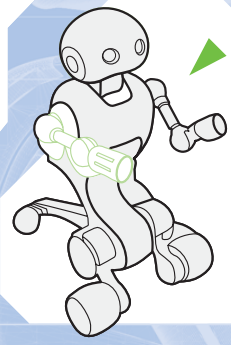
F71-LAB-p.12-14-ok 17-01-2007 8:46 Pagina 12

IL VASSOIO PORTAOGGETTI

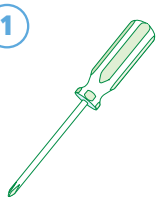


Ecco il primo degli 'hand tool': il piccolo vassoio portaoggetti che potrai collegare agli arti superiori di I-D01.

La fase di montaggio corrente è stata chiamata 'Braccia e Hand Tool', perché riguarda l'ultima parte del robot non ancora montata (le braccia, appunto) e i primi strumenti che possono essere collegati agli arti superiori, tramite i connettori alle loro estremità. Dell'insieme di tool faranno parte anche dispositivi elettronici e meccanici, ma si comincia con qualcosa di più semplice, il vassoio portaoggetti. Esso è costituito da diversi componenti, che sono tutti allegati a questo fascicolo. In particolare, trovi le due metà del vassoio propriamente detto, a forma di V arrotondata, e le due parti del connettore che consentirà l'aggancio con i corrispondenti elementi degli arti del robot. Completano l'insieme di elementi allegati un albero di metallo e quattro viti da 3x10 mm, che permetteranno di montare il tutto. Ovviamente, per il momento il vassoio non può essere utilizzato: sarà necessario prima completare almeno uno degli arti superiori.

COSA TI SERVE

1



1. Un cacciavite magnetico a croce piccolo

COMPONENTI

1. Prima metà del connettore
2. Seconda metà del connettore
3. 4 viti da 3x10 mm
4. Albero di metallo
5. Parte sinistra del vassoio
6. Parte destra del vassoio

IL VASSOIO

MONTAGGIO

1 Prendi la parte sinistra del vassoio e l'albero di metallo. Inserisci l'albero nell'apposito foro circolare della metà del vassoio, indicato nell'immagine a destra.

F71-LAB-p.12-14-ok 17-01-2007 8:46 Pagina 13

IL VASSOIO (CONTINUA)

MONTAGGIO

2 In seguito, prendi le due metà del connettore e apprestati a comporle (a sinistra). Nota come esse presentino alcuni piccoli pioli e le corrispondenti fessure, per garantire un buon fissaggio.

3 Componi le due parti del connettore e poi premi su di esse, così da fissarle bene l'una all'altra (sotto).

4 Prendi il connettore che hai appena assemblato e posizionalo sopra l'asse metallico. Orienta il connettore come mostrato nell'immagine sotto, in modo che l'estremità a forma di croce sia puntata verso l'esterno del vassoio: dovrà collegarsi alla mano di I-D01.

5 Prendi ora la parte destra del vassoio e avvicinala a quella sinistra (immagine a destra). Come vedi, le due parti presentano alcuni fori telescopici e supporti circolari che dovranno combaciare nella composizione del vassoio.

F71-LAB-p.12-14-ok 17-01-2007 8:46 Pagina 14

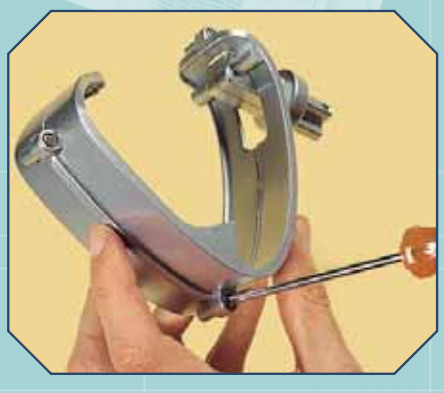
COMPOSIZIONE DEL VASSOIO

MONTAGGIO

1 Poni una parte del vassoio sull'altra, in modo che i supporti e i fori coincidano, poi premi con attenzione, ma anche con decisione, sulle due metà, in modo che aderiscano bene l'una all'altra. Presta particolare attenzione al piccolo piolo posto all'estremità della parte destra, indicato nell'immagine a destra.

2 Adesso che le due parti del vassoio sono aderenti l'una all'altra, verifica che il connettore sia libero di ruotare attorno all'asse metallico (immagine a sinistra). Se così non fosse, prova a rimuovere il connettore e a ripulire con un cacciavite il suo foro circolare da eventuali sbavature o residui.

3 Controlla che tutto sia a posto (a sinistra), poi recupera le quattro viti da 3x10 mm allegate a questo fascicolo e serrale negli appositi fori della parte sinistra del vassoio (immagine a destra).



