardware dotata di microcontrollori e di schede elettroniche, viene utilizzata come strumento per la prototipazione rapida.

Raccolta dei requisiti Utente

In questa sezione vengono descritti gli aspetti che il progetto intende affrontare e quali caratteristiche esso dovrà avere. Il progetto **RoboSpider** garantirà all'**Utente** un robot quadrupede in Arduino, che ricordi la forma di un ragno, ed una **Mobile App** per il proprio dispositivo. Il suo utilizzo si presenta in due varianti, il movimento autonomo del **RoboSpider** o quello guidato tramite **Mobile App** dall'**Utente.**

RoboSpider

L'**Utente** accende il robot, avvia la **Mobile App** nel proprio dispositivo per poterlo telecomandare.

In seguito, l'**Utente** collega la **Mobile App** al **RoboSpider** tramite un collegamento Bluetooth, sceglie se pilotarlo manualmente o inserire il **RoboSpider** in modalità movimento autonomo.

Scegliendo la modalità manuale, l'**Utente** interagisce direttamente con il **RoboSpider**, mentre con la modalità di movimento autonomo il **RoboSpider** si muove da solo.

In tale modalità sarà in grado di:

- Evitare gli ostacoli rilevati tramite dei sensori.
- Non andare mai oltre la portata del Bluetooth.

L'**Utente**, quando lo desidera, può spegnere il **RoboSpider** effettuando le seguenti operazioni di spegnimento:

- Fermare il **RoboSpider** con l'apposito comando di stop tramite l'app.
- Scollegare la **Mobile App** dal **RoboSpider.**
- Prendere il **RoboSpider** e premere l'interruttore di spegnimento.

Casi d'uso – UML

Caso D'uso: Interagire con il **RoboSpider**.

Attore Primario: Utente.

L'**Utente** interagisce con il **RoboSpider** facendolo muovere nella direzione desiderata attraverso l'uso di un'app **con interfaccia Utente**.

Altri Attori: RoboSpider, **Mobile App, Sensore**.

Precondizioni: Il **RoboSpider** deve essere acceso. Inoltre, l'**Utente** deve avere la **Mobile App** installata su di un dispositivo per potersi connettere al **RoboSpider**.

Garanzia di successo (o postcondizione): Il **RoboSpider** ha eseguito dei movimenti nelle direzioni desiderate.

Scenario principale di successo:

1. L'**Utente** apre la **Mobile App** nel suo dispositivo.
2. L'**Utente** accoppia il **RoboSpider** alla **Mobile App**.
3. La **Mobile App** verifica che l'accoppiamento sia avvenuto con successo.
4. L'**Utente** seleziona la modalità di movimento manuale.
5. L'**Utente** tiene premuto la direzione dove far muovere il **RoboSpider**.
6. La **Mobile App** comunica al **RoboSpider** in che direzione muoversi.
7. Il **RoboSpider** esegue un movimento nelle direzioni desiderate dall'**Utente**.
8. L'**Utente** rilascia la direzione desiderata.
9. La **Mobile App** comunica di interrompere il movimento corrente.
10. Il **RoboSpider** termina l'esecuzione del movimento nella direzione desiderata dall'**Utente**.

I passi 5-10 vengono ripetuti finché l'**Utente** desidera.

11. L'**Utente** chiude la **Mobile App** nel suo dispositivo.
12. L'**Utente** spegne il **RoboSpider.**

Scenari Alternativi (o Estensioni):

3a. L'accoppiamento tra la **Mobile App** e il **RoboSpider** fallisce.

1. L'**Utente** verifica che la **Mobile App** riconosca le reti Bluetooth disponibili.
2. L'**Utente** riprova a eseguire nuovamente l'accoppiamento.

2.1 L'accoppiamento continua a fallire.

1. L'**Utente** installa nuovamente la **Mobile App**.
2. Il caso d'uso riprende al passo 2 dello scenario principale.

3. L'**Utente** apre la **Mobile App** nel suo dispositivo, in questo caso ritorna al passo 4 dello scenario principale.

4a. L'**Utente** vuole attivare la modalità di movimento autonomo da **Mobile App**.

1. L'**Utente** seleziona la modalità di movimento autonomo.
2. Il **RoboSpider** esegue movimenti in modo autonomo.
3. Il **RoboSpider** incontra un ostacolo.
4. Il **Sensore** comunica **RoboSpider** la presenza di l'ostacolo.
5. Il **RoboSpider** evita l'ostacolo.

   I passi 3-5 vengono eseguiti finché l'**Utente** desidera.
6. Il caso d'uso riprende al passo 10 dello scenario principale

4b. L'**Utente** vuole attivare la modalità di movimento autonomo da remoto

1. L'**Utente** sposta la levetta posta sul **RoboSpider** su ON.
2. Il **RoboSpider** esegue movimenti in modo autonomo.
3. Il **RoboSpider** incontra un ostacolo.
4. Il **Sensore** comunica **RoboSpider** la presenza di l'ostacolo.
5. Il **RoboSpider** evita l'ostacolo.

