UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica Mobile
(Prof. Riccardo Cassinis)

Controllo di Morgul mediante iPhone ®

Elaborato di esame di:

Joan López

Consegnato il: **14 giugno 2012**

Nota: L'autore si scusa per gli errori d'italiano involontariamente commessi.

Sommario

Il lavoro fatto consisteva in realizzare un'applicazione per iPhone ® per spostare il robot Morgul manualmente attraverso l'accelerometro. Cioè, secondo la posizione del dispositivo, il robot si sposterà avanti, indietro, oppure farà una rotazione verso un lato. In questo modo, quelli che vogliono controllare Morgul, riusciranno a guidarlo in maniera intuitiva.

1. Introduzione

Lo scopo del lavoro fatto è sviluppare un'applicazione per iPhone ® che sia in grado di controllare gli spostamenti di Morgul attraverso dei gesti fatti dall'utente con il dispositivo. Nell'interfaccia dell'applicazione appaiono due pulsanti: il primo serve a pausare il robot e il secondo serve a far partire il controllo del robot e a finirlo. C'è anche sul display immagini che indicano la direzione con cui si sposta in quel momento Morgul. Finalmente, esiste anche la possibilità di fare spostare il robot senza tener conto gli ostacoli oppure cercare di evitarli mediante un elemento dell'interfaccia chiamato switch.

2. Il problema affrontato

L'obiettivo del lavoro svolto è fornire un sistema di guida del robot Morgul intuitiva mediante gesti semplici fatti con la mano dell'utente e quindi l'applicazione:

- Deve essere in grado di individuare le posizioni del dispositivo e dopo deve tradurre ogni posizione in un tipo di spostamento: traslazione, rotazione oppure una combinazione tra una traslazione e rotazione.
- Deve saper calcolare la velocità di traslazione e rotazione.
- Deve collegarsi al server per trasmettere i dati necessari verso il programma che gira sul server per guidare il robot: le due velocità e il segno che indica se si attiva la funzione '*safe drive'* per il controllo automatico delle collisioni.

Allora, il lavoro può essere diviso in tre fasi.

2.1. Ottenimento dell'orientamento del dispositivo

Bisogna usare i dati che l'accelerometro dell'iPhone ® fornisce per ottenere i diversi orientamenti. Poi, per ogni orientamento si aggiunge un'operazione che verrà fatta da Morgul.

2.2. Calcolo delle velocità

Per variare le velocità, si terranno conto due coordinate che verranno modificate secondo l'orientamento dell'iPhone ®.

2.3. Collegamento e trasmissione di dati

Occorre fare collegamenti al calcolatore che contiene il lato server per le trasmissioni dei parametri che fanno spostare Morgul. Cioè, questo calcolatore esegue il programma che tradurre i messaggi ricevuti dall'iPhone ® in movimenti di Morgul. S'incarica anche d'inizializzare le variabili opportune per il buon funzionamento del robot e del suo collegamento.

3. La soluzione adottata

3.1. Traduzione dei dati dell'accelerometro in orientamenti

L'accelerometro dell'iPhone ® fornisce l'orientamento del dispositivo usando la forza di gravità registrata nelle tre assi: X, Y e Z (fig. 1). I dati ottenuti dall'accelerometro servono anche ad individuare i cambi nella velocità con cui va il dispositivo. L'accelerazione del dispositivo si misura in g. Cioè, un'accelerazione di 1g è equivalente all'accelerazione fatta dalla forza gravitazionale della Terra: 9,81 m/s^2. E dunque, i valori che possono prendere ogni coordinata sono compresi tra -1 e 1.

