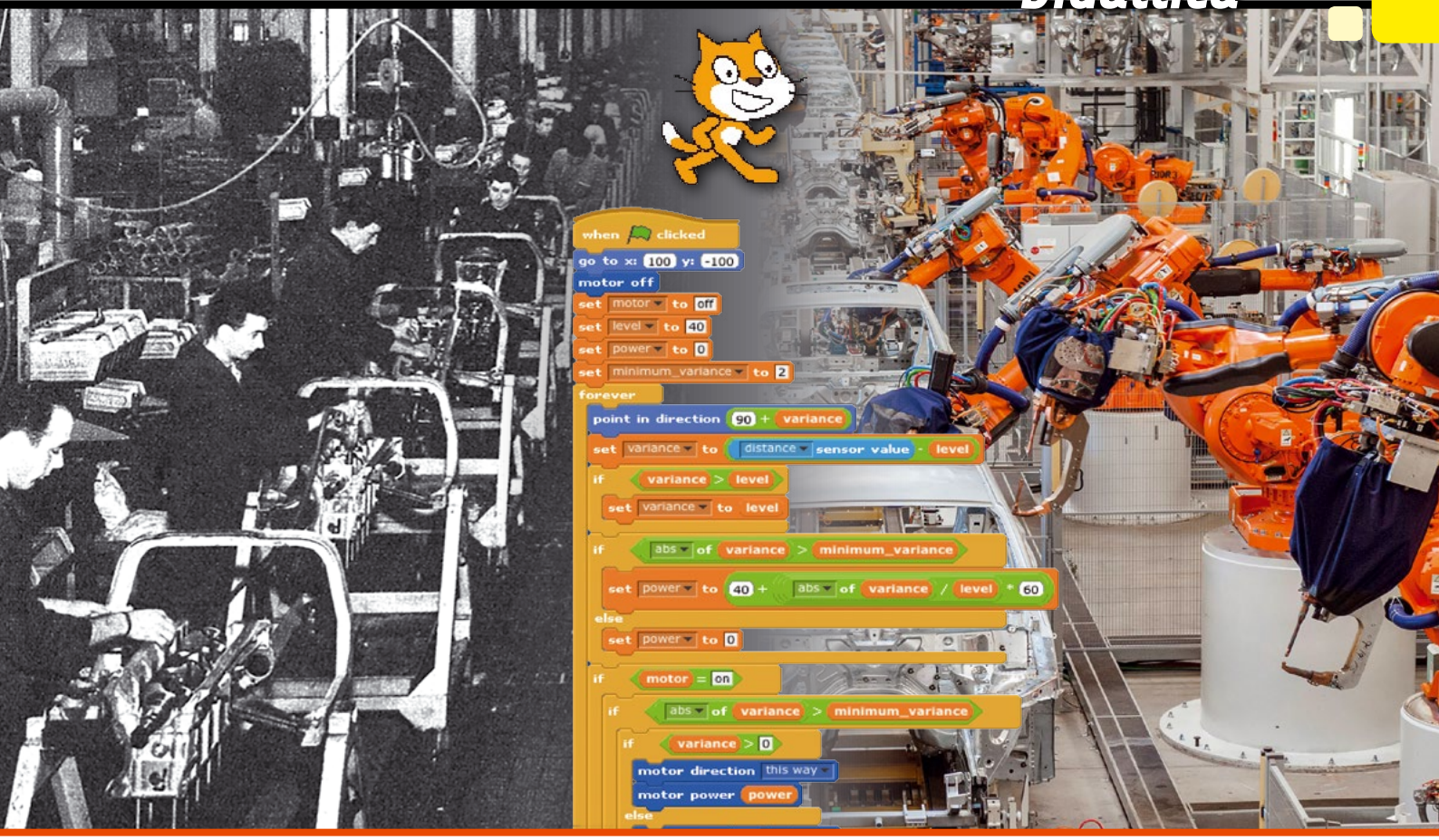



# SCRATCH: LA PROGRAMMAZIONE RESA FACILE

di MARCO MAGAGNIN

**Ambiente di sviluppo studiato particolarmente per chi vuole muovere i primi passi nel mondo della programmazione. È rivolto a tutti, senza limiti di età: conta solo la voglia di imparare divertendosi. Prima puntata.**

In questi ultimi anni abbiamo presentato una serie di articoli dedicati al mondo embedded, con particolare attenzione alle soluzioni basate su GNU/Linux. Il filo conduttore di questa scelta è stato "facilitare" il passaggio a queste tecnologie a chi ha un background "elettronico", ed è eventualmente già avvezzo al mondo informatico per aver utilizzato Arduino o altri microcontrollori "programmabili". Sempre in base a questa "visione" abbiamo approfondito gli aspetti legati al primo impatto con tali sistemi, ossia la "messa in moto" iniziale, la configurazione e la personalizzazione per un utilizzo il più immediato possibile, senza partire con spiegazioni teoriche sull'architettura dei calcolatori, le reti, i sistemi operativi, i Web Server o i database. Per questi aspetti abbiamo proposto percorsi di confi-

gurazione guidata passo-passo. Quando necessitava la scrittura di qualche programma abbiamo fornito il listato, commentato con le indicazioni necessarie all'installazione e all'utilizzo. Per lo stesso motivo ci siamo sempre tenuti lontani dal fornire indicazioni sul come "programmare" da sé le proprie applicazioni.
In compenso abbiamo sempre proposto l'utilizzo di linguaggi e ambienti alla portata di tutti, pur diffusamente presenti nello sviluppo delle applicazioni professionali d'avanguardia. D'altra parte, affrontare la spiegazione delle specifiche di ciascuno dei linguaggi utilizzati avrebbe occupato uno spazio inaccettabile in una rivista, senza peraltro essere di grande aiuto per chi non ha ancora "digerito" i fondamenti della programmazione. Un elenco di "referenze" non aiuta molto a capire come combinarle tra di loro per risolvere un problema in modo compiuto...
È anche vero che per imparare a programmare, così come a leggere e a scrivere, bisogna pur fare riferimento ad un qualche linguaggio: leggere significa portare a sé un messaggio creato da altri, mentre scrivere vuol dire esprimere in modo codificato un proprio pensiero, materializzarlo perché possa giungere ad altri ed essere compreso.
Una volta il concetto di programmazione si applicava pressoché unicamente alla programmazione dei calcolatori elettronici; oggi è esteso ad un'infinità di entità che compongono l'ambiente nel quale viviamo. Già si dice programmare il videoregistratore, piuttosto che la radiosveglia, o l'impianto di irrigazione, la gestione dell'energia nella casa, la sicurezza e anche i calcolatori. Qualcuno obietterà che un conto è programmare un elettrodomestico con la pressione di una sequenza di tasti, altro è scrivere un videogioco interattivo sulla rete: vero, ma è anche vero che sono sulla bocca di tutti termini come domotica, teleassistenza sanitaria, digitalizzazione, Internet delle cose, innovazione, automazione, robotica e simili.
Tutti questi neologismi, pur non avendo ancora una definizione formalizzata, che permetta di riferirsi e individuare qualcosa di ben definito, nascondono una ben precisa realtà: spiegare a qualcosa che assomiglia molto ad un computer, cosa deve fare per ottenere ciò che abbiamo in mente; e questo si ottiene, almeno per ora, con la programmazione, che come qualsiasi attività espressiva, richiede di aver studiato accuratamente il problema che dobbiamo risolvere, aver identificato una soluzione, verificata in tutti i suoi dettagli e condizioni operative e poi espressa in una forma di scrittura comprensibile al "computer" o qualsivoglia altra diavoleria che si incaricherà, come un fedele maggiordomo, di eseguire "scrupolosamente" le nostre disposizioni, fossero anche "accendere il barbeque di domenica alle 11 dentro il garage di fianco all'automobile"!
Ora si stanno affacciando sul mercato i primi sistemi che si ispirano alla filosofia dell'Internet delle cose. I più recenti modelli di lavatrici hanno a bordo un sistema operativo simil-android, che permette il collegamento in WiFi ed il controllo remoto dallo smartphone o da qualsivoglia dispositivo collegato in rete.
Nella "nuvola" fanno capolino i primi servizi che permettono di elaborare le cosiddette "scene" ovvero risposte attive condizionate dallo stato complessivo e ragionato di diversi "sensori" (per esempio avvisare di spegnere l'asciugacapelli se la lavatrice è in funzione). Le istruzioni sono del tipo "IF THEN ELSE", "AND", "OR" e sono associate ad eventi di tipo cronologico (calendario) o di stato fisico come, alla mattina (?), se piove o peggio, se non piove, e simili.
Avere dimestichezza con i concetti base della programmazione non fa certo male anche in questi casi, che saranno sempre più diffusi. E aiuteranno molto anche chi vorrà o dovrà vedersela con i sistemi computerizzati "veri".
Se analizziamo la crisi che stiamo attraversando, in compagnia del resto del mondo, è ascrivibile anche se non unicamente alla distanza esistente tra le capacità medie della popolazione attuale e le esigenze di conoscenza e formazione richieste dall'evoluzione del mondo produttivo e dei servizi.
In un passato non troppo remoto (che sarà mai un secolo...?!) per vivere una vita normale, secondo gli standard di allora, non era necessario sapere leggere e scrivere: quello che serviva lo si imparava sul "campo" (In senso sia figurato, sia agricolo!).
Poi la rivoluzione industriale ha richiesto, a chi lavorava in fabbrica di sapere leggere e scrivere: in realtà, più leggere che scrivere. La crescita degli addetti alle macchine di produzione non poteva essere più coordinata con il metodo dell'esempio sul campo, ma bisognava essere in grado di leggere ordini di lavorazione, impostazioni di macchine, numeri di riferimento, eccetera. A seguire, scolarizzazione di massa, trasmissioni televisive "Non è mai troppo tardi", corsi delle 150 ore e via di seguito, per risolvere il problema "nazionale".
Ora siamo da capo: la nuova esigenza non è più avere "la-

voratori" che sappiano leggere ed eseguire disposizioni scritte, oltre naturalmente a saper "fare" ciò che viene richiesto; la nuova frontiera del lavoro, almeno per una parte non marginale, richiede "lavoratori" in grado di "spiegare" ai robot o più in generale a macchine e sistemi, come si devono comportare.
Il problema che si pone è come insegnare a "insegnare" alle macchine ed agli altri umani secondo modalità del tutto nuove e non ancora ben definite ed esplorate e, più in generale, a diventare parte attiva dei nuovi processi di insegnamento e produttivi e non solo "spettatori" o meri "clienti" dei processi stessi.
In pratica, gli obiettivi del "nuovo" modo di insegnare e di affrontare le esigenze dei nuovi scenari di lavoro si possono riassumere nei seguenti:

- riconoscere la "natura" dei problemi e scoprire diversi modi di trovare soluzioni che risolvano i problemi stessi;
- ragionare in modo astratto e "quantitativo" per poi trasformare ciò in processi concreti mediante l'individuazione di "variabili" che rappresentino le grandezze trattare ed algoritmi che ne rappresentino le relazioni (in breve: adottare la simulazione matematica alla risoluzione dei problemi);
- realizzare modelli che riproducano il comportamento di entità reali;
- prestare attenzione a tutti i dettagli di un problema implementandoli accuratamente negli algoritmi;
- adottare approcci strutturati alla progettazione e realizzazione delle soluzioni individuando i moduli ripetitivi, scambiando esperienze e codice con le comunità di ricerca e sviluppo, costruendo progetti sempre più complessi utilizzando componenti modulari.

## PERCHÈ SCRATCH

Questa lunga premessa serve a spiegare perché abbiamo deciso di proporre come "campo" di studio e di divertimento per apprendere l'arte della programmazione l'ambiente di sviluppo "Scratch".
Scratch è il risultato di lunghi anni di ricerche e di "confronti" filosofico-tecnologici sul tema dell'insegnamento della tecnologia informatica al più alto numero di persone possibile.
Scratch nasce nel Lifelong Kindergarten research group dei Media Lab dell'MIT, sulla base delle esperienze del Prof. Seymour Papert, uno dei fondatori del LAB stesso e realizzatore negli anni '60, del linguaggio LOGO, oltre che collaboratore della LEGO nello sviluppo dei LEGO Mindstorm.
Scratch è un linguaggio di programmazione didattico ed insieme al suo ambiente di sviluppo, ispirato alla teoria costruttivista dell'apprendimento, è stato progettato per l'insegnamento della programmazione tramite primitive visive. Scratch e il suo ambiente di sviluppo sono adatti a studenti, insegnanti e genitori, e possono essere utilizzati per progetti pedagogici e di intrattenimento che spaziano dalla matematica alla scienza, consentendo la realizzazione di simulazioni, la visualizzazione di esperimenti, l'esecuzione di animazioni e musica, l'arte interattiva, semplici giochi e, nel nostro caso, anche di interfacciarsi con sensori e circuiti elettronici.

## CARATTERISTICHE DEL LINGUAGGIO SCRATCH

Al di là del suo aspetto "infantile" Scratch è un ambiente di sviluppo di tutto rispetto. Prevede un approccio, in un certo senso, orientato agli oggetti, dove gli "oggetti" sono fisicamente rappresentati da "spiritelli" "sprite", dotati di struttura (forme) metodi e costumi.
Presenta la possibilità di gestire le strutture logiche fondamentali, la programmazione ad eventi e la gestione di "messaging" tra diversi processi in esecuzione contemporanea. Scratch è un linguaggio di programmazione che consente di elaborare storie interattive, giochi, animazioni, arte e musica. Inoltre permette di condividere i progetti con altri utenti del web.
L'idea ispiratrice di questo linguaggio è che anche i bambini o le persone inesperte di linguaggi di programmazione possano arrivare ad imparare importanti concetti di calcolo matematico, ragionare in modo sistematico, pensare in modo creativo e anche lavorare in gruppi partecipativi.
Scratch è caratterizzato da una programmazione "grafica" con blocchi di costruzione (blocchi grafici) creati per adattarsi l'un l'altro, ma solo se inseriti in una corretta successione; questa importante caratteristica evita errori nella sintassi.