   I passi 3-5 vengono eseguiti finché l'**Utente** desidera.
6. L'**Utente** sposta la levetta posta sul **RoboSpider** su OFF
7. Il caso d'uso riprende al passo 11 dello scenario principale.

5a. La **Mobile App** termina in modo imprevisto.

1. Il **RoboSpider** perde la connessione con la **Mobile App** e si disaccoppia da questo.
2. Il **RoboSpider** interrompe eventualmente qualsiasi movimento.
3. Il caso d'uso riprende al passo 2 dello scenario principale.

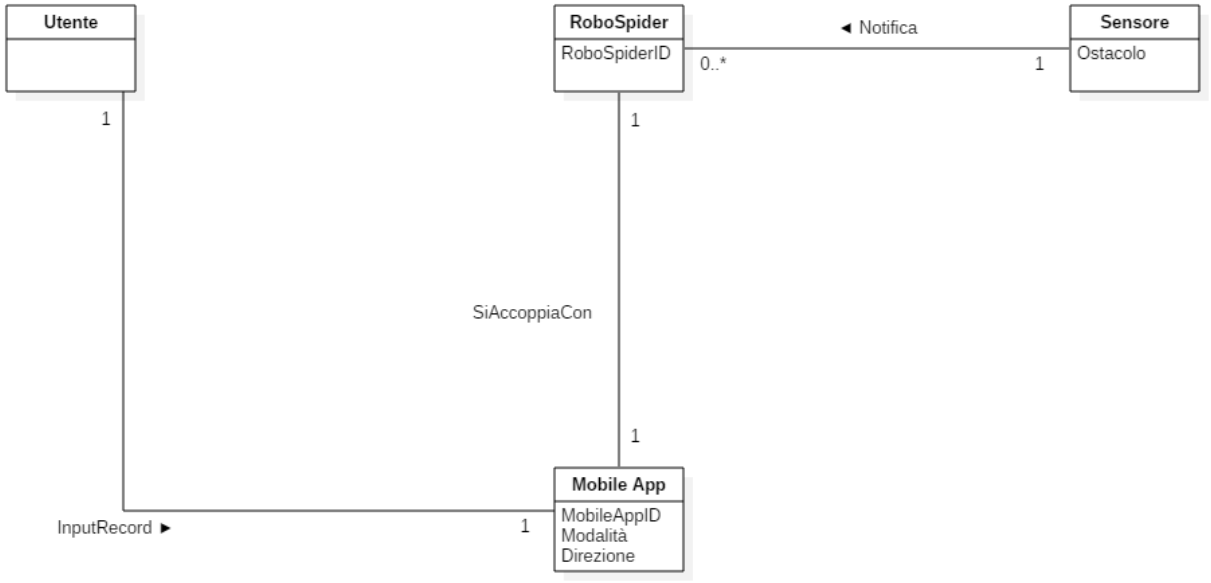
7a. Il **RoboSpider** si comporta in modo anomalo.

1. L'**Utente** preme il tasto STOP d'emergenza sulla **Mobile App**.
2. Il **RoboSpider** interrompe eventualmente qualsiasi movimento.
3. Il caso d'uso riprende al passo 2 dello scenario principale.

Requisiti speciali (o non funzionali):

-Interfaccia **Utente** di tipo touch screen. Il testo deve essere visibile ad una discreta distanza.
-Il **RoboSpider** dovrà muoversi avanti o indietro oppure ruotare su sé stesso.
-Si desidera un tasto STOP d'emergenza in caso di malfunzionamento.
-La **Mobile App** lavorerà su S.O. Android
-Si desidera che il tempo di reazione all'input dell'**Utente** sia inferiore a un secondo.

Modello di Dominio



Classi concettuali

In questa sezione viene riportata una descrizione delle classi concettuali che assumono un ruolo importante nel modello di dominio precedentemente illustrato.

- **Utente**

  Interagisce con la **Mobile App**, generando dei record di movimento da inviare al **RoboSpider**.

- **RoboSpider**

  Deve essere accoppiato esattamente con una **Mobile App** riconosciuta per consentire l'esecuzione dei movimenti.
  Riceve le notifiche inviate dal sensore di movimento.