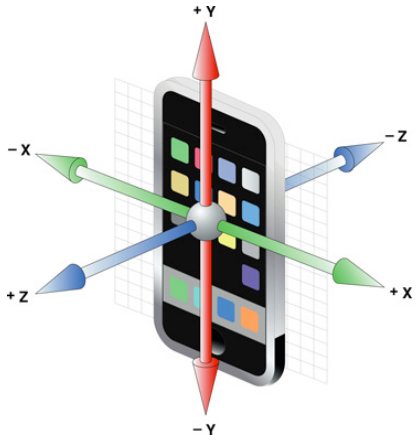

Fig. 1 - Assi

La successiva fase della traduzione, consiste in ottenere l'angolo del dispositivo, cioè quello tra l'accelerazione della componente x inversa e quella della componente y. Per fare questo, si usa la funzione atan2, la variante della funzione arcotangente. Per ogni coppia di parametri x e y diversi da zero, atan2(y,x) è l'angolo tra l'asse x positivo di un piano e il punto dato per le coordinate (x,y).

Dunque, dipendendo del risultato di atan2, riusciamo a individuare ogni orientamento del dispositivo. Sia angolo = atan2(y,x), e x e y le componenti dell'accelerazione (fig. 2):

- Se -2.25 <= angolo <= -0.75: indietro
- Se -0.75 <= angolo <= 0.75: destra
- Se -2.25<= angolo <= 2.25: sinistra
- Se 0.75 <= angolo <= 2.25: avanti

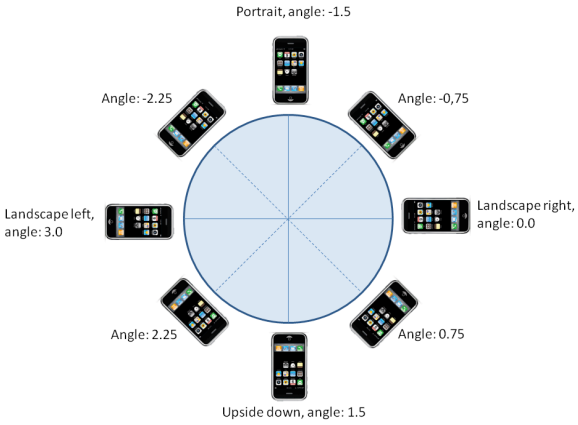

Fig. 2 - Relazione tra gli angoli e l'orientamento

A questo punto, abbiamo individuato gli orientamenti che ci interessano. Ora, bisogna avere qualche modo di ottenere i parametri che verranno spediti verso Morgul secondo l'orientamento.

Attraverso l'array *direcciones* (direzioni), si salva le posizioni date dal dispositivo. Ciòe, se si fa un gesto per andare avanti, si aggiungerà la stringa "avanti" sull'array. Vediamo il suo contenuto:

```
[avanti]
```

Se invece va indietro, si aggiunge la stringa "indietro":

```
[indietro]
```

Per quanto riguarda alle rotazioni, se il gesto precedente è stato andare avanti o indietro, un movimento combinato verrà fatto aggiungendo nel secondo elemento dell'array "destra" o "sinistra":

```
[avanti,destra]

[avanti,sinistra]

[indietro,destra]

[indietro,sinistra]
```

Ma se sta andando avanti e si vuole andare indietro, si vuota l'array e poi s'inizializza con il nuovo movimento.

```
[avanti]

[]

[indietro]

[]

[avanti]
```

3.2. Ottenimento delle velocità

A questo punto, sappiamo il tipo di spostamento che deve eseguire Morgul mediante l'array che abbiamo appena descritto. Ora bisogna calcolare i parametri che devono essere trasmessi al robot. Il funzionamento è semplice: immaginiamo che abbiamo un quadrato diviso dalle due assi (X e Y) e due variabili legate alle coordinate di un punto su questo quadrato xPosition e yPosition. Quando il robot non si muove e, quindi, i valori di velocità valgono 0, abbiamo il nostro punto nell'origine di coordinate del quadrato. Poi, se incliniamo il dispositivo verso destra, soltanto la variabile xPosition verrà incrementata. Oppure, se lo incliniamo verso sinistra, xPosition verrà decrementata. Analogamente succede con l'orientamento verso avanti o indietro con la variabile yPosition. Quando abbiamo un movimento combinato, secondo della direzione che prende il robot, xPosition e yPosition sono modificati come abbiamo appena detto.

