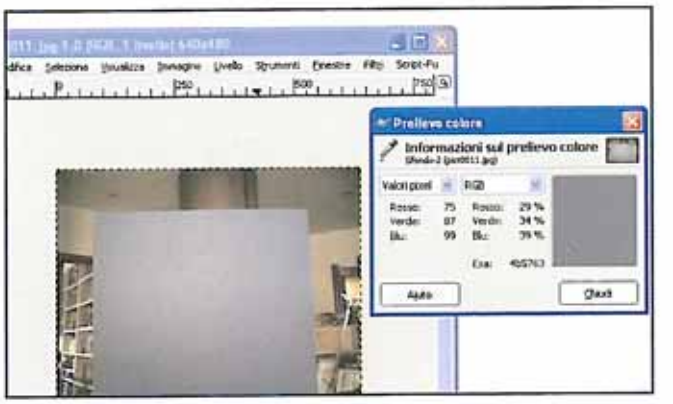
# UN 'CLONE' IN C-LIKE DI SEGUIVISIONE

**Alcuni comportamenti predefiniti attivabili su I-Droid01 sono facilmente riproducibili utilizzando il linguaggio di programmazione C-like. Nel codice presentato in questo esempio realizzeremo un semplice clone del comportamento SeguiVisione, introducendo un'interessante variante.**

Nel seguente esempio in C-like vediamo come implementare un programma 'clone' del comportamento predefinito di I-Droid01 'SeguiVisione'. Ricordiamo che quest'ultimo fa sì che il robot 'segua' con la propria testa i movimenti di un oggetto opportunamente colorato, rilevato per mezzo della CMOS camera. Il riconoscimento degli oggetti avviene elaborando le immagini acquisite dal sensore: i pixel aventi un colore simile a quello indicato come 'target' vengono raggruppati al fine di localizzare aree contigue che corrispondano agli oggetti identificati. Nel comportamento predefinito tale colore è impostato su tonalità simili a quelle delle pelle umana e non è modificabile. In questo esempio, tuttavia, abbiamo aggiunto la possibilità di personalizzare a proprio piacimento il colore degli oggetti 'seguiti' da I-Droid01. Più precisamente è possibile settare, nel codice C-like del programma, due differenti colori. Per passare da un colore all'altro è sufficiente impartire i comandi vocali 'uno' e 'due'. Con il comando 'zero', invece, l'applicazione viene fatta terminare.

## IMPOSTAZIONE DEI COLORI

Per far sì che I-Droid01 riconosca in modo adeguato i colori che intendiamo settare come target, è necessario impostare questi ultimi con una certa precisione. La CMOS camera, infatti, non percepisce i colori esattamente come l'occhio umano. Così, quello che alla nostra vista si presenta come un giallo intenso, ad esempio, potrebbe essere percepito come un giallo meno saturo. Effettuare la calibrazione dei colori è un'operazione molto semplice. Per prima cosa dovrai procurarti una coppia di cartoncini colorati (con due colori diversi) di dimensioni pari a circa 15x15 centimetri. Ognuno dei due cartoncini sarà l'oggetto che I-Droid01 'seguirà' quando saranno impartiti, rispettivamente,




*Sopra, l'immagine di un cartoncino colorato scattata da I-Droid01 viene analizzata con un programma di grafica per rilevare le tre componenti di colore rosso, verde e blu.*

i comandi vocali 'uno' e 'due'. Dopo aver fatto ciò, fai scattare al robot una foto a ciascuno dei due cartoncini. Per ottenere il colore 'percepito' dalla CMOS camera, è necessario aprire le immagini con un programma di grafica (come Gimp, scaricabile gratuitamente dal sito www.gimp.org) e utilizzare lo strumento 'contagocce' per avere le informazioni sui colori nelle immagini stesse, in scala RGB (rosso, verde e blu). Per ognuno dei tre canali di colore sarà restituito un valore compreso tra 0 e 255. Annota la terna ottenuta per i due cartoncini: tali valori ti serviranno per settare in maniera opportuna i colori target nel codice C-like del programma.

## IL CODICE DEL PROGRAMMA

Analizziamo ora il codice del programma. Dopo l'importazione dei file di libreria **robot.h** e **c-like.h**, necessari per il funzionamento dell'applicazione, sono presenti due variabili **X** e **Y**, adibite alla memorizzazione del baricentro dell'oggetto identificato da I-Droid01. Oltre al comportamento **Main** sono presenti altri due comportamenti,

## ESEMPI DI PROGRAMMAZIONE

**Codice C-like del programma 'clone' di SeguiVisione, descritto in queste pagine. Il codice dovrà essere memorizzato in un file denominato visione.clike, caricato sul robot e compilato utilizzando il software Visual C-like Editor. Oltre al comportamento principale (Main) sono presenti altri due comportamenti, SeguiColore e ComandiVocali. Ricorda di 'personalizzare' il codice del programma inserendo le opportune terne nelle istruzioni set_tracking_color.**