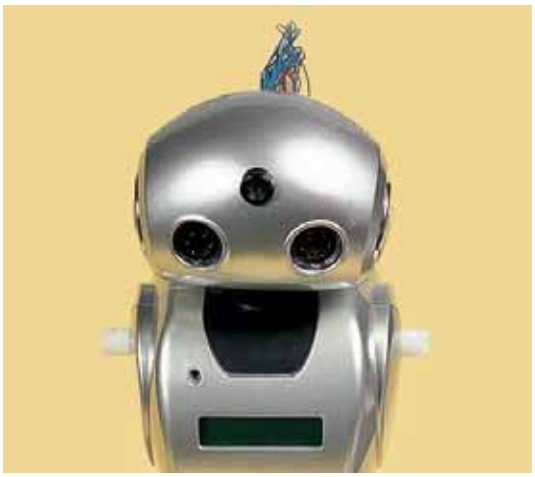
4 Una volta fissate le viti, il vassoio è completo (a destra). Mettilo da parte, in attesa di poterlo utilizzare quando almeno uno degli arti superiori del robot sarà pronto.

F71-LAB-p.15-16-ok 17-01-2007 8:47 Pagina 15

I-DroidO1, IL C-LIKE E JAVA

Il terzo CD include il supporto per il C-like e per Java: adesso si può programmare I-D01 a livello avanzato.

Grazie al terzo CD-ROM, sono ora disponibili tutti i livelli di programmazione per I-Droid01. Infatti, dopo i software di controllo e il Visual C-like Editor, è ora possibile realizzare programmi direttamente in codice C-like e in Java. Nel primo caso, la procedura per l'esecuzione è sostanzialmente identica a quella relativa al Visual C-like Editor: in sostanza, i programmi vengono compilati ed eseguiti direttamente a bordo di I-D01, grazie al modulo B&V e al suo firmware. Per quanto riguarda i codici scritti in Java, invece, la situazione è diversa: i programmi vengono compilati ed eseguiti a bordo del PC, che poi comunica con I-Droid01 tramite il collegamento Bluetooth e ne utilizza le funzioni per realizzare i comportamenti voluti. La programmazione in Java, in effetti, non fa altro che utilizzare il robot come se fosse una periferica del PC molto sofisticata, in grado di fornire dati in ingresso (provenienti dai vari sensori) e di produrre azioni (grazie ai LED, ai motori e alla 'voce' del robot). Se i codici scritti in C-like utilizzano costrutti sostanzialmente equivalenti alle funzioni impiegate nel Visual Editor, per Java l'impostazione è molto diversa, proprio a causa del diverso approccio che viene esibito nei confronti di I-Droid01. Per fare un esempio, l'esecuzione dei programmi in Java sfrutta spesso la scrittura e la lettura di particolari 'registri di comunicazione', ossia determinati parametri utilizzati per lo scambio di informazioni tra PC e I-D01, mentre con il C-like non si arriva a questo livello di dettaglio. In un certo senso, quindi, con Java è necessario programmare più 'a basso livello'. Per questo motivo, la programmazione in Java è rivolta soprattutto a utenti esperti. D'altra parte, però, non si hanno problemi di complessità dei codici realizzati: essendo compilati ed elaborati dal PC, infatti, essi possono sfruttare tutte le potenzialità di calcolo fornite da quest'ultimo. Ci si può sbizzarrire davvero, quindi.



La programmazione di I-Droid01 in C-like e, soprattutto, in Java permette di elaborare comportamenti anche molto sofisticati, che sfruttano appieno le funzioni del robot.

IN C-LIKE...

La programmazione in C-like mette a disposizione le stesse funzioni implementate dai blocchi del Visual C-like Editor. Per conoscere i costrutti utilizzabili si può consultare la documentazione relativa inclusa nel terzo CD. Per poter eseguire i programmi scritti in C-like è necessario utilizzare comunque il Visual C-like Editor: è tramite esso, infatti, che il codice (scritto con qualsiasi editor di testi) va aperto, trasferito a bordo del robot, compilato e infine eseguito, con la stessa procedura impiegata per i diagrammi a blocchi.

...E IN JAVA

Utilizzando Java, la situazione è un po' diversa. Come detto, la compilazione e l'esecuzione sono mantenute a bordo del PC, che poi comunica in corso di esecuzione con I-D01, via Bluetooth. I programmi scritti in Java devono quindi utilizzare le funzioni di collegamento e interscambio di dati con il robot: a tal proposito possono essere impiegate diverse librerie di supporto che realizzano questo tipo di interfaccia. Nel CD ne è inclusa una (la Java Communications API), ma ne esistono di alternative (ad esempio ne sono disponibili alcune 'open source' sul sito www.rxtx.org). In ogni caso, è necessario includere tali librerie nel codice scritto in Java e assicurarsi che esse siano 'linkate' nella lista di classi utilizzate nella compilazione dei codici. Come già indicato, la programmazione in Java è destinata

F71-LAB-p.15-16-ok 17-01-2007 8:47 Pagina 16

a utenti esperti, che abbiano conoscenze relative a tale linguaggio e alle procedure di compilazione ed esecuzione. Nel terzo CD sono presenti, oltre alle librerie Java, anche diversi esempi, alcuni dei quali sviluppati in ambiente Eclipse (www.eclipse.org), un'applicazione per la realizzazione di progetti Java. Per eseguire gli esempi, è sufficiente scompattare nella stessa cartella di destinazione i file di archivio java-binaries.zip, java-demo.zip e java-library.zip, presenti nel CD, poi lanciare il file run.bat, ottenuto nella stessa cartella di destinazione: sarà richiesto di indicare uno tra i possibili esempi e la porta COM creata dal modulo Bluetooth all'atto di collegamento tra PC e robot. Nel corso dei prossimi fascicoli, comunque, vedremo più nel dettaglio altri esempi di programmi costruiti in C-like e in Java.