Poi, ogni volta che si deve spedire i parametri, si prendono le coordinate della posizione corrente data dalle variabili xPosition e yPosition. Allora, per il calcolo della velocità di traslazione, si ottiene attraverso il prodotto tra la coordinata y e il fattore di traslazione uguale a 3. E per la velocità di rotazione, si ottiene mediante il prodotto tra la coordinata x e il fattore di traslazione uguale a 0.3.

```
float advCoeff = 3.;

float rotCoeff = .3;

int vel = (int) (-(yPosition - dimensioneAltezza/2)*advCoeff);

int rotVel = (int) (-(xPosition -dimensioneLarghezza/2)*rotCoeff);
```

DimensioneAltezza/2 e dimensioneLarghezza/2 servono ad ottenere il punto mezzo del quadrato.

3.3. Trasmissione dei dati

Ogni volta che le velocità vengono calcolate, devono essere trasmesse a Morgul mediante la costruzione di una stringa con questo formato: velocità di traslazione + spazio + velocità di rotazione + spazio + segno di controllo di collisioni (1 o 0, secondo se si vuole attivare l'opzione oppure no). Come il lato server ha bisogno di una stringa di larghezza 11, mettiamo degli spazi alla fine della stringa fino

raggiungere questa dimensione meno uno. È così perché prima di inviarla, bisogna aggiungere un a-capo.

```
NSString *straa = [[NSString alloc]initWithFormat:@"%d %d %d",vel,rotVel,weight];

    while ([straa length]<10) {

        straa = [straa stringByAppendingString:@" "];

    }
```

Poi, si usa la classe connection per le connessioni. S'inizializza un oggetto indicando il nome della macchina (o indirizzo) dove si deve fare il collegamento e il numero della porta:

```
connection *c = [connection alloc]initWithHost:"herbie.ing.unibs.it" andPort:8102];
```

Mediante quest'oggetto si aprono due stream attraverso un socket con lo scopo di leggere e scrivere i dati; si fanno gli scambi di messaggi e, finalmente, si chiude il socket.

Il funzionamento più dettagliato degli scambi di messaggi è questo: una volta aperti gli stream, l'applicazione è pronta per fare i collegamenti a Morgul. Si utilizza il metodo `messages:message` per gli scambi di messaggi. Cioè, il metodo riceve i parametri che devono essere spediti verso Morgul. Quando si è mandata la stringa, l'applicazione aspetta una risposta per riuscire ad andare avanti. Se tutto va bene, il metodo restituirà YES. Invece, se c'è qualche errore, restituirà NO e in questo caso farà fermare il ciclo che chiama al metodo `transmit`, quindi, fermerà Morgul.

Finalmente, si chiudono gli stream attraverso il metodo `close`.

Tutta questa procedura, viene fatta in un Thread e così non blocca l'esecuzione dell'applicazione.

4. Modalità operative

4.1. Componenti necessari

Per fare funzionare il sistema fatto, servono i seguenti:

- Il robot Morgul.
- Il lato server dell'applicazione scritto in C++ su Morgul.
- L'applicazione scritta in objective-c per comandare il robot mediante gesti.
- Un iPhone ®

Per usare il sistema fatto bisogna:

- Che Morgul sia correttamente alimentato.
- Un iPhone ® e l'applicazione installata.
- Essere in grado di collegarsi alla rete 'Robotica' con l'iPhone ®.
- Avere l'iPhone ® aggiornato alla versione iOS 5.0 per installare l'applicazione.

4.2. Modalità di compilazione

Prima di tutto, bisogna dire che l'applicazione si trova in fase di sviluppo e quindi la modalità d'installazione sarebbe diversa se fosse già stato finita e caricata sull'App Store perché basterebbe scaricarla dall'App Store e installarla sull'iPhone ®.

In fase di sviluppo bisogna avere un account di sviluppatore in questo sito web: https://developer.apple.com/ios/manage/overview/index.action (fig. 3.) Si deve pagare circa 80€ per anno.