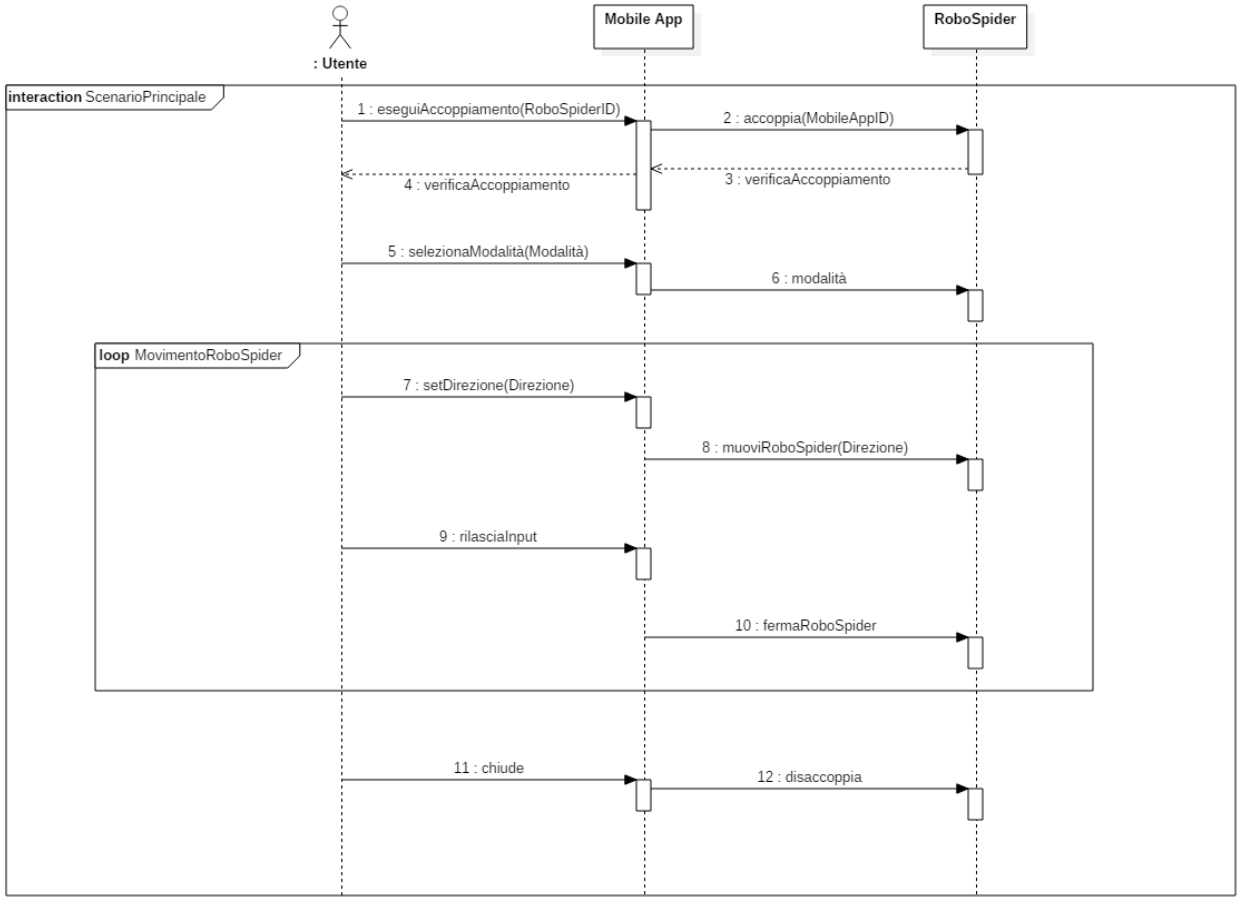
- **Mobile App**

  Utilizza un'interfaccia **Utente**, in cui l'**Utente** seleziona una modalità. In modalità manuale riceve i record dei movimenti. Deve essere accoppiata esattamente con un **RoboSpider** riconosciuto.

- **Sensore**

  In modalità autonoma, rileva eventuali ostacoli presenti nelle vicinanze notificandoli, se presenti, al **RoboSpider.**

Diagramma di Sequenza di Sistema



Contratto CO1: eseguiAccoppiamento

Operazione	eseguiAccoppiamento(**RoboSpider**Id:Stringa)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Nessuna
Post-condizioni	È stato richiesto un accoppiamento tra **MobileApp** e il **RoboSpider**

Contratto CO2: accoppia

Operazione	accoppia(MobileAppId:Stringa)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere richiesto un eseguiAccoppiamento
Post-condizioni	È stata creata una istanza di accoppiamento tra **MobileApp** e il **RoboSpider**

Contratto CO3: selezionaModalità

Operazione	selezionaModalità(modalità:Stringa)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	L'**Utente** ha accoppiato la **MobileApp** con il **RoboSpider**
Post-condizioni	È stata selezionata una modalità.