*Ambiente*

L'ambiente Scratch è disponibile sia via web, sul sito ufficiale, sia come software installabile ed utilizzabile offline, disponibile per le piattaforme Windows, Mac OS X e GNU/Linux. Su Raspberry Pi ed altri microPC è incluso nella distribuzione ufficiale. L'ultima versione stabile è la 1.4.

*Comunità*

Lo slogan della comunità online di Scratch, che recita "Immagina, Programma, Condividi", sottolinea l'importanza della condivi-

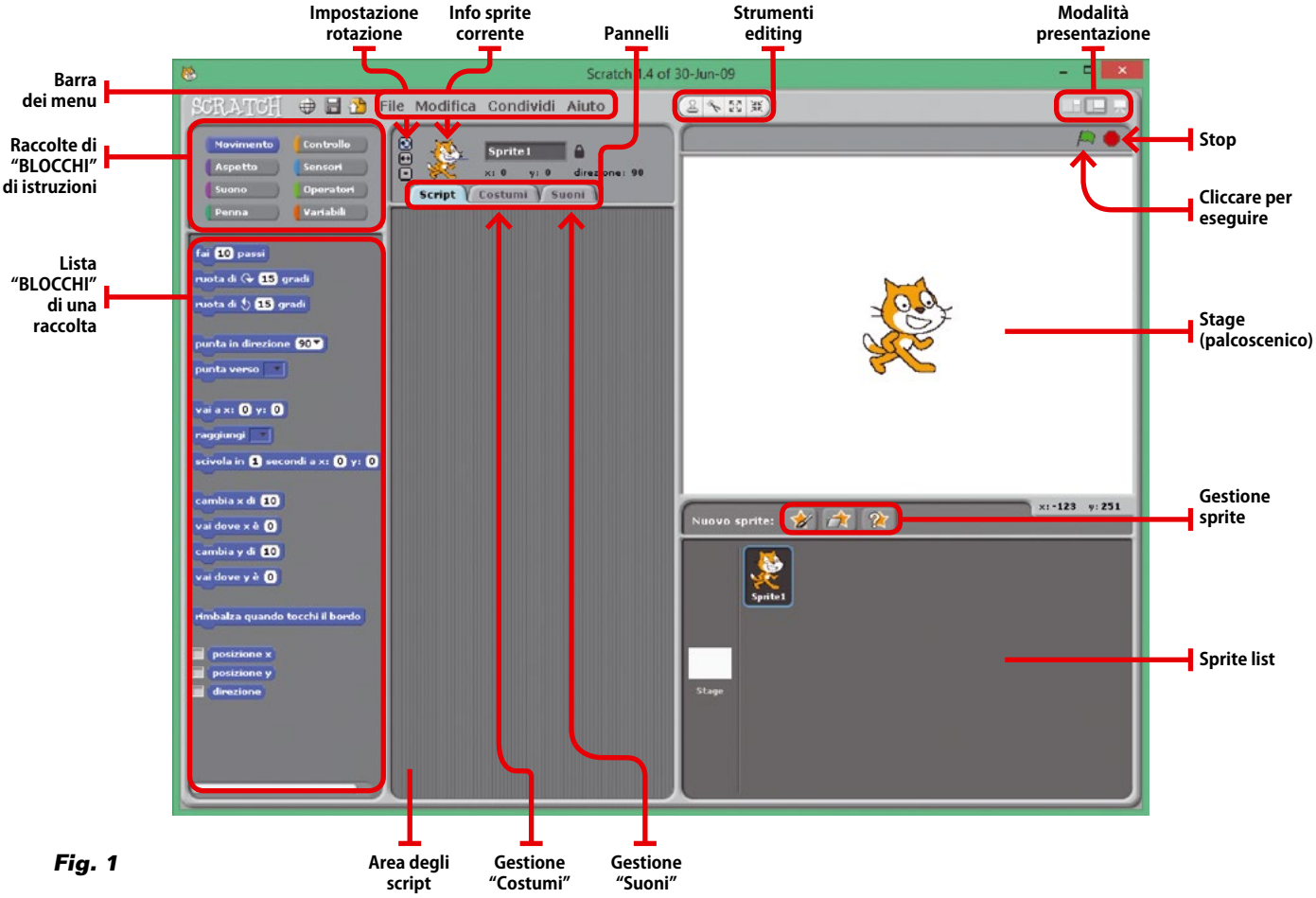



***Fig. 1***

sione e degli aspetti sociali della creatività nella filosofia alla base di Scratch.
I progetti Scratch, essendo a codice aperto, sono liberamente modificabili e utilizzabili per crearne di nuovi.
I progetti possono essere inviati direttamente dal programma al sito web di Scratch, e qualsiasi membro della comunità può scaricarne il codice per studiarlo o modificarlo in un nuovo progetto. I membri possono inoltre creare gallerie di progetti, commentare, "taggare" e aggiungere progetti ai "preferiti". Tutti i progetti sul sito sono condivisi con licenza Creative commons "Share-Alike" e riprodotti su un browser (utilizzando Flash Player).
Il sito web riceve quasi 10 milioni di visite al mese e i membri registrati sono oltre 1.400.000 (di cui oltre 400.000 hanno condiviso progetti), per un totale di più di 4.200.000 progetti condivisi (più di un progetto inviato al minuto).
Il sito web organizza periodicamente lo "Scratch Design Studio", competizione per incoraggiare la creazione e condivisione di progetti dalla grafica elementare.
Nel 2008, la piattaforma della comunità online di Scratch (denominata "ScrachR") ha ricevuto una menzione onoraria all'Ars Electronica Prix.
Esiste anche una comunità online per educatori, denominata ScratchEd.

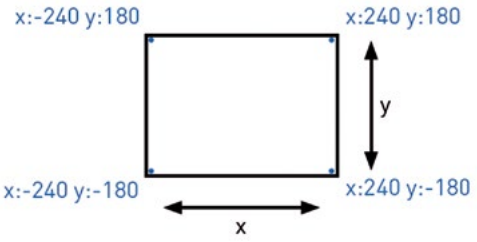



***Fig. 2***

## DOWNLOAD E INSTALLAZIONE DI SCRATCH

Per quanto riguarda il download e l'installazione, niente di più semplice: nell'ambito Windows basta scaricare il programma dall'indirizzo *http://scratch.mit.edu/scratch_1.4/* ed eseguire il file **.exe** una volta terminato il download. Per quanto riguarda l'installazione nel sistema operativo Raspbian (Raspberry Pi) si utilizzano i soliti comandi:

```
apt-get update
apt-get install scratch
```

Questi valgono ovviamente anche per installare Scratch sulle distribuzioni Debian del sistema operativo GNU/Linux per altri microPC. Comunque nelle distribuzioni recenti, in genere, Scratch è già installato.
In tutti i casi, per eseguire Scratch dal desktop basta cliccare

sull'icona del collegamento che riproduce il muso di un gattino (la mascotte di Scratch).

## GLI "INGREDIENTI" DI UN PROGETTO SCRATCH

In genere i progetti Scratch sono realizzati attorno ad oggetti grafici chiamati "Sprite"; ciascun oggetto può assumere diversi aspetti rappresentati da differenti "Costume" (costumi). Nel caso elettronico potremmo avere uno sprite "LED" con due costumi: "Acceso" e "Spento". Con gli sprite si può rappresentare pressoché qualunque entità: persone, mezzi di trasporto, componenti elettronici, animali o qualsiasi altra cosa. Come "costumi" si possono usare immagini di qualsiasi tipo, create con un editor di immagini, importate da un disco fisso, scaricate da un sito web o riprese da una fotocamera. Gli "sprite" possono essere "istruiti" ad eseguire una quantità di diverse azioni: si possono muovere, cambiare di aspetto, possono emettere suoni e musica o interagire con altri sprite. I "comandi" che guidano il comportamento degli sprite sono organizzati in veri e propri programmi, realizzati componendo moduli grafici che rappresentano le diverse tipologie di "istruzioni" e che hanno forme tali che possono essere "incastrati" tra di loro in modo da rispettare una corretta logica di programmazione. I programmi costruiti con i blocchi grafici sono definiti "script". Per eseguire uno script basta cliccarci sopra due volte con il tasto sinistro del mouse.
In seguito vedremo altre modalità di esecuzione degli script.

## L'INTERFACCIA PER L'UTILIZZATORE

Nella **Fig. 1** è rappresentata l'interfaccia di sviluppo dei progetti "Scratch", un vero e proprio IDE di sviluppo grafico. Nella figura sono indicati le principali finestre e i comandi essenziali per interagire con l'ambiente di sviluppo.

*Stage (Palcoscenico)*
Partiamo dallo *stage*, che è la finestra di lavoro principale: il "palcoscenico" della nostra applicazione. In questa finestra viene realizzata l'interfaccia utente del progetto dove si inseriscono gli sprite, le animazioni e i comandi del progetto. Il posizionamento degli oggetti nello stage avviene con riferimento alla griglia grafica che caratterizza lo stage stesso: la relativa finestra è composta da 480 "unità" di larghezza e "360" "unità" di altezza; ciascuna locazione è individuata nella finestra con riferimento agli assi x, y secondo lo schema visibile in **Fig. 2**. Il centro della finestra è caratterizzato dalle coordinate 0,0.
Per individuare visivamente le coordinate di una data locazione, basta posizionare il puntatore del mouse sulla locazione desiderata e leggere le coordinate nell'indicatore in basso a destra

*Fig. 3*

della finestra. Cliccando sul pulsante "Passa alla modalità presentazione" visibile in **Fig. 3**, è possibile vedere la finestra di stage a schermo intero.
Per uscire dalla presentazione a schermo intero basta utilizzare il tasto "Esc".
Gli altri due tasti a fianco permettono di allargare o restringere l'area dello stage.

*Gestione degli sprite nello stage*
Quando si apre un nuovo progetto, la finestra di stage si presenta con all'interno lo sprite del gatto. Per gestire gli sprite si utilizzano i pulsanti (**Fig. 4** e **Fig. 5**):
- **Disegna il tuo sprite**; apre l'editor grafico in stile "Paint" con il quale è possibile dare vita ai propri "sprite".
- **Scegli un costume per uno**

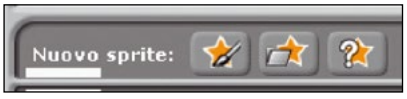


*Fig. 4*

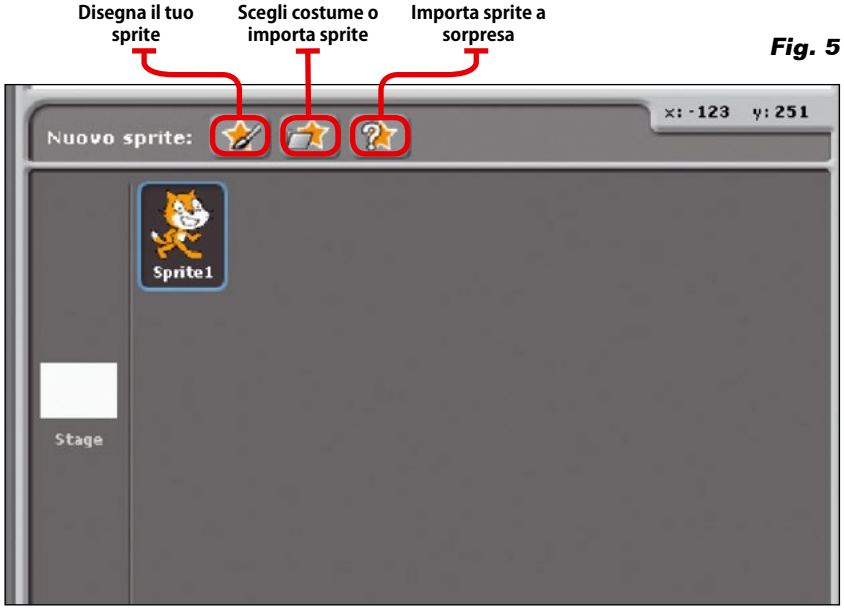


*Fig. 5*

**sprite o importa uno sprite completo**; permette di scegliere ulteriori costumi per uno sprite già incluso nel progetto, oppure permette di importare uno sprite con i suoi costumi associati, in genere "riutilizzato" da un progetto esistente;

- **Importa uno sprite a "sorpresa"**; aggiunge al progetto uno sprite a "caso" tra quelli presenti nella galleria degli sprite; se si desidera cancellare uno sprite, basta selezionare la funzione "forbici" dalla barra degli strumenti e cliccare sullo "sprite" da eliminare (un altro modo di eliminare uno sprite è cliccarvi sopra con il tasto destro del mouse e selezionare "delete" dal menu a tendina).

*Galleria degli sprite*

Nell' IDE di Scratch esiste una nutrita galleria di sprite preconfezionati, completi di costumi ed effetti sonori.
La funzione "Sprite List" mostra le anteprime di tutti gli sprite disponibili. Per ciascuno sprite vengono indicati il nome, da quanti script è utilizzato e quanti costumi sono disponibili.
Per visualizzare e modificare gli script, i costumi o gli effetti sonori associati ad uno sprite; se siete nella Sprite List cliccate sull'anteprima dello sprite, oppure, nella finestra di stage, cliccate due volte direttamente sullo sprite. Lo sprite selezionato verrà evidenziato con un contorno blu.
Nella finestra "Sprite List" è possibile organizzare gli sprite a piacere, semplicemente cliccandovi sopra e trascinandoli nelle posizioni desiderate.
È possibile personalizzare l'aspetto di uno sprite assegnando costumi differenti, ma anche modificare gli sfondi dell'area di stage.