### CODICE C-LIKE DEL PROGRAMMA

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

int X = 0;
int Y = 0;

declare( behavior(ComandiVocali) );
declare( behavior(SeguiColore) );

void ComandoZero();
void ComandoUno();
void ComandoDue();

define( behavior(Main))
{

      set_tracking_color(-,-,-,40); //Sostituire ai trattini i valori RGB
      led_off(LED_ALL);
      led_on(LED_RED);
      start(ComandiVocali);
      start(SeguiColore);

} // Main

define( behavior(ComandiVocali) )
{
      local(voice_cmd) = wait_for (voice_cmd, update);
      switch (local(voice_cmd))
      {
      case CMD_ZERO:
            ComandoZero();
            break;

      case CMD_ONE:
            ComandoUno();
            break;

      case CMD_TWO:
            ComandoDue();
            break;
      }

} // ComandiVocali

define( behavior(SeguiColore) )
{

      local(vision) = wait_for (vision, update);

      X = local(vision).x;
      Y = local(vision).y;
      lcd_clear();
      lcd_write_int(1,1,X);
      lcd_write_int(2,1,Y);
      if (X > 35) {
            head_pan_r(1);
      } else if (X < -35) {
            head_pan_r(-1);
      }

      if (Y > 20) {
            head_tilt_r(1);
      } else if (Y < -20) {
            head_tilt_r(-1);
      }

} // SeguiColore

void ComandoZero()
{

      led_off(LED_ALL);
      end();

}

void ComandoUno()
{

      set_tracking_color(-,-,-,40); //Sostituire ai trattini i valori RGB
      led_off(LED_ALL);
      led_on(LED_RED);

}

void ComandoDue()
{

      set_tracking_color(-,-,-,40); //Sostituire ai trattini i valori RGB
      led_off(LED_ALL);
      led_on(LED_YELLOW);

}
```

**ComandiVocali** e **SeguiColore**. Il primo si occupa di cambiare il colore target, in base al comando impostato vocalmente ('uno' o 'due'), oppure terminare il programma (comando 'zero'). Vengono inoltre impostati due diversi colori per i LED degli occhi, in base al comando ricevuto. Il secondo comportamento rappresenta invece il 'cuore' del programma. In base al baricentro dell'oggetto identificato dal robot, la testa viene spostata in modo da mantenere l'inquadratura 'centrata'. Nel tuo codice, che dovrai memorizzare in un file denominato **visione.clike**, ricordati di sostituire i primi tre parametri (relativi al colore) nelle istruzioni **set_tracking_color**, in base alle tue rilevazioni. L'ultimo parametro (la soglia) dovrà invece restare impostato su un valore pari a 40. Le coordinate del baricentro sono mostrate sul display di I-Droid01, tramite le istruzioni **lcd_write**. Passiamo ora ad analizzare brevemente il comportamento **Main**: esso inizializza il colore target di default (scegliendo un colore tra quelli rilevati sperimentalmente) e avvia gli altri comportamenti del programma.

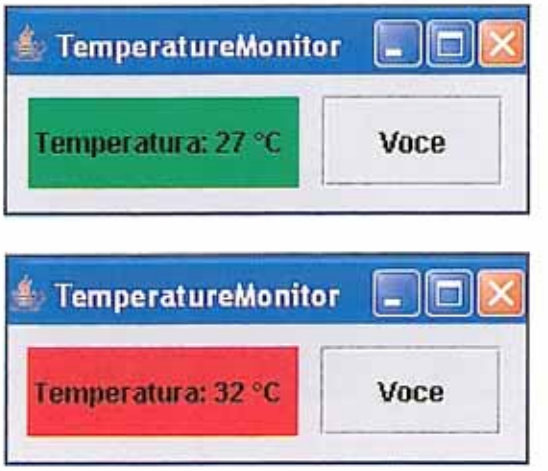