Contratto CO4: modalità

Operazione	modalità
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere selezionata una modalità
Post-condizioni	È stata eseguita la modalità selezionata

Contratto CO5: setDirezione

Operazione	setDirezione(direzione:Stringa)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere eseguita in modalità manuale
Post-condizioni	È stato richiesta di muovere il **RoboSpider** in una direzione tramite la **MobileApp**

Contratto CO6: muoviRoboSpider

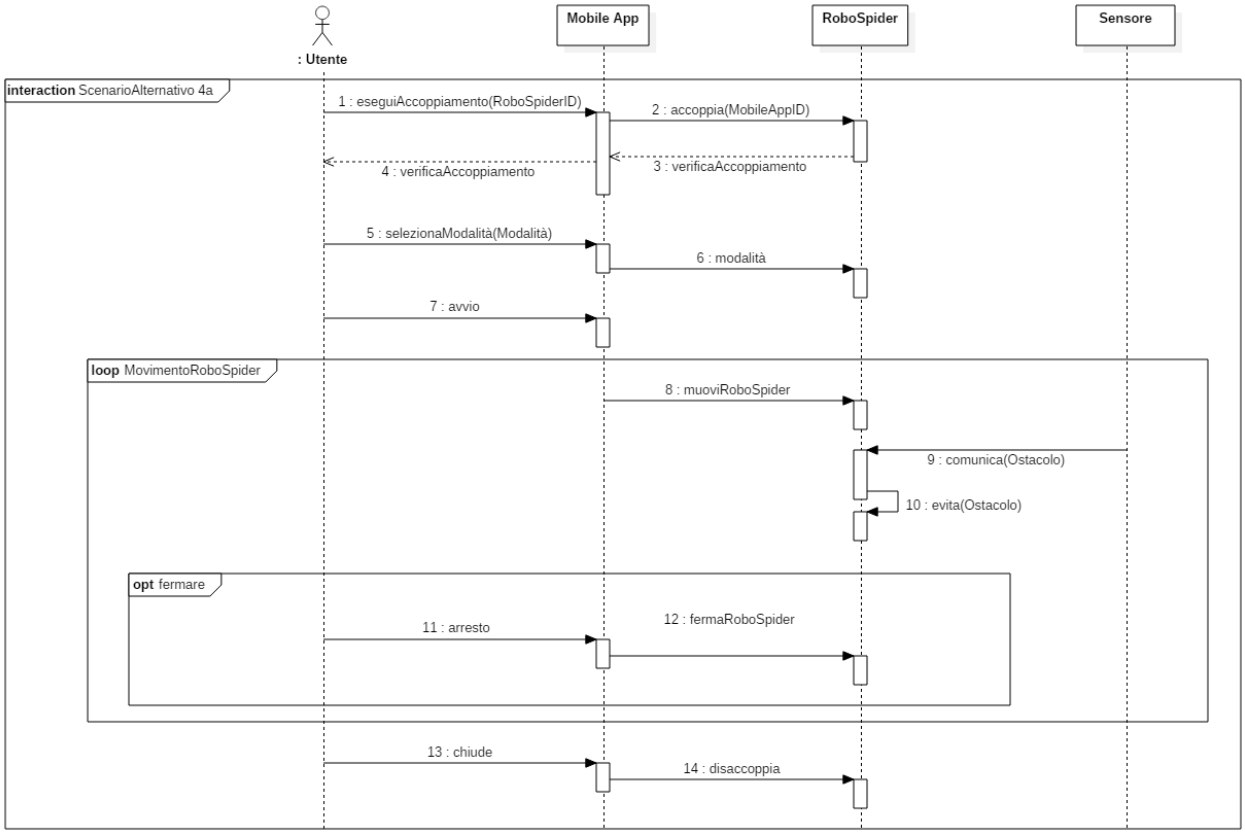
Operazione	muovi**RoboSpider**(direzione:Stringa)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere richiesta una direzione dall'**Utente**
Post-condizioni	È stato richiesto un movimento da eseguire

Contratto CO7: rilasciaInput

Operazione	rilasciaInput
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere in atto un movimento continuo
Post-condizioni	È stato richiesto di non eseguire più il movimento selezionato in precendenza

Contratto CO8: fermaRoboSpider

Operazione	ferma**RoboSpider**
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere in atto un rilascioInput
Post-condizioni	È stato eseguito il rilasciaInput



Contratto CO1: eseguiAccoppiamento

Operazione	eseguiAccoppiamento(**RoboSpider**Id:Stringa)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	nessuna
Post-condizioni	È stato richiesto un accoppiamento tra **MobileApp** e il **RoboSpider**

Contratto CO2: accoppia

Operazione	accoppia(MobileAppId:Stringa)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere richiesto un eseguiAccoppiamento
Post-condizioni	È stata creata una istanza di accoppiamento tra **MobileApp** e il **RoboSpider**

Contratto CO3: selezionaModalità

Operazione	selezionaModalità(modalità:Stringa)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	L'**Utente** ha accoppiato la **MobileApp** con il **RoboSpider**
Post-condizioni	È stata selezionata una modalità.