***Fig. 6***


***Fig. 7***


***Fig. 8***



Per visualizzare e modificare gli script, gli sfondi e gli effetti sonori associati allo stage cliccate sull'icona "Stage" sulla sinistra della finestra "Sprite List".

*Palette blocchi di istruzioni e area degli script*

Per animare uno sprite si deve creare un programma (script) trascinando i blocchi grafici corrispondenti alle istruzioni richieste, dalla Blocks Palette nella Scripts Area.
Per eseguire un blocco di istruzioni basta farvi doppio clic. Per creare un programma bisogna impilare i blocchi grafici uno sull'altro, rispettando la loro "forma" e la possibilità che si "incastrino" correttamente tra di loro; in questo modo si ha un controllo preliminare sul rispetto della "sintassi" del linguaggio. In altre parole il programma è corretto dal punto di vista della sequenza e coerenza logica delle istruzioni. Non necessariamente, il programma eseguirà quello che noi desideriamo. Per questo dovremo eseguire i cicli di debug fino a quando otterremo i risultati voluti. Facendo doppio clic su un blocco di istruzioni si avvia l'esecuzione delle istruzioni stesse, dall'alto verso il basso, rispettando le iterazioni e le istruzioni di controllo presenti. Per approfondire le funzionalità di un blocco basta cliccarvi sopra col tasto destro del mouse e selezionare la voce "Aiuto" dal menu a tendina. Quando si trascina un blocco all'interno della "Script Area" una linea bianca luminosa indica se un blocco può essere o no inserito nella posizione in cui si trova per costituire una connessione valida con i blocchi circostanti; la forma dei blocchi aiuta a capire se possono essere incastrati tra loro oppure se sono incompatibili.
Per spostare uno script bisogna "afferrarne" il centro del blocco superiore; se si "stacca" un blocco che si trova all'interno di una sequenza di istruzioni, verranno

staccati anche tutti i blocchi sottostanti. Alcuni blocchi presentano dei campi bianchi al loro interno (**Fig. 6**) che possono essere utilizzati per introdurre quelle che nei linguaggi di programmazione tradizionali sono definite "costanti" e "variabili". Per modificare il valore della costante nel blocco bisogna cliccare sul campo bianco col tasto destro del mouse ed inserire un nuovo valore. Per utilizzare il campo come una variabile, occorre trascinare un blocco di forma simile a quello in **Fig. 7** nel campo bianco.
È possibile utilizzare "variabili" predefinite, oppure crearne di nuove. Vedremo in seguito come creare e utilizzare le variabili che ci necessitano per i nostri progetti.
Alcuni blocchi presentano dei menu a tendina come quello in **Fig. 8**. Basta cliccare sul menu per aprire la tendina e cliccare una seconda volta sulla selezione desiderata.
La pagina che si apre cliccando sulla linguetta "Costumi" permette di vedere e modificare i costumi degli sprite. Nella **Fig. 9** sono visibili i due costumi dello sprite "gatto"; come si vede, quello corrente appare evidenziato. Per impostare per lo sprite un costume differente da quello attuale, è sufficiente cliccare sul costume desiderato. Per creare nuovi costumi si può ricorrere a tre modalità:

- cliccare sul pulsante "Disegna" per disegnare un nuovo costume nell' "Editor di immagini" (**Fig. 10**);
- cliccare sul pulsante "Importa" per importare un'immagine da un file (**Fig. 11**);
- cliccare sul pulsante "Webcam" per importare un'immagine da webcam (**Fig. 12**).

Per quel che riguarda l'importazione, l'ambiente di sviluppo Scratch riconosce diversi formati di file d'immagine: JPG, GIF, BMP e PNG.
I costumi possono essere disposti a piacere nel pannello trascinando e riposizionando le relative immagini. Cliccando su un costume con il tasto destro del mouse, si può trasformarlo in un nuovo sprite oppure esportarlo in un file separato. Cliccando sulla linguetta "Suoni" si apre la pagina di gestione degli effetti sonori. Cliccando sui rispettivi pulsanti, si possono registrare suoni in diretta mediante i dispositivi audio in dotazione al sistema, oppure importarli da fonti esterne (**Fig. 13**); a tal proposito va detto che scratch accetta file codificati secondo gli standard MP3, WAV (non compresso), AIF ed AU campionati ad 8 o 16 bit (non è accettata la codifica a 24 bit).
In **Fig. 14** è raffigurata la porzione dell'ambiente di sviluppo che riassume le informazioni principali sullo sprite: il nome, la posizione (espressa in coordinate x e y) e la direzione. In questo pannello è possibile modificare il nome dello sprite digitandolo nell'apposito campo.
Il campo "direzione" indica in quale direzione si muoverà lo sprite a fronte dell'esecuzione di un blocco "Muovi": in particolare, il valore "0" indica un movimento verso l'alto, 90 verso destra, 180 in basso e -90 verso sinistra. La stessa informazione è evidenziata graficamente dall'orientamento della linea blu sull'immagine dello sprite.
È possibile afferrare e trascinare la linea blu per modificare la direzione di spostamento dello sprite. Cliccando sull'immagine dello sprite si ripristina la direzione allo stato originale (direzione=90).

*Fig. 9*

*Fig. 10*

*Fig. 11*

*Fig. 12*

*Fig. 13*

*Fig. 14*

Per verificare come si comporta lo sprite durante i movimenti di rotazione utilizzate i pulsanti Rotation style:

- **Può ruotare** = il costume ruota in modo coerente con il cambiamento di direzione dello sprite;
- **Voltati solo a destra-sinistra** = il costume si gira verso destra o sinistra a seconda della direzione;


*Fig. 15*

- **Non ruotare** = il costume non ruota durante i cambiamenti di direzione.

Per salvare lo sprite come un file separato, in modo da poterlo utilizzare in un altro progetto, cliccate sulla figura dello sprite nella "Sprite list" e selezionate l'opzione "Esporta" dal menu a tendina (**Fig. 15**).
La barra degli strumenti permette di eseguire le operazioni di editing sui diversi oggetti (**Fig. 16**):

- **Freccia** = modalità di utilizzo normale, afferra e sposta sprite, blocchi e oggetti in genere, si attiva cliccando fuori dalla barra degli strumenti;
- **Duplica** = esegue una copia dello sprite (selezionare duplica e, con il mouse a forma di timbro, cliccare sullo sprite, nella "sprite list", che si vuole duplicare);
- **Cancella** = elimina uno sprite dalla "sprite list";
- **Espandi sprite** = aumenta le dimensioni dello sprite;
- **Riduci sprite** = diminuisce le dimensioni dello sprite.

In **Fig. 17** vediamo le diverse opzioni della barra del menu generale dell'ambiente di sviluppo, che permette la gestione dei progetti e la configurazione e personalizzazione dell'ambiente stesso, e la gestione del debug passo passo del programma.
Sono presenti anche le opzioni di esportazione e condivisione del progetto.
Il pulsante con la bandierina verde fornisce un modo comodo per mettere in esecuzione diversi script contemporaneamente. Infatti il pulsante con la bandiera verde mette in esecuzione tutti gli script che hanno come "cappello" il blocco visibile in **Fig. 18**.
Nel modo "Presentation" la bandierina verde appare come un'icona nell'angolo in alto a destra dello schermo. Premere il tasto "Enter" ha lo stesso effetto di cliccare sulla bandierina; in questo modo si possono realizzare progetti composti da molti task che lavorano in concorrenza, in grado di gestire molteplici eventi e processi contemporanei.
Già da questa possibilità, possiamo comprendere che siamo di fronte ad un ambiente di sviluppo didattico, orientato a imparare a programmare in modo grafico ed interattivo, e non certo ad un "giocattolo".
Con questo ambiente possiamo realizzare videogiochi interattivi, ed anche applicazioni in grado di interfacciare sensori e moduli di ingresso e uscita elettronici.
Passiamo a descrivere i diversi tipi di "mattoni" o blocchi che utilizzeremo per "costruire" i nostri programmi e progetti.
Semplificando molto, nelle raccolte di blocchi (Block Palette) vi sono i quattro seguenti tipi principali di "mattoni".
- *Controllo*: questi blocchi permettono di realizzare la logica principale dei programmi e riproducono graficamente le tre principali strutture logiche della programmazione: sequenza, alternanza (if, then else oppure case) e iterazione (for, while). I blocchi hanno degli incavi nella parte superiore e delle protuberanze nella parte inferiore che fungono da "chiavi" per il loro corretto "impilaggio". I blocchi, come abbiamo già visto, hanno al loro interno delle ulteriori "finestre" di forme diverse dove vengono inseriti gli "argomenti" delle diverse "istruzioni", definiti in Scratch come "reporters". In alternativa, alcune istruzioni mostrano un menu a tendina che offre una serie di scelte

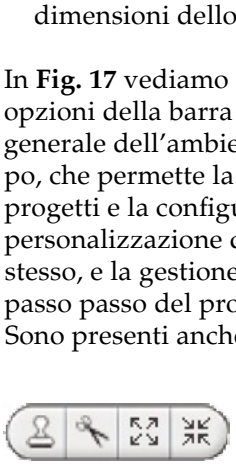


*Fig. 16*

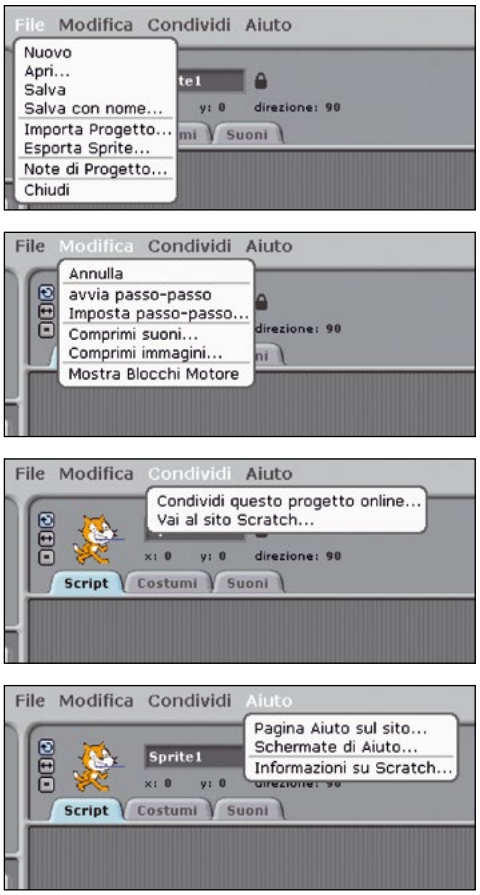



*Fig. 17*


*Fig. 18*


**Fig. 19**


**Fig. 20**


**Fig. 21**


**Fig. 22**

preconfezionate per "personalizzare" il comportamento dell'istruzione stessa. I blocchi di "controllo" della sequenza di elaborazione hanno forme a "C", come le istruzioni di controllo delle iterazioni ed a "E" come nelle istruzioni di controllo delle alternanze soggette a condizioni che possono verificarsi o meno. Nel caso vero, viene eseguita la sequenza di istruzioni contenute nella "bocca" superiore della "E", in caso contrario il blocco di istruzioni contenute nella "bocca" inferiore. Anche gli interni delle "bocche" sono conformati graficamente in modo da poter accogliere solo blocchi di forma che combaciano con la forma dell'alloggiamento (**Fig. 19**).

- *Cappelli (Hats)*: questi blocchi (**Fig. 20**) hanno la parte superiore arrotondata e sono utilizzati per indicare l'inizio di ciascun "blocco" di istruzioni. Nella stessa area di script è possibile inserire più blocchi, ciascuno con il proprio "cappello" che resta in attesa dell'evento in base al quale dare inizio alla sequenza di istruzioni sottostante. Si va dal semplice clic sulla bandierina verde all'attesa della pressione di un tasto sulla tastiera, all'azione su uno sprite, ad un messaggio da un altro blocco o ad un evento legato ad un sensore esterno, come vedremo nelle prossime puntate.
Grazie a questi blocchi, utilizzando Scratch si possono realizzare applicazioni "basate su eventi", come si suole dire e strutturare i programmi in modo molto simile a quella che viene definita "programmazione orientata agli oggetti". Ottenere lo stesso risultato didattico, utilizzando linguaggi tradizionali, richiederebbe uno sforzo di logica ed astrazione di dimensioni tali che rappresenta la principale causa di "abbandono" da parte di chi affronta per la prima volta la programmazione secondo il metodo "tradizionale".
- *Comandi (Istruzioni nei linguaggi classici)*; rappresentano le singole azioni che devono essere eseguite per completare una funzione di senso compiuto (**Fig. 21**).
- *Reporters*: questi blocchi, dalle forme simili a quelle visibili in **Fig. 22** sono predisposti per "incastrarsi" negli appositi spazi all'interno di altri blocchi. Ovviamente questi blocchi possono essere incastrati unicamente in aree della stessa forma. I blocchi di forma arrotondata "riportano" il valore delle variabili espresse dal nome del blocco stesso mentre i blocchi con le estremità appuntite riportano la condizione di "vero" o "falso" dell'espressione contenuta al loro interno.
Alcuni blocchi di questa "famiglia" hanno un campo di spunta (check box) affiancato alla forma grafica che li rappresentano. Cliccando sul campo di spunta (attivandolo), si fa apparire nell'area di stage un elemento di visualizzazione che evidenzia di volta in volta il valore assunto dalla variabile. Il campo di visualizzazione può assumere tre formati differenti con i significati visibili nella **Fig. 23**.
Per passare da un formato all'altro del campo cliccate due volte sul campo stesso oppure una sola volta col tasto destro del mouse. Cliccando col tasto destro del mouse su un indicatore a slitta è possibile modificare i valori di minimo e massimo.