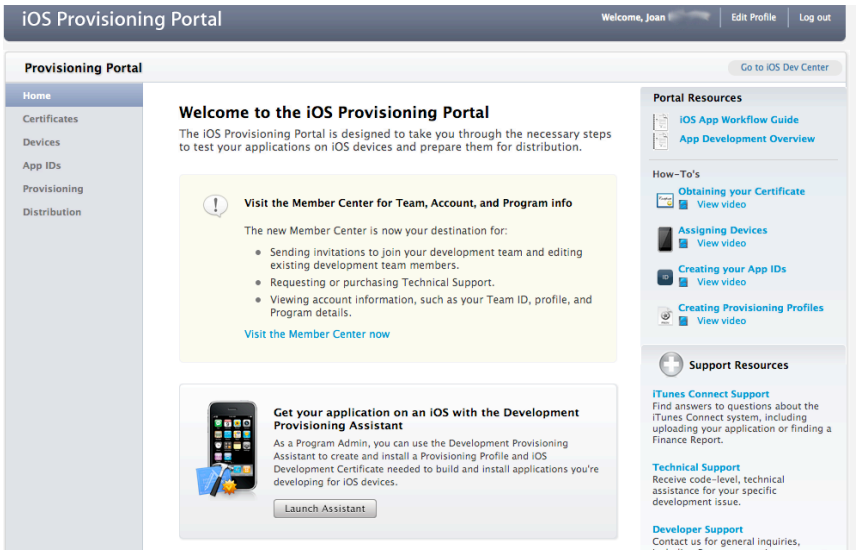

Fig. 3 - Portale per gli sviluppatori di iOS

Una volta registrato, si deve registrare tutti i dispositivi che saranno usati per testare l'applicazione che si sta sviluppando cliccando su Devices (parte sinistra della figura 3). Non si deve dimenticare, però, di creare un certificato su la sezione Certificates per essere in grado di dimostrare che si ha l'account di sviluppatore e, poi, potere testare l'applicazione sul dispositivo.

Poi, bisogna creare un'App ID. È una combinazione di una stringa di 10 caratteri formata da un "Bundle Seed ID" e da un "Bundle Identifier". Un Bundle Seed ID può essere usato per condividere l'accesso al keychain del Mac tra diverse applicazioni che lo stesso sviluppatore può fare con un'unica app ID. Invece, il Bundle Identifier è una stringa che identifica l'applicazione che si sta programmando ed è necessario per permettere installarla sull'iPhone ® in fase di sviluppo. Ecco l'app ID per il sistema fatto: (fig.4).

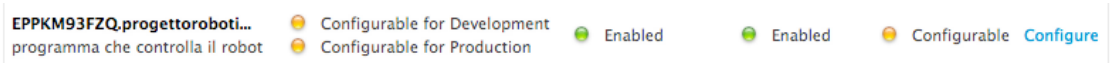

Fig. 4 - App ID

Il "Bundle Seed ID" è EPPKM93FZQ e il "Bundle Identifier" è progettorobotica.

Finalmente, nella sezione Provisioning, si crea un profilo che indica i dispositivi che sono permessi ad usare l'applicazione in sviluppo, qual è l'app id che appartiene all'aplicazione, il certificato dello sviluppatore e il nome del profilo. Poi, si deve scaricare questo profilo e metterlo sull'iPhone ® (figura 5). In questo modo, sarà possibile l'esecuzione dell'applicazione che si sta programmando.

