AIBO Sony – Oggetto MoveHead

Alessandro Quattro
Laboratory of Applied Intelligent Systems (AIS-Lab)
http://ais-lab.dsi.unimi.it

Sommario

1. L’oggetto MoveHead

1.1 Descrizione

È un oggetto OPEN-R che si occupa della gestione del movimento della testa dell’Aibo.

Accetta dei comandi sottoforma di stringhe e comunica con l’oggetto di sistema OVirtualRobotComm per far eseguire al robot i movimenti corrispondenti al comando ricevuto.

L’oggetto MoveHead accetta i seguenti 15 comandi:

```
"MOVE_BP"
"MOVE_MP"
"MOVE_FP"
"MV_RGHT"
"STOP_MR"
"MV_LEFT"
"STOP_ML"
"MV_FWRD"
"STOP_MF"
"MV_BACK"
"STOP_MB"
"MOVE_UP"
"STOP_MU"
"MV_DOWN"
"STOP_MD"
```

Al ricevimento del comando "MOVE_BP" fa muovere la testa dell’Aibo fino al raggiungimento della posizione testa dietro.

Fig.1 - Posizione testa dietro.

Al ricevimento del comando "MOVE_MP" fa muovere la testa dell’Aibo fino al raggiungimento della posizione testa in mezzo.

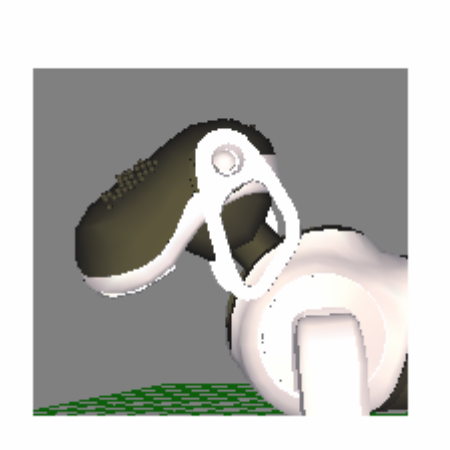

Fig.2 - Posizione testa in mezzo.

Al ricevimento del comando "MOVE_FP" fa muovere la testa dell'Aibo fino al raggiungimento della posizione testa davanti.

Fig.3 - Posizione testa davanti.

Al ricevimento del comando "MV_RGHT" fa girare la testa verso destra fino al raggiungimento dell'angolazione limite (-88°) o fino al ricevimento del comando "STOP_MR".

Fig.4 - Raggiungimento del limite destro.

Al ricevimento del comando "MV_LEFT" fa girare la testa verso sinistra fino al raggiungimento dell'angolazione limite (88°) o fino al ricevimento del comando "STOP_ML".

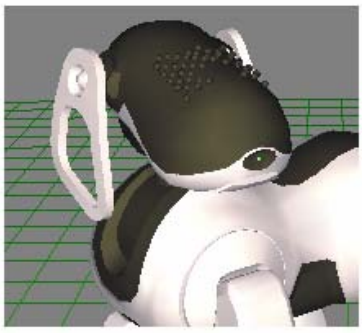

Fig.5 - Raggiungimento del limite sinistro.

Al ricevimento del comando "MV_FWRD" fa muovere il collo in avanti fino al raggiungimento dell'angolazione limite (-75°) o fino al ricevimento del comando "STOP_MF".

Al ricevimento del comando "MV_BACK" fa muovere il collo indietro fino al raggiungimento dell'angolazione limite (0°) o fino al ricevimento del comando "STOP_MB".

Al ricevimento del comando "MOVE_UP" fa muovere la testa verso l'alto (senza muovere il collo) fino al raggiungimento dell'angolazione limite (45°) o fino al ricevimento del comando "STOP_MU".

Fig.6 – Raggiungimento del limite superiore.

Al ricevimento del comando "MV_DOWN" fa muovere la testa verso il basso (senza muovere il collo) fino al raggiungimento dell'angolazione limite (-15°) o fino al ricevimento del comando "STOP_MD".

Fig.7 - Raggiungimento del limite inferiore.

1.2 Utilizzo

Utilizzare MoveHead è molto semplice.