## UN PRIMO ESEMPIO

Prima di concludere questa prima puntata, presentiamo un semplice esempio per iniziare a comprendere la "logica" sottostante l'ambiente di sviluppo ed il linguaggio Scratch. Per questa volta non rendiamo disponibile il listato sul sito di Elettronica In. Il progetto in questione consiste nel creare uno sprite a forma di "palla" da fare muovere all'interno dell'area di stage facendolo "rimbalzare" quando tocca uno dei bordi. Il tipo di rimbalzo è simile a quello che avviene su un tavolo da biliardo.
Per prima cosa ci serve una palla, non il gatto, quindi clicchiamo


*Una casella con il nome ed il valore della variabile*

*Una casella di dimensioni maggiori con il solo valore della variabile*

*Una casella con nome, valore ed un cursore che permette di modificare il valore della variabile, chiaramente utilizzabile per le sole variabili in input*

**Fig. 23**

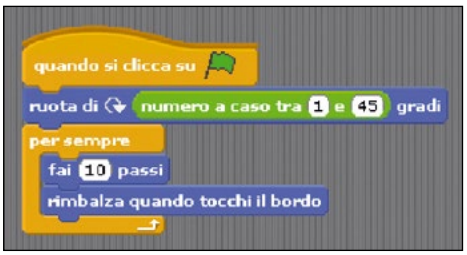



***Fig. 24***

col tasto destro del mouse sullo sprite del gatto nella "Sprite list" e selezioniamo "Cancella". Ora selezioniamo "Seleziona nuovo sprite da file" e dalla cartella "Things" scegliamo una "Palla". Rinominiamo il nostro sprite in "Palla".

Ora creiamo uno script per fare rimbalzare la palla all'interno dell'area di stage. Trasciniamo nell'area degli script un "Cappello" del tipo, "Quando si clicca su bandierina verde". Per fare in modo che, ogni qualvolta eseguiamo il programma, la palla esegua un percorso sempre diverso, impostiamo un angolo di rotazione differente per ogni volta.

Trasciniamo un blocco blu con l'istruzione "ruota di … gradi" e nel campo bianco della variabile trasciniamo un blocco "numero a caso tra ..." e fissiamo gli estremi dell'intervallo tra 1 e 45. "Numero a caso" (Random) è una funzione presente in pressoché tutti i sistemi di calcolo per generare un numero casuale in assoluto o all'interno di un intervallo, se quest'ultimo è indicato, come nel nostro caso.

Inseriamo sotto al blocco blu un blocco che racchiuda le istruzioni che devono essere eseguite in continuazione, ovvero quelle che costituiscono il corpo principale del programma. Trasciniamo il blocco a "C" "per sempre". Al suo interno inseriamo un blocco di avanzamento "fai <x> passi" ed indichiamo in "10" i passi di avanzamento ogni volta che viene eseguito il blocco. Lo trasciniamo all'interno della "bocca" della "C".

Ora dobbiamo fare in modo che quando la "palla" tocca un bordo, rimbalzi tornando all'interno dell'area di stage. Abbiamo un blocco che fa al caso nostro "rimbalza quando tocchi il bordo"

Potete eseguire il "programma" cliccando sulla bandierina verde in alto a destra (**Fig. 25**).

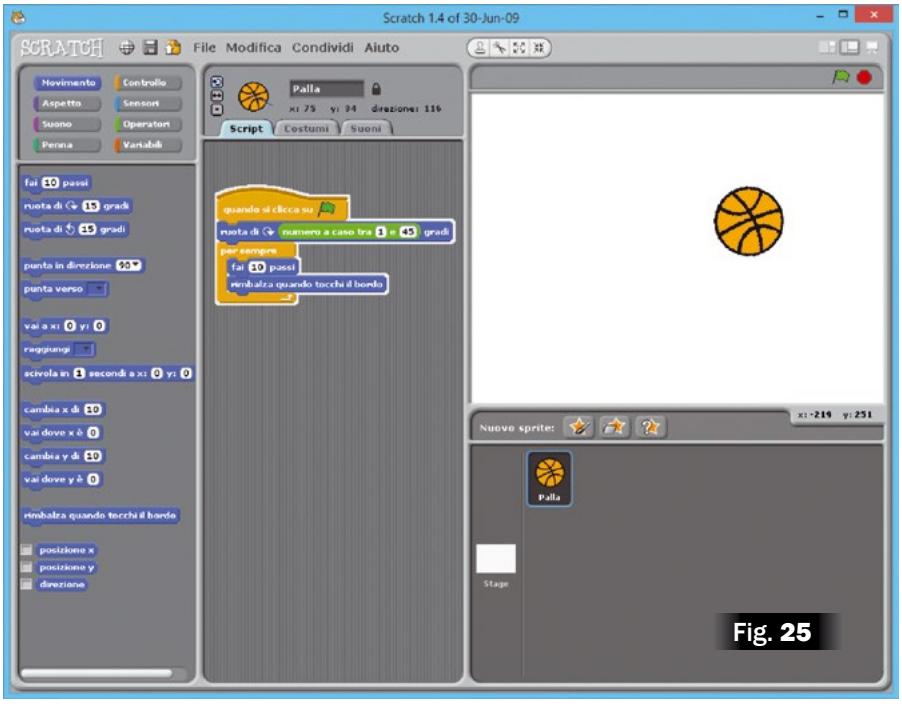



**Fig. 25**


***Fig. 26***


***Fig. 27***

Carino, no?

Con una manciata di blocchi abbiamo scritto un programma da far venire il mal di testa se avessimo utilizzato un linguaggio "tradizionale".

In realtà in questi pochi blocchi sono presenti due delle "strutture" di un linguaggio di programmazione (sequenza e iterazione) oltre ad una prima funzione basata su eventi. Divertitevi a trovare funzionalità da aggiungere al giochino e scoprire come fare. Per eliminare un'istruzione dallo script bisogna prima "staccarla" dallo script, trascinandola in una zona libera dell'area degli script, poi cliccarvi sopra col tasto destro del mouse e selezionare "Cancella".

Un piccolo suggerimento: aggiungete nell'area degli script il modulo visibile in **Fig. 26**; così facendo, ogni volta che riuscite a cliccare sulla palla in movimento, questa cambierà "colore". Non vedete cambiare colore? Rallentate il movimento della palla come visibile in **Fig. 27** e riprovate, cliccando il centro della palla.

Già avete implementato un micro video gioco. E questo è solo l'inizio.

Per ora ci lasciamo, in attesa di tornare a parlare presto di Scratch e delle sue magnifiche possibilità di utilizzo.


# SCRATCH: LA PROGRAMMAZIONE RESA FACILE

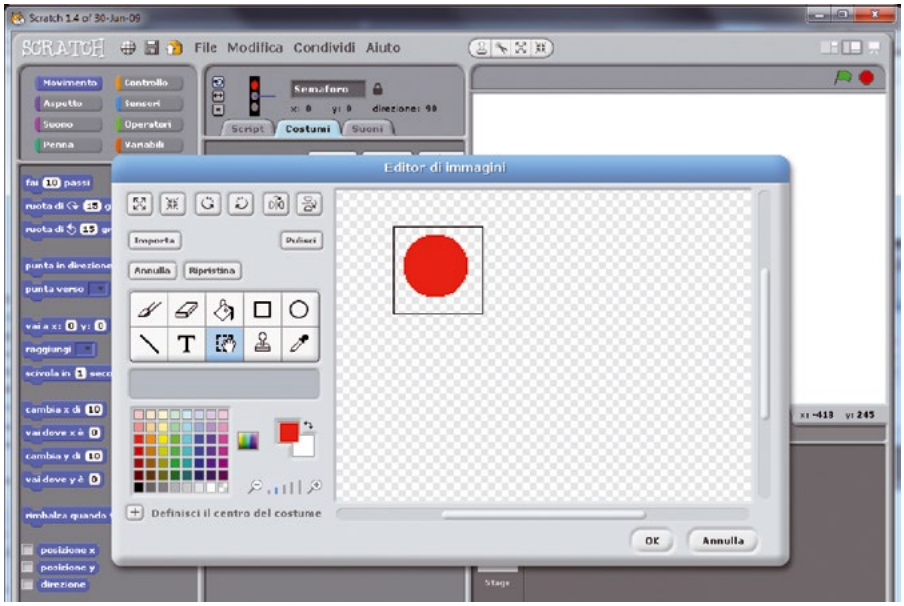
di MARCO MAGAGNIN

**Dopo aver preso dimestichezza con l'ambiente di sviluppo che accompagna il linguaggio Scratch approfondiamo le principali strutture logiche che stanno alla base dell'attività di programmazione dei calcolatori elettronici. Un paio di simpatici esempi renderanno tutto più semplice. Seconda parte.**

Riprendiamo il nostro cammino nell'apprendimento dell'arte della programmazione dei calcolatori elettronici, approfondendo le strutture di base della programmazione in generale ed in particolare di come sono state implementate nel linguaggio Scratch. Non ci stancheremo mai di ricordare che la progettazione e realizzazione di programmi e, da un punto di vista più ampio, di applicazioni, richiede sicuramente la padronanza delle tecniche e delle pratiche proprie della disciplina ma anche, se non soprattutto, la capacità di saper "progettare" gli elementi di base in una "architettura" ben strutturata, dal funzionamento affidabile, con prestazioni adeguate e, non ultima, bella da vedere e facile ed intuitiva da utilizzare. Questa capacità è ciò che di solito usiamo chiamare creatività, estro e perché no, genio. Non a caso il


**Fig. 1**

**Fig. 2**




corso di "programmazione" più prestigioso al mondo, tenuto dal Prof. Donald Knuth, all'università di Stanford si intitola "The art of computer programming". I temi del suo corso sono raccolti in un'opera di quattro volumi monumentali dal titolo omonimo. Se riuscite a studiarli tutti un posto in Google non ve lo leva nessuno. Non meravigliatevi se sottolineo in continuazione la componente artistica associata alle attività di programmazione, così come avviene in molte altre discipline umane, come la musica, la pittura e la scultura, per citarne solo alcune. Prendiamo il solo esempio della pittura, supporti come tele, tavole e intonaci così come pennelli e colori sono gli "ingredienti" di base, che richiedono, per essere realizzati, conoscenze tecniche di chimica, muratura, tessitura, conoscenza dei materiali, ecc. Le "strutture" disciplinari principali sono costituite dalla conoscenza degli effetti di luce, delle ombre, della prospettiva, delle tecniche di miscelazione e deposizione dei colori, ecc. Tutte queste conoscenze, tuttavia, non danno grandi garanzie sulla possibilità di realizzare un bel quadro, e nemmeno esiste un manuale che possa guidare un artista passo passo a realizzare un'opera d'arte, o comunque qualcosa di giudicabile come "bello". Potremmo continuare con un numero infinito di esempi, ma torniamo ad occuparci degli ingredienti di base con i quali possiamo "comporre" i nostri programmi e le nostre applicazioni. L'essenza della programmazione, come abbiamo già anticipato nella prima puntata di questa serie di articoli, consiste nello spiegare a qualcosa che assomiglia ad un computer, cosa deve fare per ottenere ciò che abbiamo in mente. E questo richiede di aver studiato accuratamente il problema che dobbiamo risolvere, aver identificato una soluzione, verificata in tutti i suoi dettagli e condizioni operative e poi espressa in una forma di scrittura comprensibile al "computer" o qualsivoglia altra diavoleria che si incaricherà, come un fedele maggiordomo, di eseguire "scrupolosamente" le nostre disposizioni, fosse anche accendere il barbeque di domenica alle 11 dentro il garage di fianco all'automobile. Con mia figlia, quando era piccola, facevamo un gioco che chiamavamo "Ordine a Robot". Il robot ero io ed eseguivo "scrupolosamente" le istruzioni che mi venivano impartite, con risultati esilaranti, tipo mettere la pasta nell'acqua senza toglierla dal sacchetto. Semplificando al massimo, spiegare ad un computer cosa deve fare, significa fornire a quest'ultimo una serie di operazioni da eseguire in sequenza, nel caso ripeterle per un numero di volte indefinito o che può dipendere da particolari condizioni che si possono verificare di volta in volta, e dare disposizioni precise in modo che possa prendere "decisioni" che modificano il proprio comportamento, in base allo stato generale del sistema o di sue parti, che si concretizzano nelle cosiddette "variabili". In una visione "moderna" della programmazione, possiamo istruire il nostro computer ad essere pronto a rilevare e rispondere a situazioni estemporanee, che vanno sotto il nome di "eventi". Nei diversi linguaggi di programmazione intendiamo per programma un testo che contiene le tipologie di istruzioni che abbiamo appena citato, nella forma adatta ad essere poi "tradotta" in un ulteriore linguaggio comprensibile al no-