Contratto CO4: modalità

Operazione	modalità
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere selezionata una modalità
Post-condizioni	È stata eseguita la modalità selezionata

Contratto CO5: avvio

Operazione	avvio
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere selezionata la modalità autonomo da **MobileApp**
Post-condizioni	È stato effettuato l'avvio dei movimenti in autonomo

Contratto CO6: muoviRoboSpider

Operazione	muovi**RoboSpider**
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere richiesto l'avvio
Post-condizioni	È stata eseguita la richiesta di avvio

Contratto CO7: comunica

Operazione	comunica(Ostacolo:Oggetto)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Si deve essere in modalità autonoma
Post-condizioni	Il **Sensore** ha comunicato la presenza di un ostacolo

Contratto CO8: evita

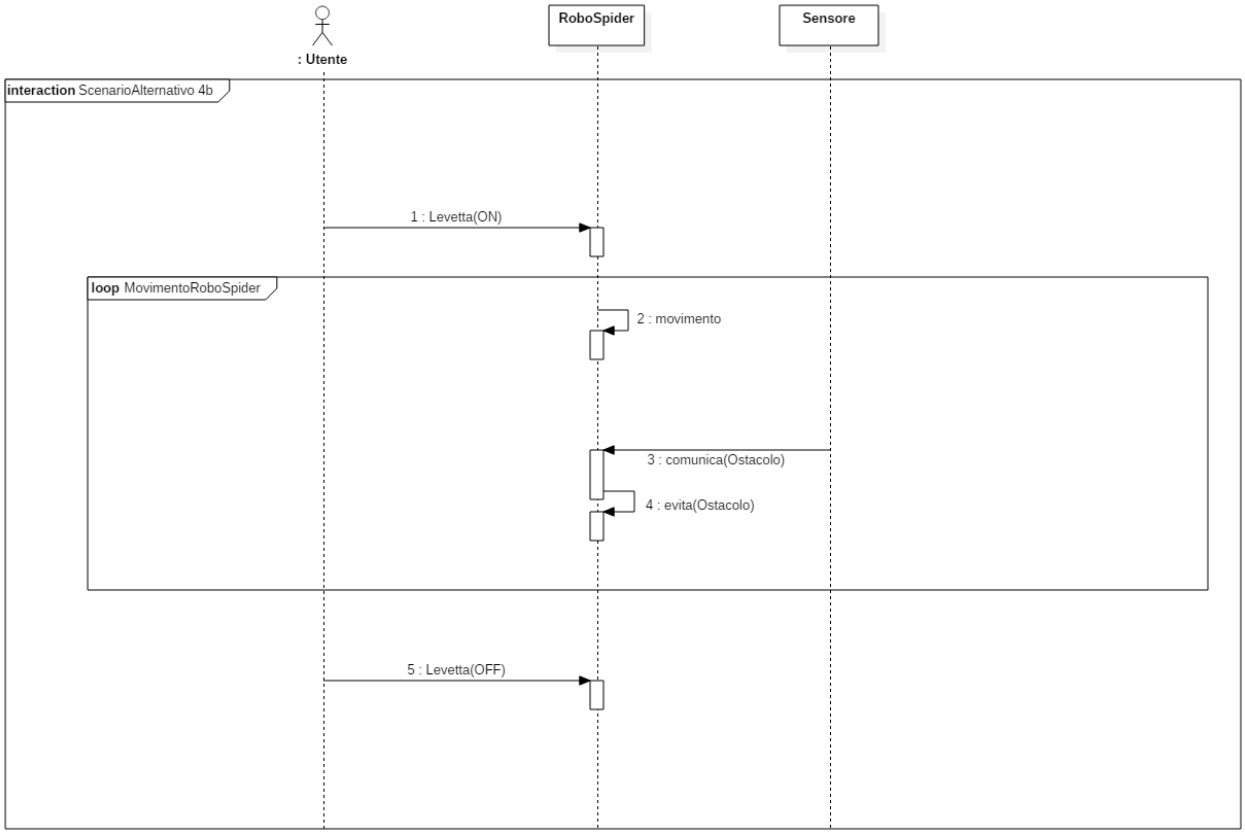
Operazione	evita(Ostacolo:Oggetto)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere presente un ostacolo
Post-condizioni	Il **RoboSpider** ha evitato l'ostacolo

Contratto CO9: arresto

Operazione	arresto
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere in atto un avvio
Post-condizioni	È stato richiesto di eseguire un arresto del **RoboSpider**

Contratto CO10: fermaRoboSpider

Operazione	ferma**RoboSpider**
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere in atto un arresto
Post-condizioni	È stato fermato il **RoboSpider**