È sufficiente connettere un servizio subject dell'oggetto che intende utilizzare MoveHead al servizio observer "ReceiveCommand" dell'oggetto MoveHead.

Successivamente quando si vorrà far muovere il robot bisognerà inviare una delle stringhe precedentemente elencate al servizio subject precedentemente connesso.

In un'applicazione di test è stato utilizzato l'oggetto Commander per inviare dei comandi all'oggetto MoveHead. L'oggetto Commander è stato dotato di un servizio SendHeadCommand per inviare comandi a MoveHead.

```
void
Commander::MoveHeadMiddlePosition()
{
      char str[8];
      strcpy(str,"MOVE_MP");
      subject[sbjSendHeadCommand]->SetData(str, sizeof(str));
      subject[sbjSendHeadCommand]->NotifyObservers();
}
```

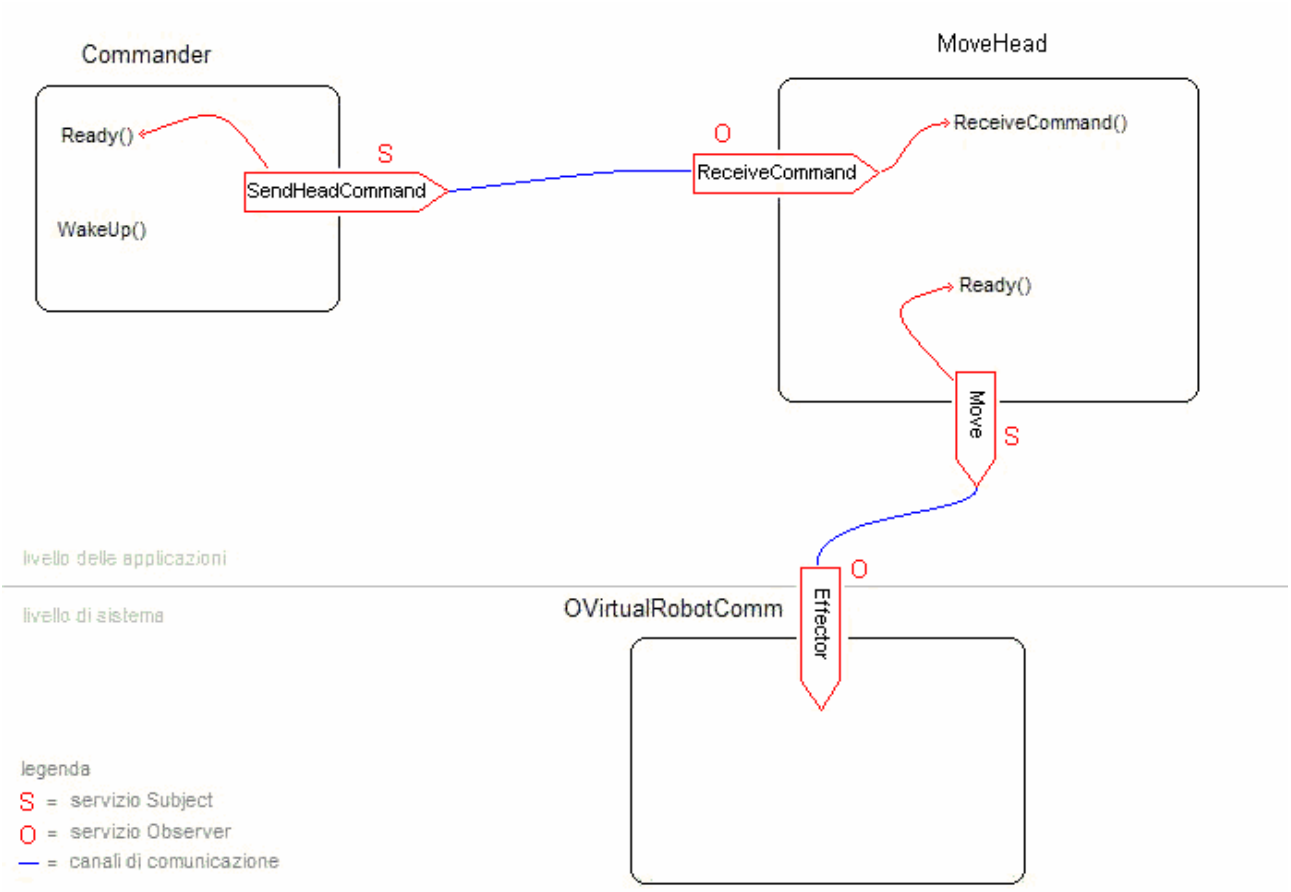

Fig.8 - Applicazione di test.