# UN MONITOR PER LA TEMPERATURA

**Abbiamo visto negli esempi in Java dei fascicoli precedenti come sia possibile e relativamente semplice accedere ai registri di I-D01 per leggere il valore rilevato da alcuni dei suoi sensori. Nel nuovo programma proposto in questo fascicolo realizzeremo una semplice interfaccia per monitorare la temperatura ambientale.**

Nel programma Java di esempio proposto nelle prossime pagine, vedremo come sia semplice realizzare un'interfaccia grafica per il monitoraggio della temperatura rilevata da I-Droid01. Quest'ultima viene misurata dallo specifico sensore posto nello zaino del robot e gestito, come già detto, dalla scheda elettronica Arms. L'interfaccia è molto semplice: sulla parte sinistra è presente una casella testuale che visualizza la temperatura rilevata; su quella destra, invece, compare un pulsante per mezzo del quale è possibile far pronunciare al robot la temperatura corrente. La casella di testo mostra due diversi colori in base al valore misurato: verde se inferiore ai 30 gradi centigradi, rosso, invece, se uguale o superiore.

## TEMPERATUREMONITOR

L'unica classe da cui è costituita l'applicazione è **TemperatureMonitor**. Essa implementa sia i metodi per la realizzazione e per la gestione dell'interfaccia grafica sia quelli per l'accesso ai registri del robot. Essendo il programma costituito da una sola classe, è sufficiente inserire l'intero codice in un unico file denominato **TemperatureMonitor.java**. Per la compilazione e l'esecuzione dell'applicazione valgono le solite note viste nei fascicoli precedenti. Nella prima parte del codice Java sono presenti le dichiarazioni di pacchetto e di importazione delle classi ausiliarie necessarie per il funzionamento del programma. Oltre alle classi per la gestione degli elementi grafici, sono presenti quelle per la comunicazione con il robot e per l'accesso ai suoi registri: notiamo come siano coinvolti i pacchetti relativi alle due schede elettroniche Arms e Voice. Il sensore di temperatura, in effetti, è gestito proprio dal primo di questi due moduli. Il secondo, invece, è necessario per far annunciare al robot la temperatura misurata. La classe **TemperatureMonitor** estende **JFrame**, adibita alla realizzazione della finestra grafica dell'applicazione. L'interfaccia del programma è 'minimale': come detto è presente una casella di testo per la visualizzazione della temperatura e un pulsante per attivare la pronuncia




*Qui sopra, il pannello per monitorare la temperatura, mentre viene rilevato un valore superiore a 30 gradi centigradi. Più in alto, invece, lo stesso display mentre il sensore del robot rileva una temperatura ambientale pari a 27 gradi.*


*Sopra, la schermata del prompt corrispondente al lancio del programma TemperatureMonitor; viene indicata come parametro di avvio la stringa 'COM3', indicante la porta seriale virtuale utilizzata in questo caso per stabilire la connessione via Bluetooth tra il robot e il PC.*

## ESEMPI DI PROGRAMMAZIONE

**Esempio di codice Java per l'implementazione di una classe per la realizzazione di un'interfaccia grafica adibita al monitoraggio della temperatura ambientale, rilevata dall'apposito sensore elettronico montato nello zaino di I-Droid01.**

### CLASSE TEMPERATUREMONITOR