*Fig. 3*

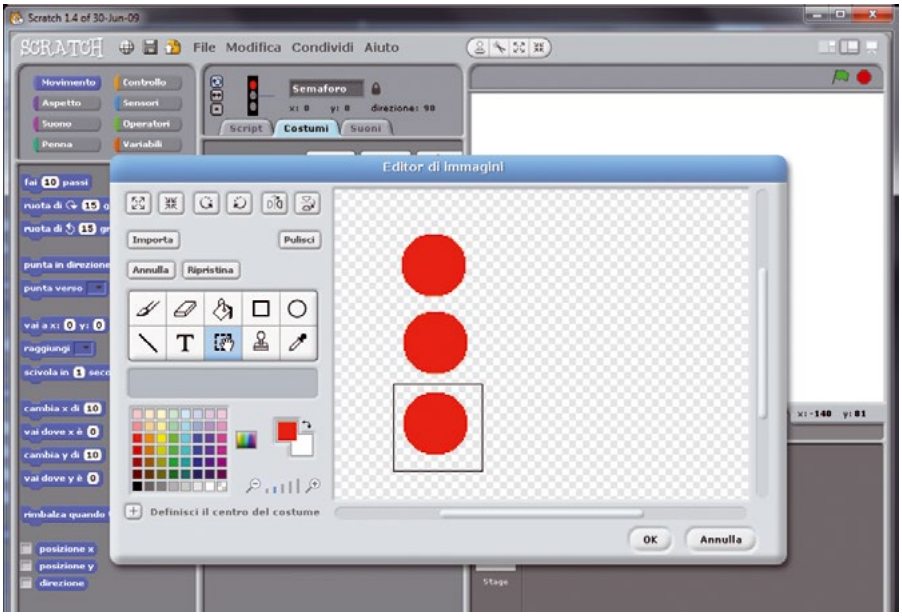
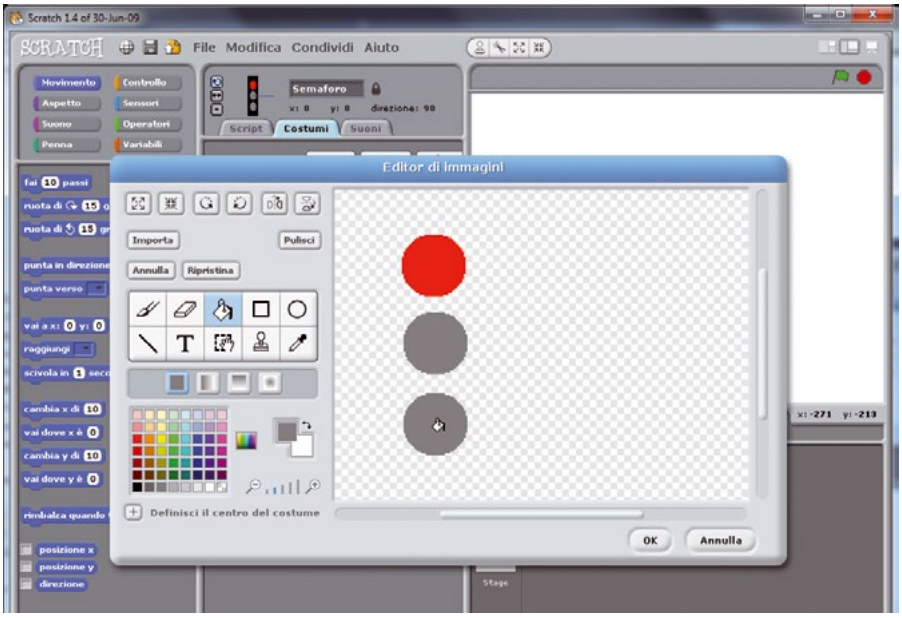



*Fig. 4*




stro computer, quello che viene genericamente definito come linguaggio macchina. Per poter essere tradotto senza ambiguità il testo deve essere redatto, come se si trattasse di un normale linguaggio umano, rispettando il vocabolario, la grammatica e la sintassi del linguaggio di programmazione adottato. Tipici linguaggi di programmazione comunemente adottati sono le varie "versioni" di C, il Python, il PHP, JAVA, HTML, Javascript e numerosissimi altri. Alcuni linguaggi adottano, in alternativa all'approccio testuale, un approccio grafico, dove le singole istruzioni sono caratterizzate da "simboli" grafici che ne mettono in evidenza le funzioni e, entro certi limiti, obbligano ad utilizzarle in modo corretto. Uno di questi è il linguaggio Scratch, che stiamo presentando in questi articoli, un altro esempio è il linguaggio di LabVIEW, oltre a molti altri. Un modo per rappresentare la "logica" di un programma in modo (quasi) indipendente dal linguaggio di sviluppo è il diagramma a blocchi, una serie di convenzioni grafiche che permettono di disegnare in modo schematico il flusso delle elaborazioni che devono essere eseguite dal codice che dobbiamo realizzare. In questa rivista avete avuto modo di vedere spesso diagrammi a blocchi a supporto della spiegazione della logica di funzionamento di programmi destinati a micro controllori come le schede della serie Arduino e gli integrati delle famiglie Microchip, ed a microcomputer come le diverse schede che ospitano il sistema operativo GNU/Linux.

## STRUTTURE DI CONTROLLO FONDAMENTALI

Quelle che descriviamo di seguito sono le strutture di controllo fondamentali così come codificate nel teorema di Böhm-Jacopini, enunciato nel 1966 dagli informatici Corrado Böhm e Giuseppe Jacopini nel quale si afferma che qualunque algoritmo può essere implementato in fase di programmazione utilizzando tre sole strutture di controllo: la sequenza (o concatenazione), il ciclo (iterazione) e la selezione. Con questa impostazione ha avuto inizio l'era della programmazione strutturata e dei relativi linguaggi di supporto. L'origine di questo approccio, storicamente, è stato sviluppato per risolvere le problematiche generate dall'utilizzo dei linguaggi di programmazione tradizionali, come COBOL, Fortran e vari dialetti Basic, che ammettono il "salto incondizionato". L'esagerazione nell'utilizzo di questo costrutto ha portato alla realizzazione di programmi e applicazioni di complessità tale da essere praticamente impossibili

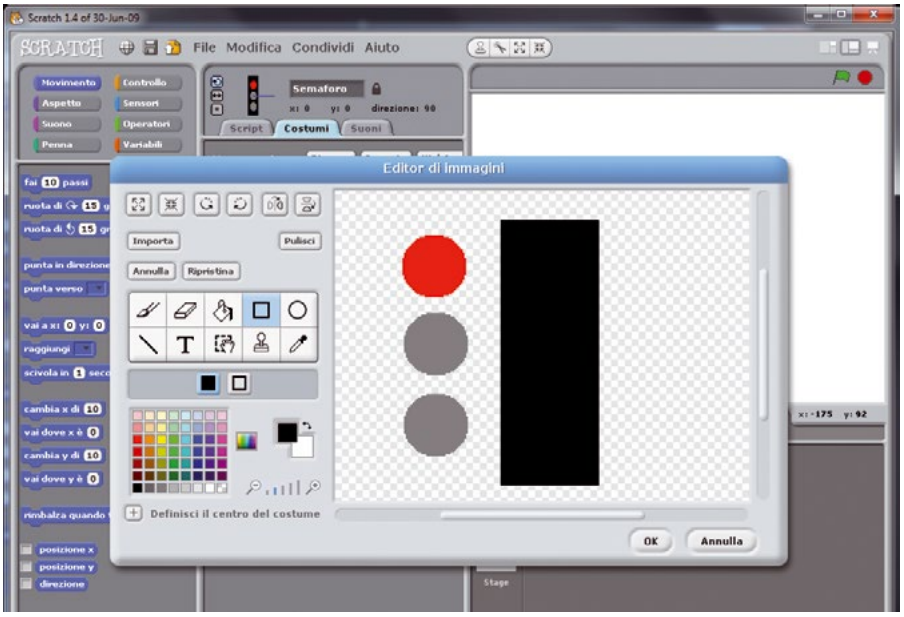



***Fig. 5***

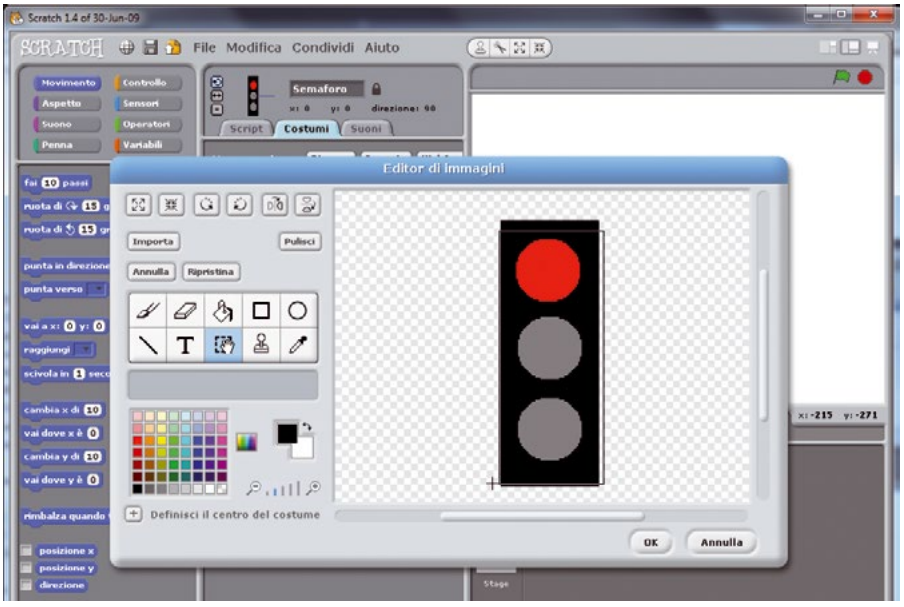



***Fig. 6***

da mantenere e da modificare per aggiungere nuove funzionalità. La loro complessità portò a definire questa categoria di programmi con la denominazione di "spaghetti code". La programmazione strutturata così come enunciata è sicuramente applicabile a tutti i tipi di programmi procedurali di tipo sincrono, cioè che non devono occuparsi di gestire eventi che possono accadere con modalità asincrona rispetto al ciclo logico stabilito. Un'estensione alla programmazione strutturata è data dall'introduzione del concetto di procedura (detto anche modulo o blocco), del concetto di oggetto (programmazione orientata agli oggetti) e dalla esigenza di gestire comunque alcuni eventi asincroni rispetto al flusso principale del programma come la necessità di "intrappolare" e gestire eventuali errori e/o situazioni anomale che possono presentarsi nel corso dell'elaborazione (on conditions). Una ulteriore evoluzione è poi data dalla necessità di gestire eventi asincroni rispetto al flusso principale del programma, necessità che ha dato origine al paradigma della programmazione orientata agli eventi, un paradigma comunque fortemente strutturato (event driven programmaing). Ci avventureremo nel seguito in queste ultime estensioni. Per ora approfondiamo le strutture di controllo proprie della programmazione strutturata: sequenza (o concatenazione), ciclo (iterazione) e selezione.

## SEQUENZA (O CONCATENAZIONE)

Per sequenza si intende un blocco di istruzioni che vengono eseguite una dopo l'altra in un ordine prestabilito. Per fare un esempio, prendiamo le istruzioni necessarie ad eseguire la sequenza di funzionamento di un semaforo. Partiamo dalla configurazione iniziale, che vede il nostro semaforo con la luce verde accesa e le altre due spente. Introduciamo un'istruzione di attesa che tenga ferma la configurazione del semaforo per un certo tempo. Al termine dell'intervallo di tempo specificato passiamo a spegnere la luce verde ed accendere il giallo. Un altro periodo di attesa e poi passiamo a spegnere la luce gialla ed accendere la luce rossa. Aggiungiamo un ulteriore periodo di attesa, dopo il quale la nostra "sequenza" di istruzioni termina. Approfittiamo per documentare questa "logica" di tipo sequenziale rappresentandola graficamente con la tecnica del diagramma a blocchi, visibili in **Fig. 1**. Ora realizziamo praticamente questo blocco di istruzioni utilizzando il linguaggio Scratch.
Approfittiamo di questo esempio per impratichirci con l'utilizzo delle funzionalità dell'ambiente di sviluppo Sctratch, che abbia-

*Fig. 7*

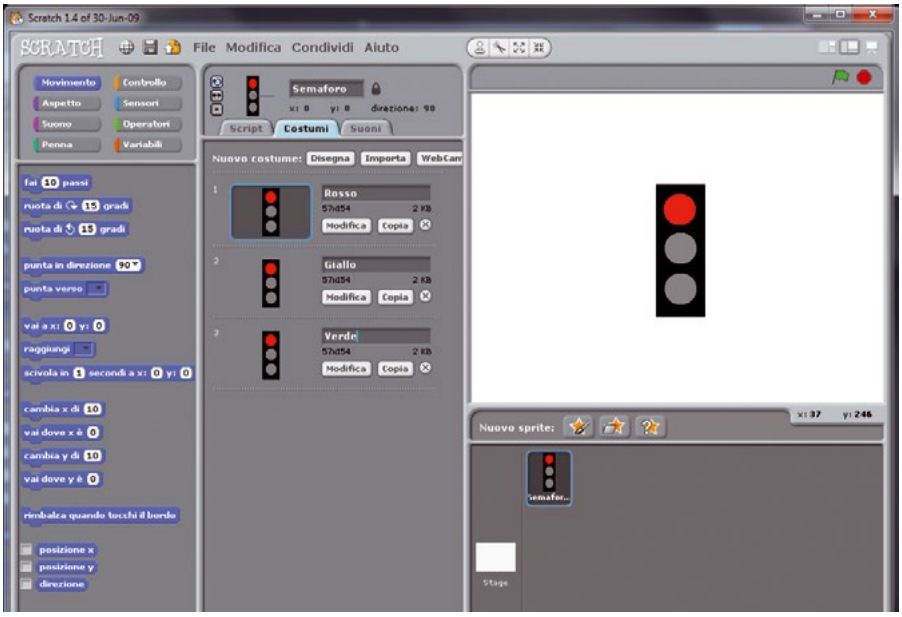
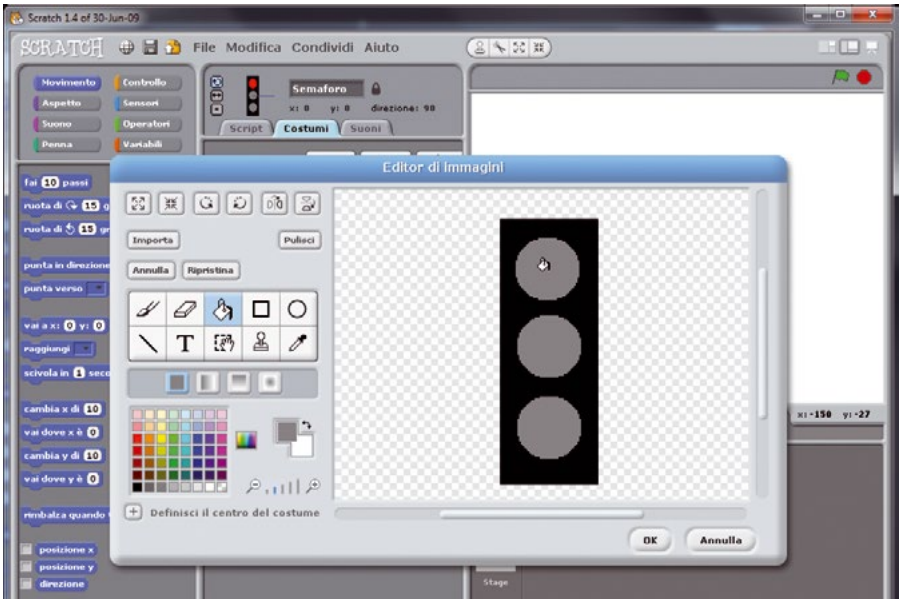



*Fig. 8*




mo presentato nella prima puntata di questa serie. Per prima cosa apriamo il nostro ambiente di sviluppo Scratch, cancelliamo lo sprite del gatto cliccandovi sopra col tasto destro del mouse e selezionando "Cancella". Nello spazio "Nuovo sprite" clicchiamo sulla funzione "Disegna nuovo sprite", la prima stellina a sinistra. Ci appare una finestra in stile "Paintbrush" con un'area di lavoro e gli strumenti necessari a comporre un disegno. Nella tavolozza dei colori selezionate il colore rosso. Poi selezionate lo strumento "ellissi" e con questo tracciate un cerchio come quello visibile in **Fig. 2**.