Per quello che riguarda l'ordine di invio dei comandi è opportuno far notare che non avrà senso inviare i comandi che fermano un movimento se il robot non sta eseguendo quel movimento.

Ad esempio avrà senso chiamare " STOP_MR " solo se l'Aibo sta girando la testa verso destra.

Comunque nel caso venga inviato un comando nell'ordine sbagliato questo non verrà eseguito e l'oggetto MoveHead si rimetterà in attesa di un nuovo comando.

1.3 Funzionamento

Nel file MoveHead.h sono state definite 3 posizioni raggiungibili dal robot e per ciascuna di queste le angolazioni dei 3 joint.

Sono stati anche definiti i diversi stati in cui può trovarsi la testa del robot: ferma (MHS_IDLE), oppure in movimento verso una nuova posizione (es. MHS_MOVING_TO_MIDDLE_POS). Per ogni stato che prevede il movimento verso una nuova posizione è stato definito il numero di blocchi di frame da 128ms in cui questo movimento deve essere diviso (es. MIDDLE_POS_MAX_COUNTER).

nel file MoveHead.h

```
...
//elenco di posizioni raggiungibili dal robot
const double BACK_POS[] = {
    0.0,    // TILT1
    0.0,    // PAN
    0.0,    // TILT2
};
const double MIDDLE_POS[] = {
    -40.0,  // TILT1
    0.0,    // PAN
    25.0,   // TILT2
};
const double FRONT_POS[] = {
    -75.0,   // TILT1
    0.0,     // PAN
    45.0,    // TILT2
};
...

...
//insieme di stati in cui può trovarsi il robot
enum MoveHeadState {
    MHS_IDLE,
    MHS_MOVING_TO_BACK_POS,
    MHS_MOVING_TO_MIDDLE_POS,
     ...
};
...
    static const int MIDDLE_POS_MAX_COUNTER =  16; // 128ms * 16 = 2048ms
...
    double START_ANGLE[3];
    double END_ANGLE[3];
    int MAX_COUNTER;

    MoveHeadState     moveHeadState;
...
```

I diversi stati in cui può trovarsi il robot possono essere paragonati a quelli di un automa a stati finiti (fig.9).

La transizione da uno stato a un altro è attivata da un messaggio.

Il movimento da eseguire dipende dal messaggio che è arrivato e dallo stato in cui si trova il robot.