Contratto CO1: Levetta

Operazione	Levetta(ON:Oggetto)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Nessuna
Post-condizioni	Il **Sensore** comunica la presenza di un ostacolo

Contratto CO2: movimento

Operazione	movimento
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere avviata in modalità autonoma remota
Post-condizioni	Il **RoboSpider** ha iniziato a muoversi

Contratto CO3: comunica

Operazione	comunica(Ostacolo:Oggetto)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Si deve essere in modalità autonoma
Post-condizioni	Il **Sensore** ha comunicato la presenza di un ostacolo

Contratto CO4: evita

Operazione	evita(Ostacolo:Oggetto)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere presente un ostacolo
Post-condizioni	Il **RoboSpider** ha evitato l'ostacolo

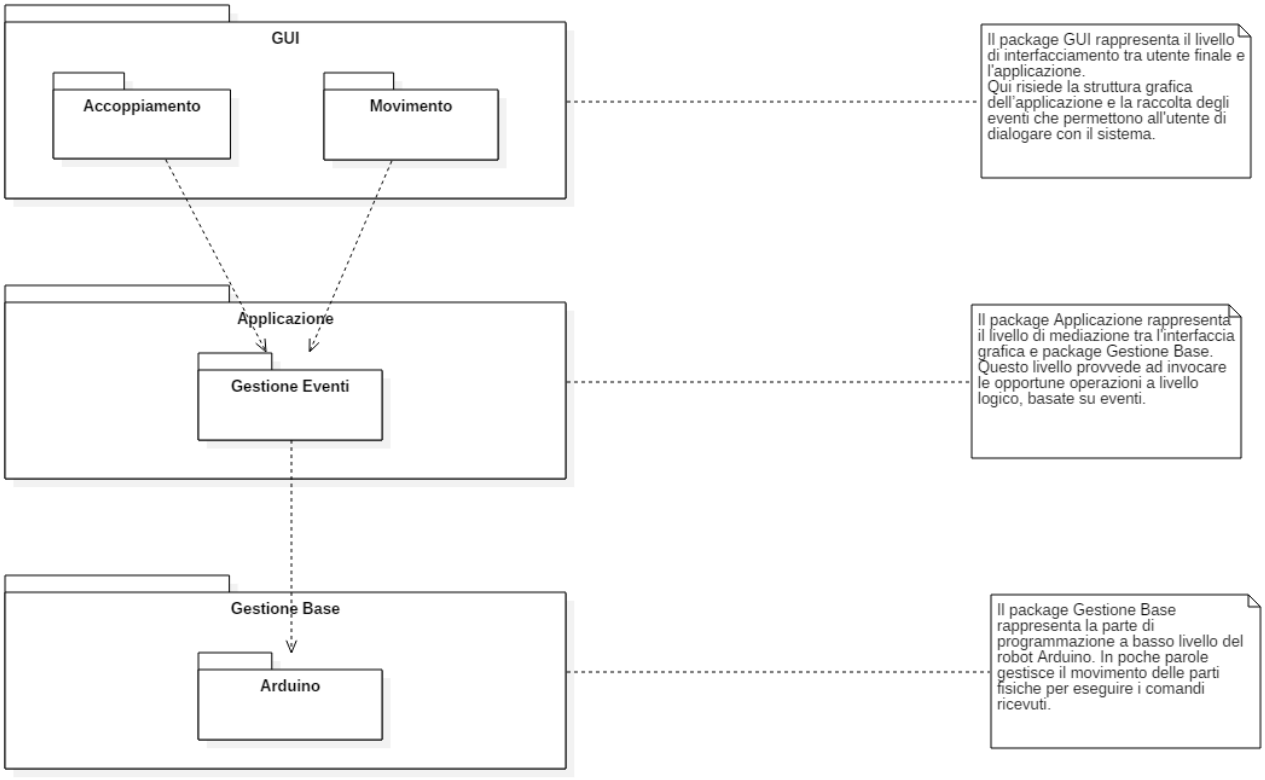
Contratto CO5: Levetta

Operazione	Levetta(OFF:Oggetto)
Riferimenti	Interagire con il **RoboSpider**
Pre-condizioni	Deve essere posizionata su ON la Levetta
Post-condizioni	Il **RoboSpider** si è spento, terminando la modalità autonoma remota

Architettura Logica di Sistema

L'architettura logica è l'organizzazione su larga scala delle classi software in package, sottosistemi e strati. È così chiamata poiché, a questo livello di specifica, non vengono prese decisioni su come questi elementi debbano essere distribuiti sui diversi elaboratori fisici, ma su come deve essere strutturata la logica di dominio a livello software. Un package è un costrutto che permette di prendere un numero arbitrario di elementi UML e raggrupparli in un'unità di livello più alto. Usiamo il **diagramma dei package di UML** per descrivere l'architettura logica del nostro sistema.