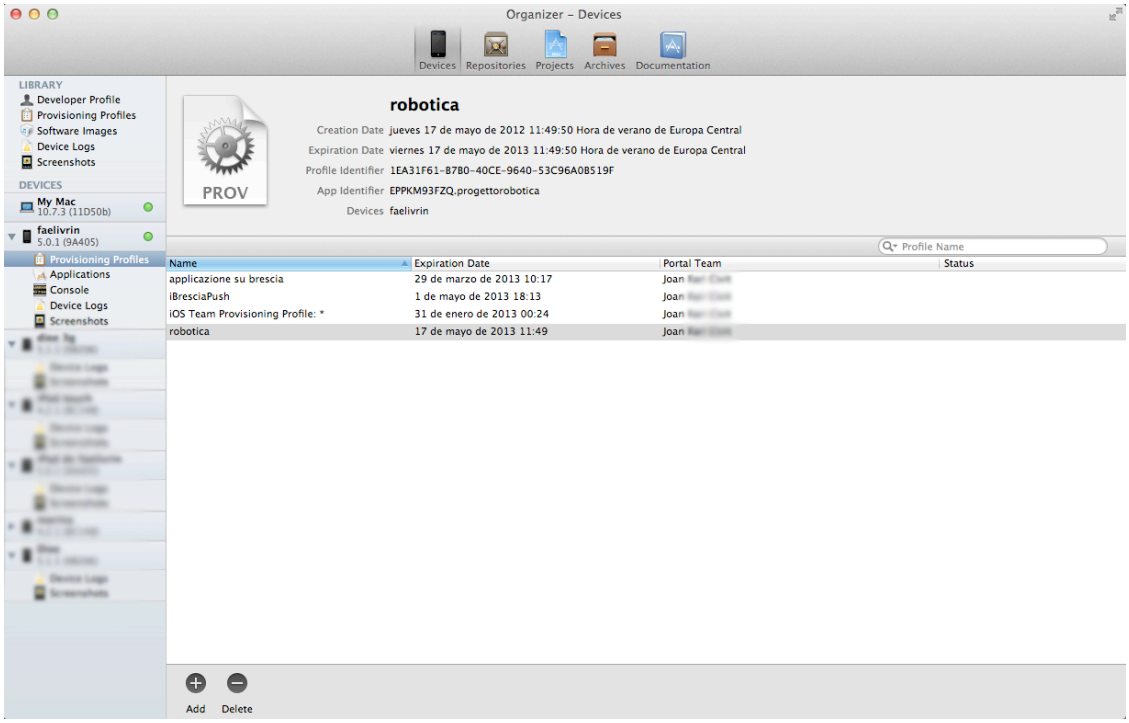

Fig. 5 - Schermata dei profili istallati sull'iPhone ®

4.3. Modalità d'uso di Morgul

Prima di tutto, si deve far partire l'applicazione del lato server sul calcolatore di Morgul. Per farlo, si può collegare a quel calcolatore mediante una sessione SSH col suo indirizzo 192.0.2.10. Ovviamente, il calcolatore con cui si pretende accedere al calcolatore di Morgul, dovrebbe essere collegato sulla stessa rete.

```
ssh eserc1@192.0.2.10
```

Una volta iniziata la sessione, si accende il robot:

```
morgulc -R
```

Quando la sua lampadina lampeggia, vuol dire che è pronto per eseguire qualche programma. Allora, eseguiamo il programma che ci interessa:

```
remoteControlServer -rp /dev/ttyUSB0
```

A questo punto, se non c'è nessun errore, possiamo usare l'applicazione sviluppata. Se vogliamo fermare Morgul, basta digitare questo:

```
morgulc -r
```

5. Conclusioni e sviluppi futuri

Il lavoro fatto può servire come base di un'applicazione più completa. Si potrebbe aggiungere un elemento che permetta di accelerare/rallentare la scala di modifica delle variabili xPosition e yPosition e quindi le velocità di traslazione e rotazione. Cioè, invece di incrementare/decrementare in una unità il valore delle variabili, potrebbero farlo con multipli di 2, di 3, ecc.

Bibliografia

[1] Joe Conway y Aaron Hillegass: *Programación iOS*, ed., Anaya, Barcelona, Ottobre 2011.

Indice

SOMMARIO **1**

1. INTRODUZIONE **1**

2. IL PROBLEMA AFFRONTATO **1**
2.1. Ottenimento dell'orientamento del dispositivo **1**
2.2. Calcolo delle velocità **1**
2.3. Collegamento e trasmissione di dati **1**

3. LA SOLUZIONE ADOTTATA **2**
3.1. Traduzione dei dati dell'accelerometro in orientamenti **2**
3.2. Ottenimento delle velocità **3**
3.3. Trasmissione dei dati **3**

4. MODALITÀ OPERATIVE **4**
4.1. Componenti necessari **4**
4.2. Modalità di compilazione **4**
4.3. Modalità d'uso di Morgul **6**

5. CONCLUSIONI E SVILUPPI FUTURI **6**

BIBLIOGRAFIA **7**

INDICE **8**