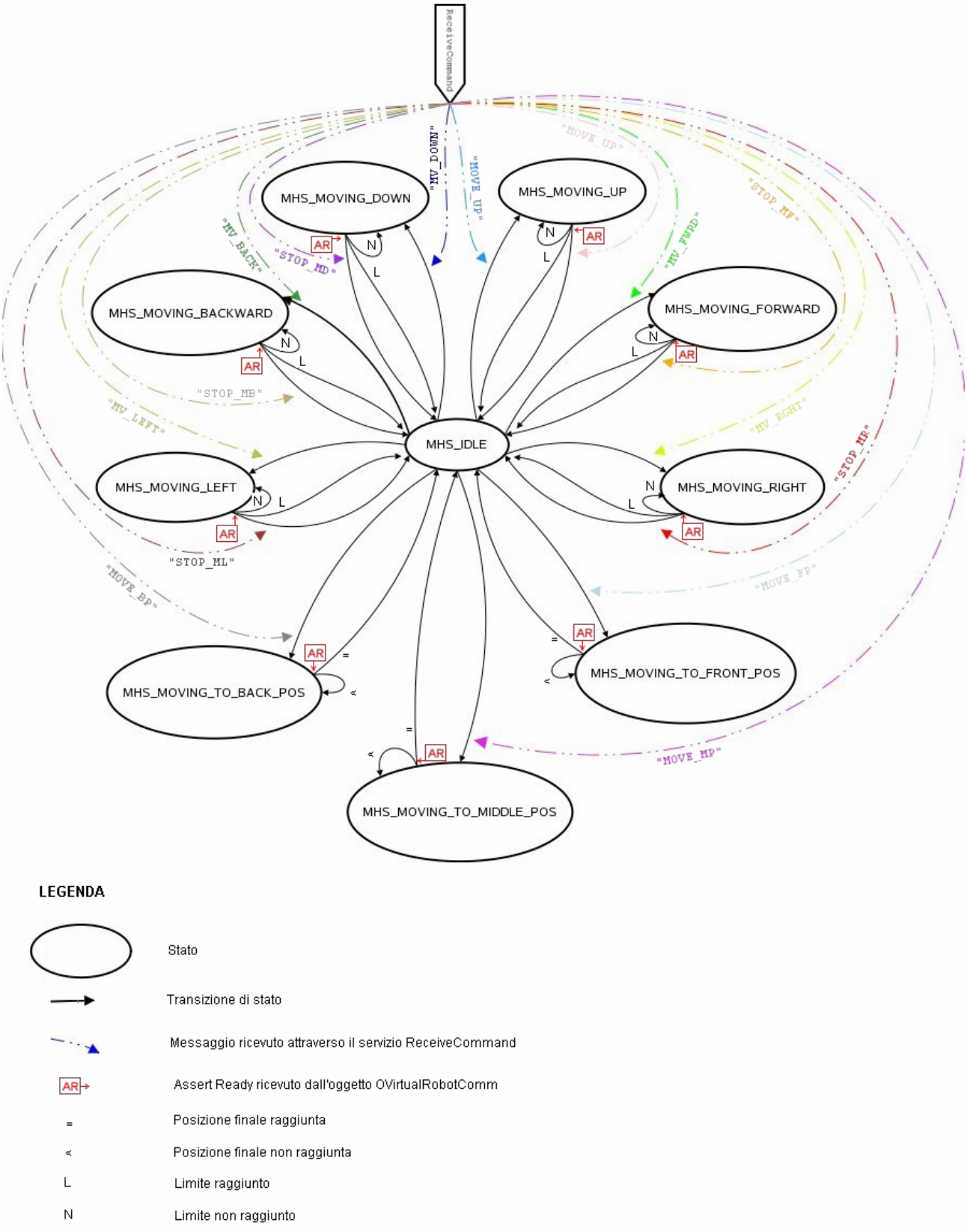

Fig.9 - Stati di MoveHead.

Il nostro oggetto può ricevere 2 tipi di messaggi: una stringa attraverso il servizio observer “ReceiveCommand” o un AssertReady dall’oggetto di sistema OVirtualRobotComm attraverso il servizio subject “Move”.

Al ricevimento di un messaggio di tipo stringa attraverso il servizio observer “ReceiveCommand” viene chiamata la funzione ReceiveCommand().

Al ricevimento di un AssertReady dall’oggetto di sistema OVirtualRobotComm attraverso il servizio subject “Move” viene chiamata la funzione Ready().

La funzione ReceiveCommand() riceve una stringa.

Se questa stringa corrisponde a un comando valido controlla anche lo stato in cui si trova l’oggetto.

Se l’oggetto si trova in uno stato compatibile con il comando rappresentato dalla stringa viene chiamata la funzione preposta all’esecuzione di quel comando.

Ad esempio se viene ricevuta la stringa ”MOVE_MP” e il robot si trova nello stato MHS_IDLE verrà chiamata la funzione MoveToMiddlePos ().

```
...
char* cmd;      //comando da eseguire
char* rcvcmd;   //comando ricevuto
...

void
MoveHead::ReceiveCommand(const ONotifyEvent& event)
{{
    rcvcmd = (char*)event.Data(0); //stringa ricevuta
    //se la stringa ricevuta è compatibile con lo stato in cui si trova il robot
    //la copio in cmd (comando da eseguire) ed eseguo il comando
    //se no invio un ASSERT_READY all'oggetto che ha inviato il comando senza eseguirlo
    ...

    //MOVE TO MIDDLE POSITION
    } else if(strcmp(rcvcmd,"MOVE_MP")==0){
       if(moveHeadState == MHS_IDLE){
          cmd = rcvcmd;
          moveHeadState = MHS_MOVING_TO_MIDDLE_POS;
          positionsSetted = false;
          MoveToMiddlePos();
       }
       else{
          observer[event.ObsIndex()]->AssertReady();
       }

    //MOVE TO FRONT POSITION
    } else if(strcmp(rcvcmd,"MOVE_FP")==0){
...
```