Ora selezionate lo strumento "selezione" ed "inquadrate" il cerchio appena realizzato, sempre come visibile in **Fig. 2**. Premete i tasti Control-C per duplicare la selezione, spostate il secondo cerchio al di sotto del primo ed eseguite l'operazione una seconda volta. Il risultato finale è visibile in **Fig. 3**, tre cerchi rossi allineati verticalmente.

Per realizzare la versione "rossa" del semaforo dobbiamo dare un colore neutro alle due luci inferiori. Nella tavolozza dei colori clicchiamo su un grigio, selezioniamo lo strumento "riempimento" (il secchiello che rovescia il colore) e clicchiamo al centro del secondo cerchio e, successivamente, del terzo. Vediamo il risultato in **Fig. 4**.

Per disegnare il "corpo" al semaforo, selezioniamo il colore nero dalla tavolozza, selezioniamo lo strumento "rettangolo", e tracciamo un rettangolo a destra dei cerchi in grado di contenerli tutti con un po' di margine. Se qualche operazione riesce male, niente paura. Si può tornare indietro con i tasti Control-Z oppure utilizzare lo strumento "gomma da cancellare". Notate che potete personalizzare gli strumenti di disegno. Ad esempio per lo strumento gomma, sotto il pannello degli strumenti, potete scegliere lo spessore dello strumento, più o meno fine. Per lo strumento riempimento potete scegliere la sfumatura, mentre per i poligoni potete scegliere se realizzarli pieni o con il solo contorno. Una volta ottenuto il risultato visibile in **Fig. 5**, selezionate ancora una volta lo strumento "selezione" e "circondate i tre cerchi, per poi trascinarli all'interno del rettangolo nero, come in **Fig. 6**. Per posizionare correttamente la selezione potete aiutarvi con i tasti freccia. Una volta soddisfatti del risultato confermate con il tasto "OK" e tornate nella finestra principale dell'IDE di Scratch. Se non è attivo il pannello "Costumi" lo attiviamo cliccando sulla linguetta "Costumi". A questo punto avete il disegno del semaforo come sprite ed il primo "costume" disponibile, quello con la versione "rossa" del semaforo. Per realizzare le versioni "gial

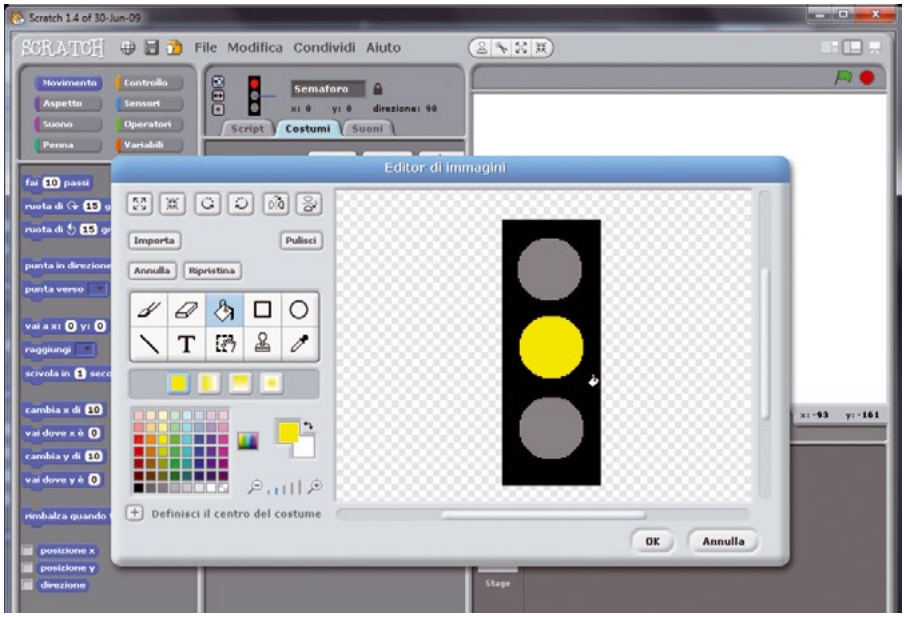



***Fig. 9***

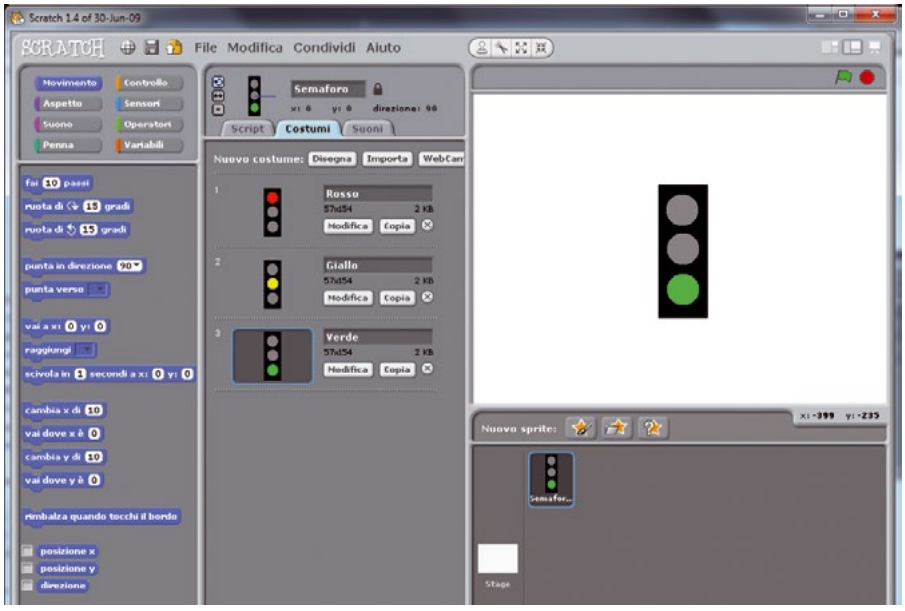



***Fig. 10***

***Fig. 11***


la" e "verde"clicchiamo per due volte sul bottone "copia" a fianco del primo "costume". Approfittate per dare dei nomi significativi allo sprite ed ai suoi costumi. Noi abbiamo scelto il nome "Semaforo" per lo sprite e "rosso", "giallo" e "verde" per i tre costumi come visibile in Fig. 7. Ora cliccate sul pulsante "Modifica" del costume "giallo". Nella finestra di modifica selezionate lo strumento "pipetta" cliccate sul colore grigio di una delle due "luci", selezionate lo strumento "secchiello" e cliccate sulla luce rossa, in modo da riempirla di grigio come in **Fig. 8**.

Ora selezionate un bel giallo dalla tavolozza dei colori e cliccate sulla luce centrale, come in **Fig. 9**. Salvate con il pulsante "OK" e ripetete l'operazione per il costume "verde" fino ad ottenere il risultato visibile in **Fig. 10**.
Ora è tempo di animare il semaforo con le istruzioni che concretizzano il diagramma a blocchi di **Fig. 1** e che potete vedere nella **Fig. 11**. Il significato delle istruzioni si commenta da solo. Per realizzarlo cliccate sullo sprite "Semaforo" nell'area in basso a destra e poi cliccate sulla linguetta "Script". Nell'area degli script riproducete il blocco di **Fig. 11**. I componenti si trovano nelle "collezioni" selezionabili mediante i pulsanti in alto a destra. I colori che evidenziano i pulsanti corrispondono al colore dei blocchi contenuti. Quindi il cappello con la bandierina, che è un blocco di colore giallo lo trovate nella collezione "Controllo" contraddistinta dalla fascetta di colore giallo. Così per gli altri blocchi. Per "portare" i blocchi nell'area degli script basta trascinarli con il mouse e rilasciarli dove desiderato. Abbiamo già descritto nella prima puntata le modalità di editing degli script. Per comodità ne riportiamo un riassunto. Quando si trascina un'istruzione all'interno di uno script il posizionamento viene facilitato dall'apparire di una linea bianca che indica che in quel punto può essere "agganciata" l'istruzione che si sta trascinando. Blocchi di istruzioni possono essere staccati e spostati oppure duplicati con le apposite funzioni raggiungibili cliccandovi sopra con il tasto destro del mouse. Per eliminare un'istruzione dallo script bisogna prima "staccarla" dallo script trascinandola in una zona libera dell'area degli script. Poi cliccarvi sopra col

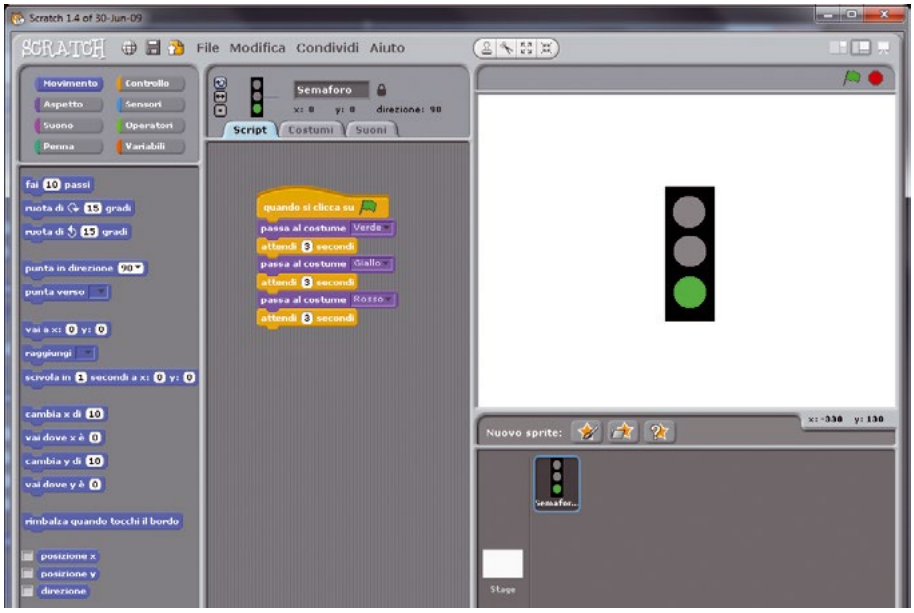


*Fig. 12*


*Fig. 13*


*Fig. 14*


*Fig. 15*


*Fig. 16*

tasto destro del mouse e selezionare "Cancella". Per selezionare il costume nelle istruzioni "passa al costume", basta selezionarli dalla lista che si apre cliccando sulla piccola freccetta nera rivolta verso il basso. Per variare il tempo nelle istruzioni "attendi <x> secondi", cliccate nell'area bianca dell'istruzione ed inserite il valore desiderato. Alla fine il risultato deve apparire come in **Fig. 12**.


*Fig. 17*

Cliccate sulla bandierina verde per mettere in funzione il semaforo.