ESEMPI DI PROGRAMMAZIONE

Esempio di una stessa applicazione in C-like e in Java, per mostrarne la diversa complessità. Il codice in C-like può essere copiato su un file nel Visual Editor e poi trasferito su I-D01. Il codice Java, nel quale non sono incluse le funzioni accessorie, invece, non può essere usato se non dopo un adeguato completamento. Le problematiche relative alla programmazione in Java di I-D01 verranno affrontate nei prossimi fascicoli.

[COMANDI VOCALI: CODICE C-LIKE]

```
#include "c-like.h"
#include "robot.h"

/** Dichiarazione dei comportamenti */
declare( behavior(ComandoZero) );
declare( behavior(ComandoUno) );
declare( behavior(ComandoDue) );

/** Dichiarazione delle procedure */
void LampeggiaOcchiRossi();
void LampeggiaOcchiGialli();
void SpegniOcchi();

/** Dichiarazione del comportamento di base */
define( behavior(Main) ) {
        start(ComandoZero);
        start(ComandoUno);
        start(ComandoDue);
}

/** Dichiarazione del comportamento ComandoZero */
define( behavior(ComandoZero) ) {
        local(voice_cmd) = wait_for(voice_cmd, zero);
        SpegniOcchi();
        stop(ComandoUno);
        stop(ComandoDue);
        end();
}

/** Dichiarazione del comportamento ComandoUno */
define( behavior(ComandoUno) ) {
        local(voice_cmd) = wait_for(voice_cmd, one);
        SpegniOcchi();
        LampeggiaOcchiRossi();
}

/** Dichiarazione del comportamento ComandoDue */
define( behavior(ComandoDue) ) {
        local(voice_cmd) = wait_for(voice_cmd, two);
        SpegniOcchi();
        LampeggiaOcchiGialli();
}

/** Dichiarazione della procedura LampeggiaOcchiRossi */
void LampeggiaOcchiRossi() {
        SpegniOcchi();
        led_blink(LED_RED);
}

/** Dichiarazione della procedura LampeggiaOcchiGialli */
void LampeggiaOcchiGialli() {
        SpegniOcchi();
        led_blink(LED_YELLOW);
}

/** Dichiarazione della procedura SpegniOcchi */
void SpegniOcchi() {
        led_off(LED_ALL);
}
```

[COMANDI VOCALI: CODICE JAVA]

```
import communication.handler.ProtocolHandler;
import communication.handler.internal.HeadData;
import communication.handler.internal.HeadRegister;
import communication.handler.internal.InternalHandler;
import communication.handler.internal.InternalModule;
import communication.handler.internal.VoiceRegister;
import communication.transport.ConnectionProvider;

public class VoiceCommander {
    private static String portName;
    private ProtocolHandler protocol;
    private InternalHandler internal;

    /** @param args */
    public static void main(String[] args) {
        if (args.length < 1) {
            System.err.println("Specify the serial port to use (e.g. COM5)");
            return;
        }
        portName = args[0];
        new VoiceCommander().run();
    }

    private void run() {
        if (!connect())
            return;

        int[] buf1 = new int[1];
        int[] buf2 = new int[2];
        try {
            // set current wordset to WS 2
            // (to allow for "Command" + "Number")
            buf1[0] = 2;
            internal.writeRegister(InternalModule.VOICE, VoiceRegister.WORDSET, buf1);
            // read recognition results for reset
            internal.readRegister(InternalModule.VOICE, VoiceRegister.COMMAND, buf2);
            // loop until we exit or press enter key
            boolean exit = false;
            while (!exit && System.in.available() <= 0) {
                Thread.sleep(500); // poll recognition status every 0.5s
                internal.readRegister(InternalModule.VOICE, VoiceRegister.COMMAND, buf2); // read rec results
                // check result is valid
                if (buf2[0] == 0)
                    continue;
                // check it's in WS 11 (numbers)
                if (buf2[1] != 11)
                    continue;
                // choose action
                int leds = -1, phrase = 0;
                switch (buf2[0])
                {
                case 1: // Zero
                    spegniOcchi();
                    exit = true;
                    break;
                case 2: // One
                    occhiRossi();
                    break;
                case 3: // Two
                    occhiGialli();
                    break;
                }
            }
        }
        catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        disconnect();
    }
```