```
package communication.examples;

import communication.handler.ProtocolHandler;
import communication.handler.internal.InternalHandler;
import communication.transport.ConnectionProvider;
import communication.handler.internal.InternalModule;
import communication.handler.internal.VoiceRegister;
import communication.handler.internal.ArmsRegister;
import java.io.IOException;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TemperatureMonitor extends JFrame {

    private static String portName;
    private ProtocolHandler protocol;
    private InternalHandler internal;

    private JPanel panel;
    private JLabel temp;
    private JButton speak;

    Thread t;
    boolean active = false;

    public static void main ( [...] )

    public TemperatureMonitor() { [...] }

    private class RefreshDisplay implements Runnable {

        int x = 0;
        public void run() {
            while(active) {
                x = checkTemperature();
                if (x < 30) {
                    temp.setBackground(Color.GREEN);
                    temp.setText(" Temperatura: " +
Integer.toString(x) + " ∞C");
                } else {
                    temp.setBackground(Color.RED);
                    temp.setText(" Temperatura: " +
Integer.toString(x) + " ∞C");
                }
                try {
                    Thread.sleep(2000);
                } catch (Exception e) {}
            }
        }

    } // class RefreshDisplay

    private void sayTemperature() { [...] }

    private boolean connetti() { [...] }

    private void disconnetti() { [...] }

} // class TemperatureMonitor
```

di quest'ultima da parte di I-Droid01. Il metodo **main** crea un'istanza di **TemperatureMonitor**, invocandone il costruttore. Prima di fare questo, però, **main** controlla che sia stata indicata dall'utente all'avvio del programma la porta seriale virtuale per comunicare con il robot. Vediamo ora come è strutturato il codice del costruttore. Per prima cosa viene associata all'evento 'chiusura della finestra' la terminazione del programma. Sono poi istanziati e posizionati sul pannello dell'applicazione i diversi elementi grafici. Per la disposizione di questi ultimi viene utilizzato un 'layout assoluto' che stabilisce la precisa posizione di ciascun oggetto grafico, mediante le chiamate ai metodi **setBounds**. I primi due parametri indicano le coordinate (X e Y) in cui posizionare l'elemento, mentre la terza e la quarta denotano la sua dimensione (larghezza e altezza). Tutti i parametri sono espressi in pixel. Alla pressione del pulsante **speak** viene associata l'invocazione del metodo **sayTemperature**. Infine viene attivata la connessione con il robot e avviato il thread **t** di tipo **RefreshDisplay**. Analizzando quest'ultimo, il metodo **run** è composto da un ciclo **while**, eseguito ripetutamente finché il valore della variabile booleana **active** rimane 'true'. Con una cadenza pari a due secondi, impostata tramite l'istruzione **Thread.sleep(2000)**, viene rilevata la temperatura ambientale misurata dal sensore. Per fare questo viene chiamato un altro metodo, **checkTemperature**, di cui parleremo tra breve. Una volta memorizzato il valore della temperatura in un'apposita variabile **X**, viene impostato il colore di sfondo della casella di visualizzazione: verde se **X** è minore di 30, rosso altrimenti. Occupiamoci adesso del metodo **checkTemperature**. Come già accennato in precedenza, quest'ultimo effettua una lettura nel registro di I-Droid01 relativo alla gestione del sensore di temperatura. Qualora si verificassero errori durante l'accesso ai registri del robot, viene restituito il valore –1 per segnalare l'anomalia. Il terz'ultimo

## ESEMPI DI PROGRAMMAZIONE

**Codice Java dei metodi della classe TemperatureMonitor. Sono mostrati per primi il metodo principale e il costruttore. Quest'ultimo si occupa anche della visualizzazione dell'interfaccia grafica sul monitor del PC e di lanciare la connessione con il robot, invocando il metodo connetti. A seguire sono indicati altri due metodi, il primo apposito per la lettura della temperatura rilevata dal sensore (checkTemperature) e il secondo per l'annuncio di tale temperatura da parte di I-Droid01 tramite il modulo Voice (sayTemperature). I metodi connetti e disconnetti permettono come al solito di attivare e disattivare la connessione tra il robot e il PC tramite comunicazione Bluetooth. Per il codice di questi ultimi si può fare riferimento al fascicolo 77, pagine 13 e 14.**

### METODI DELLA CLASSE TEMPERATUREMONITOR

```
public static void main(String[] args) {

    if (args.length < 1) {
        System.err.println("Specifica la porta da usare per la
connessione (es. COM5)");
        return;
    }

    portName = args[0];
    TemperatureMonitor gui = new TemperatureMonitor();

} // main


public TemperatureMonitor() {

    super("TemperatureMonitor");

    addWindowListener(
        new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                active = false;
                disconnetti();
                System.exit(0);
            }
    });

    setBounds(0,0,240,95);
    panel = new JPanel();
    panel.setLayout(null);

    temp = new JLabel(" Temperatura: " + " ∞C");
    temp.setBounds(10,10,120,40);
    temp.setOpaque(true);
    speak = new JButton("Voce");
    speak.setBounds(140,10,80,40);
    speak.setFocusable(false);

    speak.addActionListener(
        new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                sayTemperature();
            }
    });

    panel.add(temp);
    panel.add(speak);
    getContentPane().add(panel);

    if (!connetti()) System.exit(0);
    t = new Thread(new RefreshDisplay());
    active = true;
    t.start();
    setVisible(true);

} // class constructor


private int checkTemperature() {

    int[] buf = new int[1];
    try {

        internal.readRegister(InternalModule.ARMS,
ArmsRegister.TEMPERATURE, buf);
        return buf[0];

    } catch (Exception e) {

        e.printStackTrace();
        return -1;

    }

} // checkTemperature


private void sayTemperature() {

    int[] buf = new int[1];
    try {

        internal.readRegister(InternalModule.ARMS,
ArmsRegister.TEMPERATURE, buf);
        internal.writeRegister(InternalModule.VOICE,
VoiceRegister.TEMPERATURE, buf);
        System.out.println("Speak!");

    } catch (Exception e) {

        e.printStackTrace();

    }

} // sayTemperature
```

metodo, **sayTemperature**, legge il registro associato al sensore di temperatura e attiva l'annuncio da parte di I-Droid01 della quantità rilevata. Per fare questo è sufficiente scrivere tale quantità in un apposito registro del modulo Voice, più precisamente **VoiceRegister.TEMPERATURE**. Gli ultimi due metodi presenti nella classe **TemperatureMonitor** sono i consueti **connetti** e **disconnetti**, che rispettivamente attivano e disattivano la connessione Bluetooth tra il robot e il PC. Il codice di questi ultimi metodi è del tutto identico a quello già mostrato nel fascicolo 77, pagine 13 e 14, e utilizzato in vari programmi Java presentati nei precedenti fascicoli. Nel testare il programma qui descritto, fai molta attenzione a non sottoporre il sensore a temperature troppo elevate: oltre a danneggiare questo delicato componente, vi è il rischio di provocare danni ai circuiti adiacenti e alla stessa struttura di I-Droid01.