**CICLO (O ITERAZIONE)**

Carino il nostro semaforo? Peccato che funzioni una sola volta, per poi rimanere congelato nella configurazione finale, un rosso perenne che crea una coda di veicoli tendente a infinito. Per


*Fig. 18*

*Fig. 19*

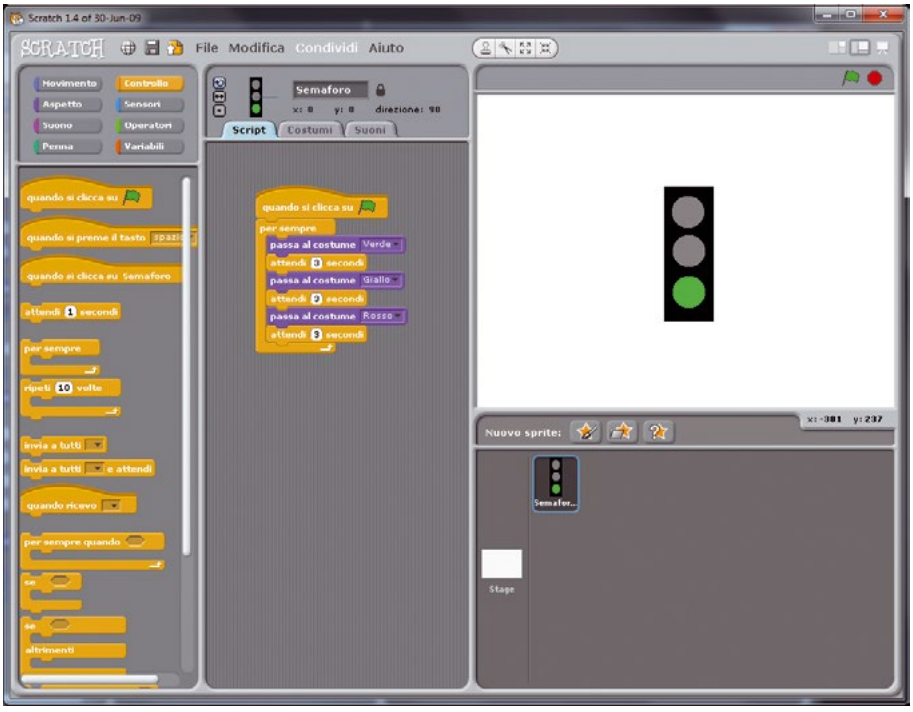



*Fig. 20*


*Fig. 21*

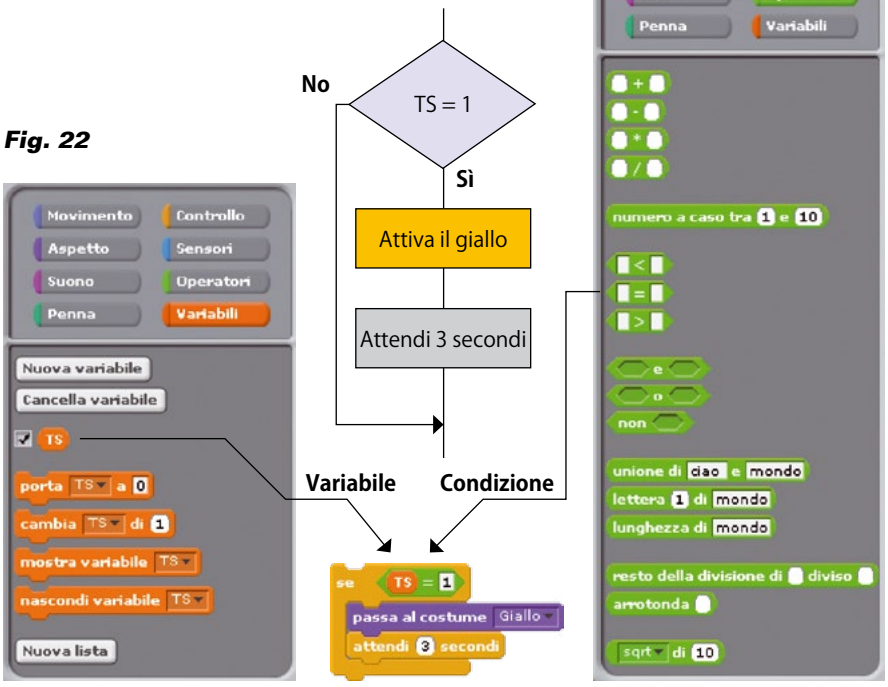



*Fig. 22*

comportarsi da vero semaforo, dopo un certo periodo di luce rossa deve tornare il verde e così di seguito. Il blocco di istruzioni che realizza il comportamento del semaforo, una volta terminato, deve essere rieseguito dall'inizio, in un ciclo continuo senza fine. Per realizzare questo comportamento ci viene in aiuto la struttura di controllo "ciclo". In realtà esistono diverse "versioni" della struttura ciclo, a seconda delle "regole" in base alle quali vogliamo che sia controllata la ripetizione del blocco di istruzioni. A seconda dei linguaggi la struttura di controllo ciclo è implementata dalle istruzioni "while", "loop", "repeat", "for" e "do … until". La grande distinzione sta tra i cicli incondizionati come "loop" o, in Scratch, "forever" o "per sempre" in italiano (**Fig. 13**), ed i cicli condizionati. Questi ultimi a loro volta possono essere "pilotati" per eseguire un blocco di istruzioni per un certo numero di volte (in Scratch, per esempio, "repeat 5" o "ripeti 5 volte" (**Fig. 14**) oppure, in base al verificarsi o meno di condizioni particolari, come "per sempre quando <condizione>" (**Fig. 15**) o "ripeti fino a quando <condizione>" (**Fig. 16**).
Siccome il blocco di istruzioni che anima il nostro semaforo deve essere ripetuto incondizionatamente, adottiamo il blocco "per sempre". Il diagramma a blocchi della logica di funzionamento del semaforo viene modificato come quello visibile in **Fig. 17**, dove il flusso di elaborazione, una volta terminato, torna all'inizio della sequenza. Per tradurlo nel linguaggio Scratch, visibile in **Fig. 18**, "stacchiamo" la sequenza di istruzioni dal "cappello" iniziale. Nel pannello "Controllo" identifichiamo il blocco a forma di "C" "per sempre" e lo trasciniamo nell'area degli script agganciandolo sotto al cappello con la bandierina verde. Poi trasciniamo il blocco di istruzioni, che "governano" il semaforo, all'interno della "bocca" del blocco "per sempre". Fatto? Verificate di trovarvi nella condizione equivalente a quella visibile in **Fig. 19**. Cliccate sulla bandierina et voilà, ora il semaforo funziona in modo corretto. Divertitevi a modificare il suo

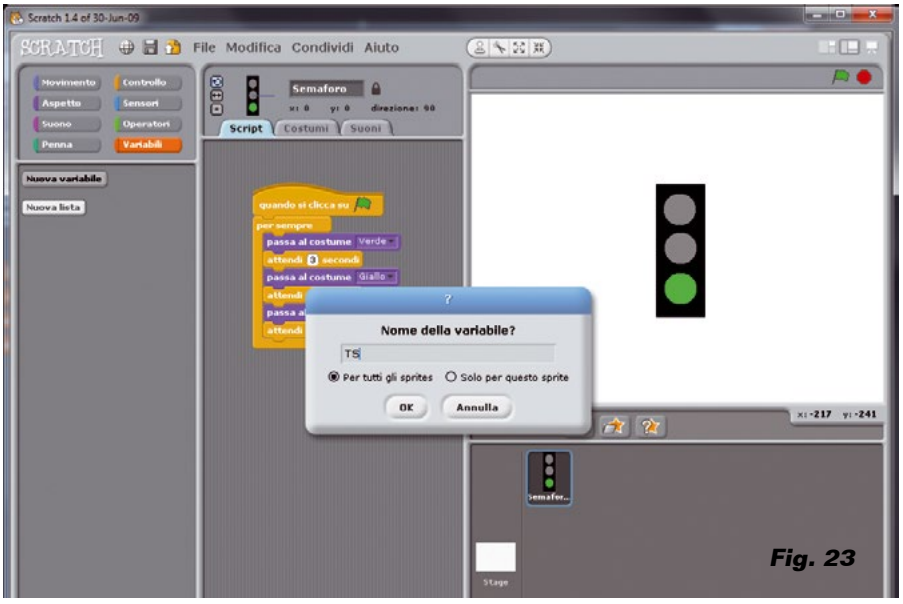



*Fig. 23*

*Fig. 24*

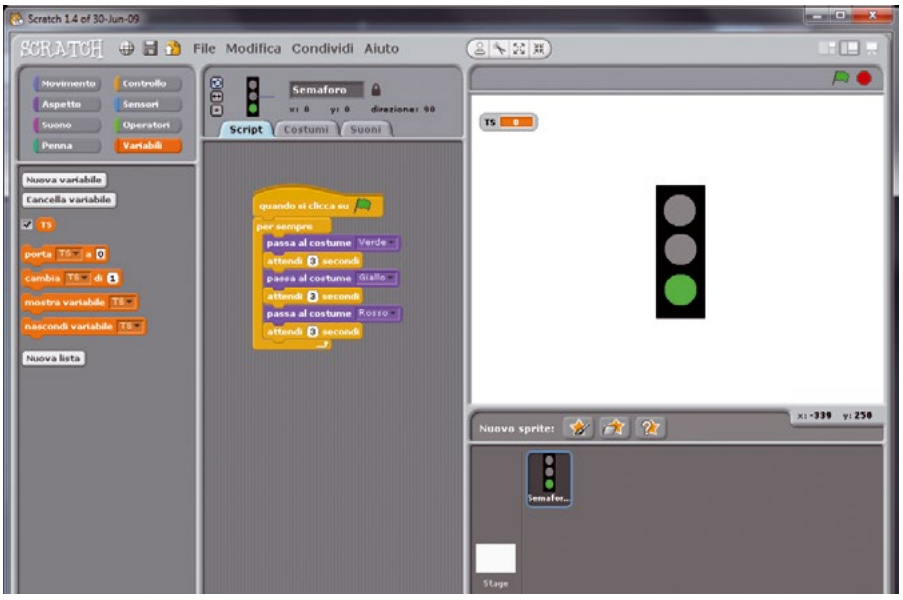
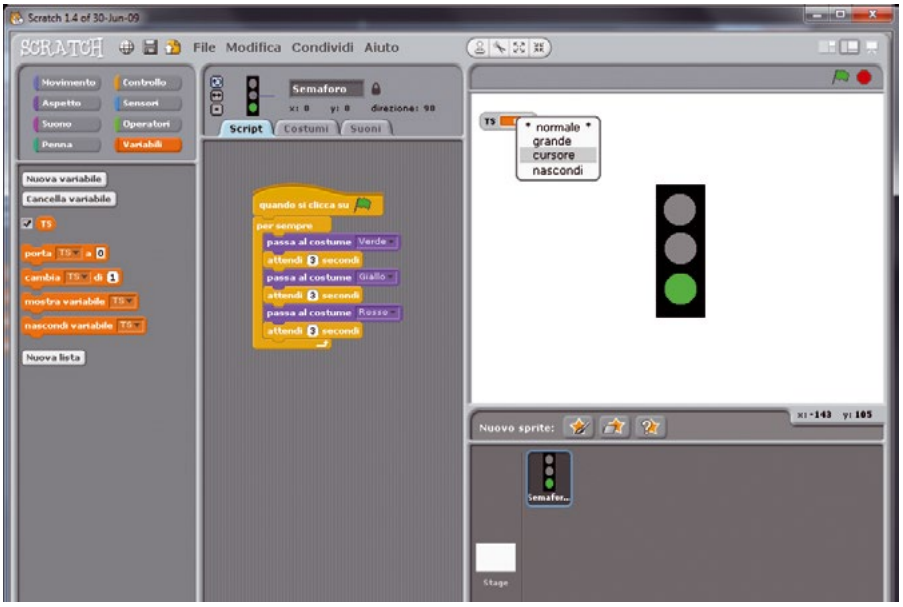



*Fig. 25*




comportamento, per esempio ripassando dal giallo tra rosso e verde, come avviene in alcuni nazioni, o modificando le temporizzazioni. Ora è il momento di cominciare a complicarci la vita, ovvero impariamo a prendere delle decisioni.

## SELEZIONE

Bella la vita del semaforo anche se decisamente noiosa e, per dirla tutta, non particolarmente intelligente. Il suo unico scopo è di cambiare colore a intervalli fissi. Poco importa se sul lato che diventa verde non c'è nessuno e dalla parte rossa interrompe un flusso di veicoli chilometrico. O, peggio ancora, non prolunga il giallo finché da un lato non si libera l'incrocio, dando via libera nell'altra direzione, generando un ingorgo inestricabile. Nella maggior parte delle attività umane, o che gli umani delegano alle macchine, i problemi non possono essere risolti con la semplicità degli esempi precedenti. A seconda delle condizioni che si presentano, occorre prendere delle decisioni. Ciò è vero sia per gli individui umani che per le macchine. Per questo ci viene in aiuto la struttura di controllo della "selezione", ovvero la capacità di intercettare e valutare diverse tipologie di condizioni che si possono verificare, ed in base a queste, condizionare l'esecuzione di alcuni blocchi di istruzioni in alternativa ad altri. Per esempio potremmo voler fare in modo che il semaforo sia predisposto per funzionare sia con il passaggio del giallo tra il rosso ed il verde che con il passaggio diretto dal rosso al verde. Tutto questo utilizzando un flag o "variabile" di configurazione, impostabile dall'esterno, senza bisogno di dover modificare il codice del programma per cambiarne il modo di funzionamento. Per prima cosa dobbiamo creare la variabile "TS" (Tipo Semaforo). Poi modifichiamo il comportamento del programma introducendo la struttura di controllo selezione nella gestione del giallo durante la transizione dal rosso al verde. A questo proposito specifichiamo la possibilità di scegliere tra due differenti "sfumature" del blocco di controllo selezione: la selezione ad una via e la selezione a due vie. La prima possibilità permette di controllare l'esecuzione di un blocco di istruzioni al verificarsi della condizione che pilota la struttura di controllo, nel caso la condizione non sia verificata il blocco viene saltato e l'elaborazione prosegue con l'istruzione successiva alla chiusura del blocco, dopo la parte inferiore della "Bocca" (**Fig. 20**). La seconda possibilità permette di eseguire due blocchi di istruzioni in alternativa a seconda del verificarsi o meno della condizione che pilota la struttura di controllo (**Fig. 21**).