La funzione Ready() non fa altro che richiamare la funzione preposta all’esecuzione dell’ultimo comando ricevuto. Essa viene chiamata ad ogni ricevimento di un AssertReady dall’oggetto di sistema OVirtualRobotComm, potrà quindi essere chiamata ripetutamente durante la transizione dallo stato iniziale allo stato finale del movimento previsto da un comando.

Perciò anche la funzione preposta a eseguire il comando ricevuto potrà essere chiamata più volte fino al raggiungimento dello stato finale previsto dal comando.

```
MoveLegs::Ready(const OReadyEvent& event)
{
...
        //MOVE MIDDLE_POSITION
        else if(strcmp(cmd,"MOVE_MP")==0){
            MoveToMiddlePos();
        }//end
...
}
```

Nelle funzioni preposte a eseguire i comandi ricevuti (es. MoveToMiddlePos()) viene controllato in quale dei possibili stati si trova il robot.

Se lo stato del robot è appena cambiato e corrisponde a uno stato che prevede il movimento tra due posizioni vengono settati i valori degli array rappresentanti le posizioni di inizio e di fine movimento e della variabile che rappresenta il numero di blocchi da 16 frame in cui deve essere diviso il movimento stesso.

Poi, sempre se lo stato in cui si trova il robot corrisponde a uno stato che prevede il movimento tra due posizioni, viene chiamata la funzione MoveJoints().

Quando viene raggiunta la posizione finale la funzione MoveJoints() restituisce il valore MOVING_FINISH e lo stato del robot viene cambiato con quello relativo allo stato successivo.

Nella funzione MoveJoints() il movimento viene diviso in intervalli regolari, viene cercata l'RCRegion libera, con la funzione FindFreeRegion(), e poi chiamata la funzione SetJointValue() che provvede a valorizzare la struttura OCommandVectorData dell'RCRegion libera appena trovata.

Infine viene inviato il messaggio all'oggetto OVirtualRobotComm.

Quando questo riceve il messaggio incrementa di 1 il reference counter dell'RCRegion, processa il messaggio muovendo i joint e poi decrementa di 1 il reference counter liberando l'RCRegion.

Una volta eseguito il movimento l'oggetto di sistema OVirtualRobotComm invia un Assert Ready all'oggetto MoveHead.

Verrà quindi chiamata nuovamente la funzione Ready() che richiamerà a sua volta la funzione preposta all'esecuzione dell'ultimo comando ricevuto. Questo ciclo si interromperà quando sarà stato raggiunto l'unico stato che non prevede movimento, cioè MHS _IDLE.

Come esempio si riporta il codice della funzione privata MoveToMiddlePos() che fa muovere la testa fino a raggiungere la posizione testa in mezzo.

Le funzioni private MoveJoints(), FindFreeRegion() e SetJointValue() sono identiche a quelle dell'oggetto MoveLegs.

```
void
MoveHead::MoveToMiddlePos()
{
    if (moveHeadState == MHS_MOVING_TO_MIDDLE_POS) {
        if(!positionsSetted) {
            for (int i = 0; i < NUM_JOINTS; i++) {
                OJointValue current;
                OPENR::GetJointValue(jointID[i], &current);
                START_ANGLE[i] = degrees(current.value/1000000.0);
                END_ANGLE[i] = MIDDLE_POS[i];
            }
            MAX_COUNTER = MIDDLE_POS_MAX_COUNTER;
            positionsSetted = true;//nuove posizioni settate
        }
        MovingResult r = MoveJoints();
        if (r == MOVING_FINISH) {
            moveHeadState = MHS_IDLE;//stato successivo
            positionsSetted = false;//posizioni non settate (vengono settate all'inizio della
funzione)
            counter = -1;         //azzero il contatore del frame raggiunto

            subject[sbjMove]->ClearBuffer();
            observer[obsReceiveCommand]->AssertReady();
        }
    }
}
```