Diagramma dei package UML



Descrizione dei package

Package GUI (Graphical User Interface)

Il package GUI rappresenta l'interfacciamento tra **Utente** finale e applicazione. Qui sono presenti la struttura grafica dell'applicazione e gli oggetti che consentono all'**Utente** di dialogare con essa. Ci sono tre sotto-package: **Accoppiamento – Movimento**.
Accoppiamento: fornisce una semplice interfaccia che garantisce l'accoppiamento tra i dispositivi, ovvero quello tra applicazione e robot. Gestione Eventi riceve i messaggi dal package Accoppiamento e si occupa di invocare le operazioni opportune.
Movimento: fornisce l'interfaccia per la selezione della modalità e raccoglie le scelte da parte dell'**Utente** riguardo il movimento del robot. Anche in questo caso il Gestore Eventi riceverà i messaggi da questo package e si occuperà di invocare le operazioni opportune.

Package Applicazione

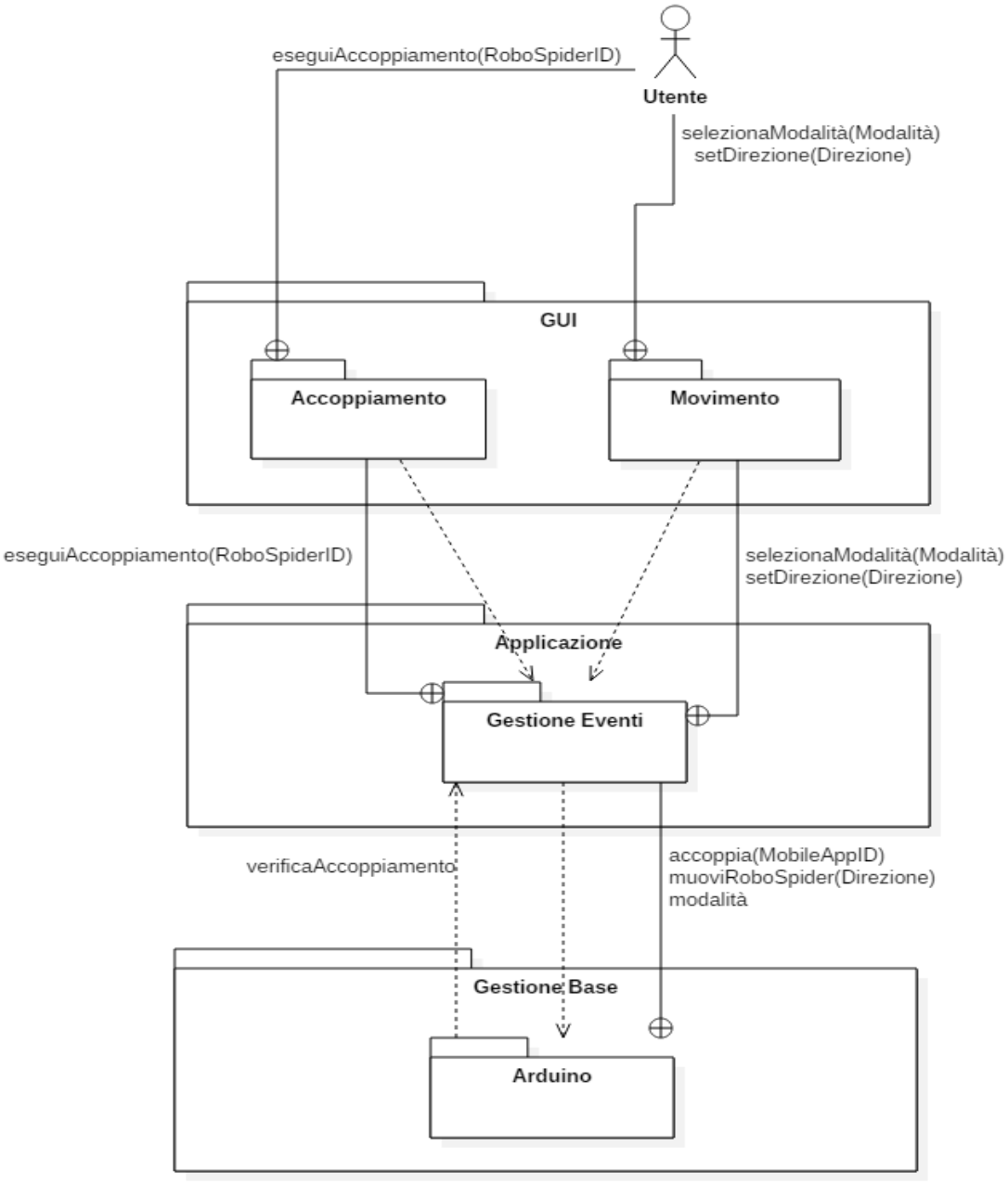
Fornisce la mediazione tra le interazioni a livello **Utente** e la logica di dominio. Tramite Gestione Eventi, vengono rilevati e identificati gli eventi sollevati dall'**Utente** e si provvede a richiamare le opportune funzioni della logica di dominio. Il package Gestione Eventi permette di separare l'interfaccia grafica dalla logica, seguendo il principio di separazione modello-vista.
Qui risiedono anche le operazioni che garantiscono l'accoppiamento e lo scambio di messaggi tra la logica dell'applicazione e quella delle funzionalità in Arduino.

Package Gestione Base

É il livello più basso dell'architettura logica del progetto, utile alla traduzione dei messaggi provenienti dall'applicazione che vengono gestiti a basso livello dal sottopackage Arduino. In poche parole, viene dato un significato ai comandi eseguiti dall'**Utente** che si trasformano in movimento delle parti fisiche del robot.

Esempio di interazione col sistema

Esempio di interazione col sistema riguardante il caso d'uso: Interagire con il **RoboSpider**. L'immagine sotto descrive come avvengono le interazioni tra i vari livelli architetturali, prendendo come esempio il diagramma di sequenza principale: Interagire con il **RoboSpider**. Si può notare che l'interazione con il sistema avviene attraverso l'interfaccia grafica, cioè il primo livello dell'applicazione. Gli eventi raccolti vengono poi passati alla Gestione Base, il quale conosce gli oggetti a basso livello e invoca le operazioni nel sottopackage Arduino, che si occuperà di gestirle in maniera corretta.



Implementazione

In questa sezione verrà discussa l'implementazione del progetto:
gli strumenti e i mezzi adoperati per lo sviluppo della **Mobile App**;
il perché delle scelte adottate;
il reperimento dei materiali per quanto riguarda il **Robospider.**

L'implementazione della **Mobile App** è stata effettuata tramite l'utilizzo di Android Studio, un IDE (*Integrated development environment - ambiente di sviluppo integrato)* Open Source per lo sviluppo nativo di applicazioni Android. La scelta del linguaggio per la stesura del codice è ricaduta su Java, in quanto linguaggio ufficiale di Android.

La **Mobile App** si presenta con cinque file d'Activity e sette file di layout. Le Activity gestiscono gli eventi mentre i file XML di layout gestiscono il Design grafico dell'applicazione.

La prima ad essere implementata è stata la MainActivity ovvero la schermata di avvio, dove sono presenti tre bottoni Exit,Enable Bluetooth e Connect Device.
L'implementazione è stata studiata anche considerando eventuali errori da parte dell'utente: esempio la pressione del bottone Connect Device senza Bluetooth attivato, che è stata risolta aggiungendo una finestra di avviso con la richiesta e in seguito l'attivazione del Bluetooth.

Le altre Activity implementate sono state rispettivamente:

- BluetoothHandlerActivity, in cui viene gestita la scansione Bluetooth dei dispositivi vicini i quali vengono mostrati nella lista posta sotto al bottone Search Device. Dopo aver effettuato una connessione con un dispositivo esso comparirà nella lista Show Paired. Make Discoverable renderà invece il dispositivo visibile a qualsiasi device con Bluetooth attivo per 120 secondi. Connect **Robospider** racchiude il passaggio alla ChooseModalityActivity;
- ChooseModalityActivity, una semplice Activity implementata per dare all'**Utente** la possibilità di scelta fra le due modalità disponibili;
- AutoControlActivity, implementata con un evento slide che dà effetto visivo ON-OFF;
- ManualControlActivity, la grafica richiama un joypad per fornire un' interfaccia facilmente intuibile all'**Utente**;

Per quanto riguarda l'implementazione e la costruzione del **Robospider** si è utilizzato:

- Arduino Genuino Uno ed il suo microcontrollore per la memorizzazione del codice scritta sull'IDE Arduino;
- Sensore Bluetooth per la ricezione dei comandi provenienti dell'applicazione;
- 12 servomotori SG90 per la simulazione del movimento, esattamente 3 servomotori per gamba;
- Un sensore ultrasuoni SRF05 utilizzato come sensore di prossimità;
- Scheletro in plastica realizzato tramite l'utilizzo di una stampante 3D;

Per i componenti elettrici si ringrazia il negozio Arduiner collocato a Rende, mentre per l'utilizzo della stampante 3D si ringrazia HackLab di Cosenza con sede all'UNICAL.

Tutto il progetto è stato realizzato in coppia utilizzando la tecnica del Pair Programming ,ed è stato anche salvato sul servizio offerto da GitHub per permettere modifiche autonome →LINK: https://github.com/VioliHate/roboSpiderINGSW.