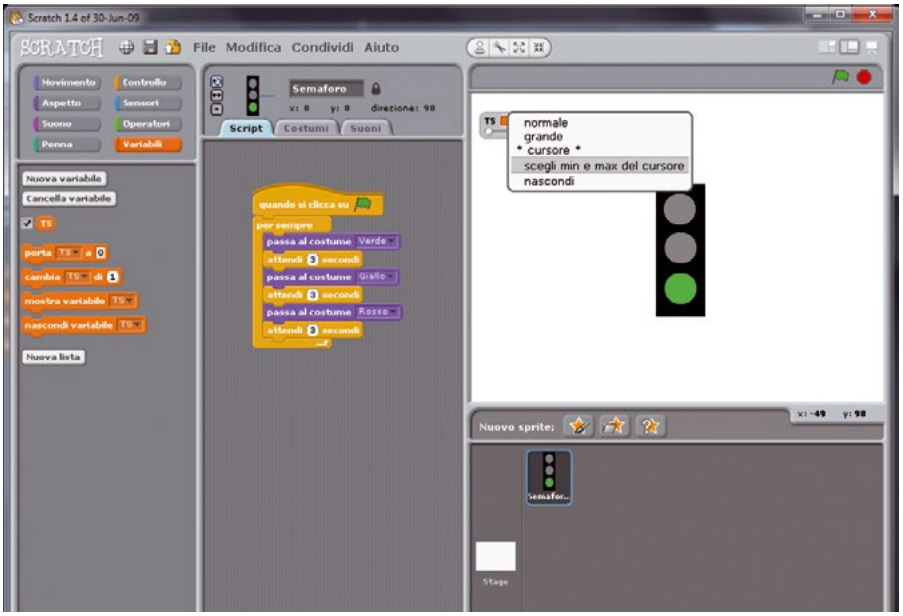



***Fig. 26***

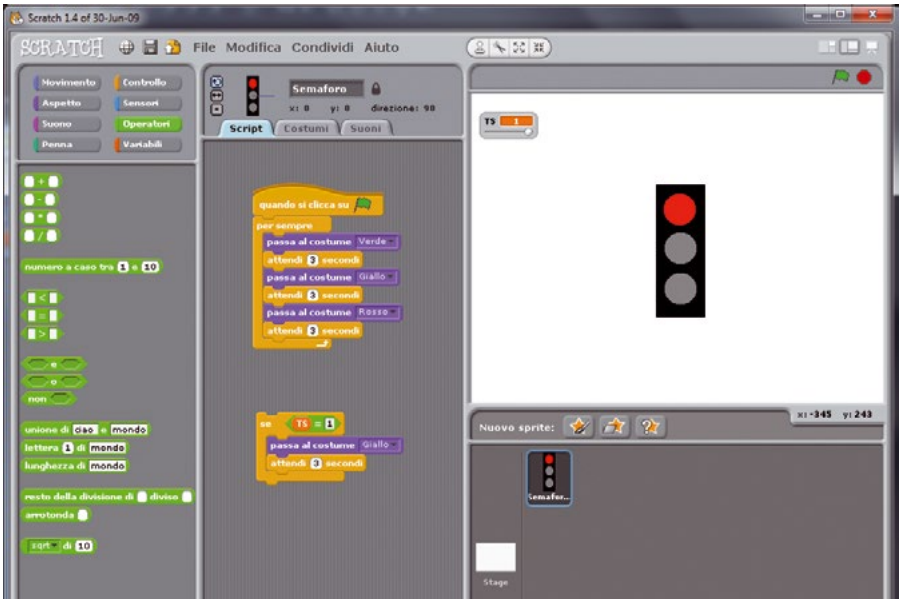



***Fig. 27***

Per il nostro semaforo è sufficiente adottare la struttura ad una via, come visibile in **Fig. 22**. Nel caso la variabile TS contenga il valore "1", facciamo apparire il giallo nella transizione tra rosso e verde, altrimenti saltiamo il blocco di istruzioni ed eseguiamo in sequenza le istruzioni successive.
Per realizzare il blocco di selezione visibile in **Fig. 22** dovete per prima cosa creare la variabile TS. Per creare la variabile TS assicuratevi di aver selezionato lo sprite "Semaforo", per essere certi cliccate sullo sprite stesso nell'area in basso a destra. Poi cliccate sul pulsante "Variabili" nel menu delle collezioni di blocchi in alto a sinistra. Nel pannello che si apre cliccate sul pulsante "Nuova variabile". Si aprira una finestra di dialogo nella quale inserire il nome della variabile, nel nostro caso "TS", come visibile in **Fig. 23**. Lasciate invariato il resto e confermate cliccando sul tasto "OK".
Vedrete apparire la nuova variabile, corredata dei blocchi di istruzioni per poterla gestire. Il primo blocchetto in alto con il segno di spunta è il riferimento della variabile stessa, da trascinare negli spazi di forma corrispondente nelle istruzioni. Il segno di spunta permette di visualizzare la variabile sul "palcoscenico" (Stage).
Lasciate il segno di spunta e cliccate col tasto destro del mouse sul segnaposto della variabile sul palcoscenico.
Selezionate l'opzione "cursore" come in **Fig. 24**.
Questa operazione fa apparire un cursore al di sotto della variabile, che può essere utilizzato per modificarne il valore, anche con il programma in esecuzione. Proprio quello che ci serve, però i valori possibili devono essere limitati a "0" e "1". Quindi cliccate di nuovo col tasto destro del mouse sul segnaposto della variabile e selezionate la voce di menu "scegli min e max del cursore" come in **Fig. 25**.
Nella finestra di dialogo indicate "0" come valore minimo e "1" come valore massimo come in **Fig. 26**.
Create il blocco "se" trascinando e collocando i vari "pezzi" come indicato nello schema di **Fig. 22** ed inseritelo all'interno del blocco principale come visibile in **Fig. 27**.
Potete mettere in funzione il semaforo cliccando sulla bandierina verde.
A seconda della posizione del cursore della variabile "TS", e quindi del valore impostato nella variabile stessa, vedrete il semaforo comportarsi in modo diverso nella transizione dal colore rosso al verde, con o senza passaggio per il giallo. Se volete conservare il vostro lavoro salvatelo selezionando "Salva con nome" dal menu "File".

*Fig. 28*

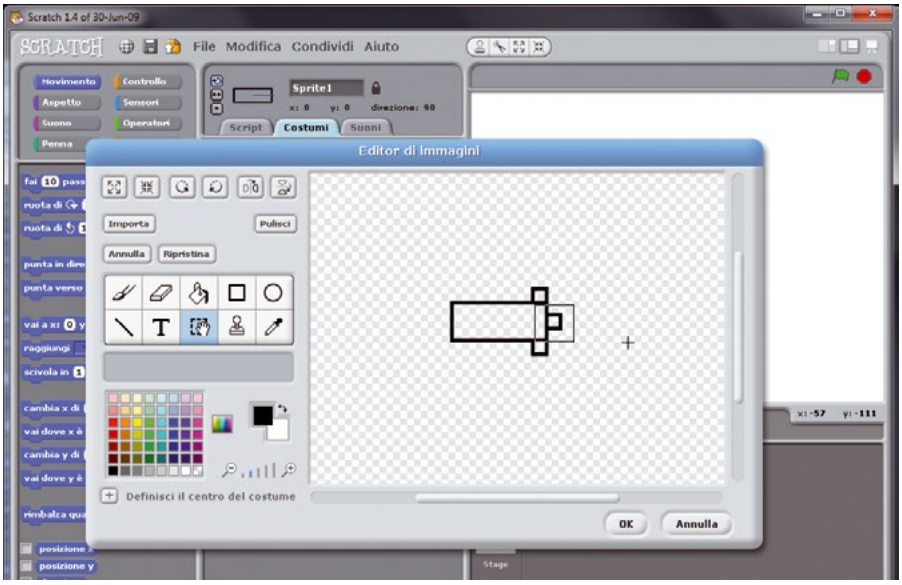



*Fig. 29*

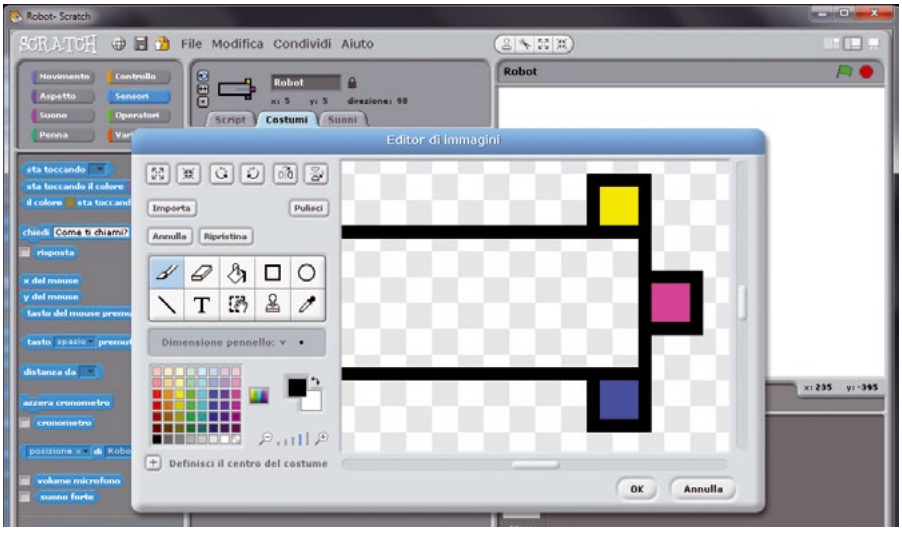
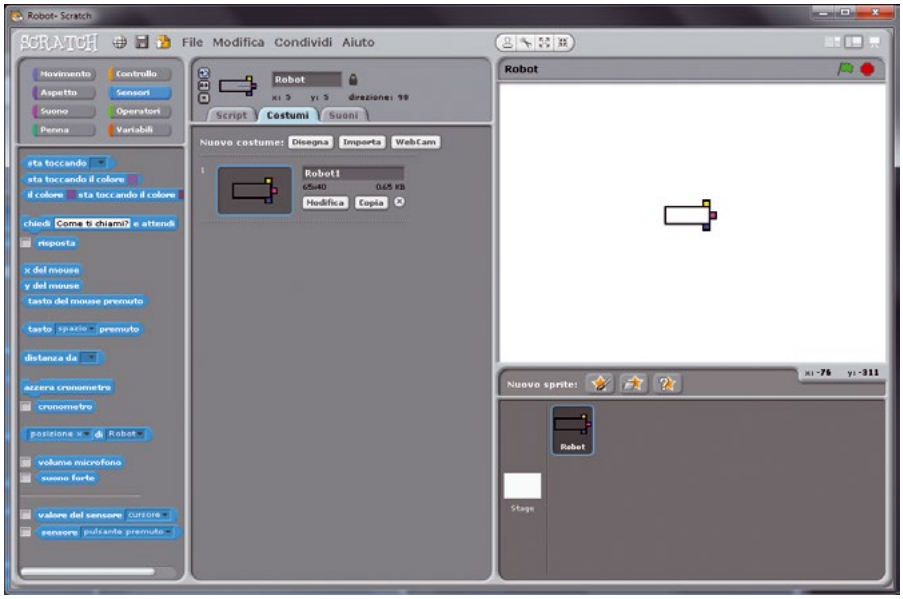



*Fig. 30*




**FACCIAMOCI UN ROBOT**

Una volta familiarizzato con le "strutture" di base della programmazione possiamo cimentarci con qualcosa di decisamente più ambizioso della gestione di un semaforo. Per esempio un simulacro di robot che deve seguire un percorso realizzato con una linea nera su un "pavimento" bianco. Con il linguaggio Scratch? Sì, con il linguaggio Scratch. Normalmente i robot che "inseguono" una linea sono dotati frontalmente di un fila di sensori sensibili all'infrarosso, almeno tre, che misurano il grado di riflessione del terreno sottostante, alto nel caso il colore sottostante sia bianco e basso nel caso sia nero. Nella simulazione del robot in Scratch, al posto dei sensori tradizionali, utilizziamo un metodo basato sui colori, in modo da poterci avvalere della condizione di selezione, disponibile nella collezione Scratch, "se il colore <X> sta toccando il colore <Y>". A questo punto possiamo usare dei sensori a "colori" che possiamo utilizzare con una logica simile a quella utilizzata con i sensori "veri". Prima di approfondire la "logica" di controllo realizziamo il nostro robot ed il "terreno" con la linea nera da seguire. Dopo aver fatto questo la logica di controllo apparirà più chiara. Aprite un nuovo progetto, cancellate lo sprite gatto e cliccate sulla stellina "Disegna nuovo sprite". Disegnate una struttura come quella visibile in **Fig. 28**.

Il rettangolo di dimensioni maggiori rappresenta il corpo del robot mentre i tre quadrati più piccoli costituiscono i sensori. Per disegnare il corpo del robot selezionate lo strumento "pennello", primo in alto a sinistra, selezionate un tratto "fine" (per esempio la seconda selezione

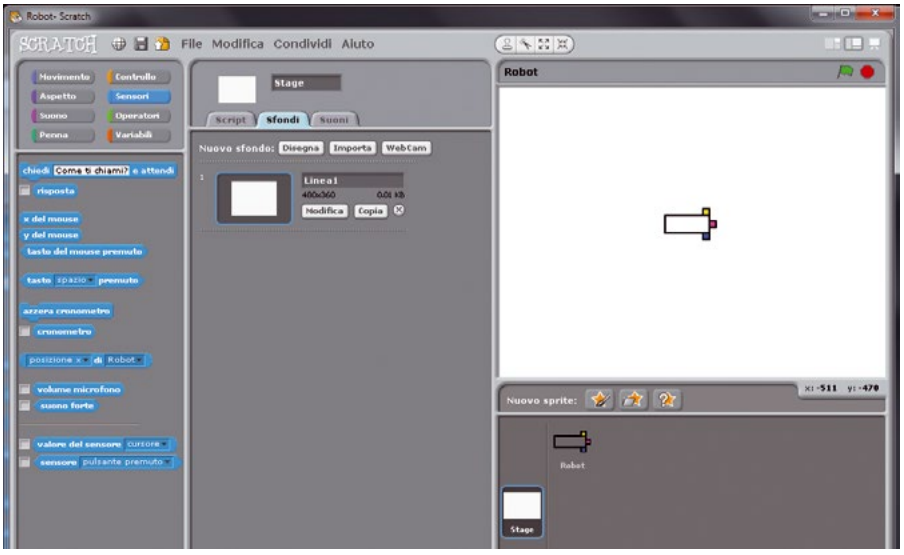



***Fig. 31***

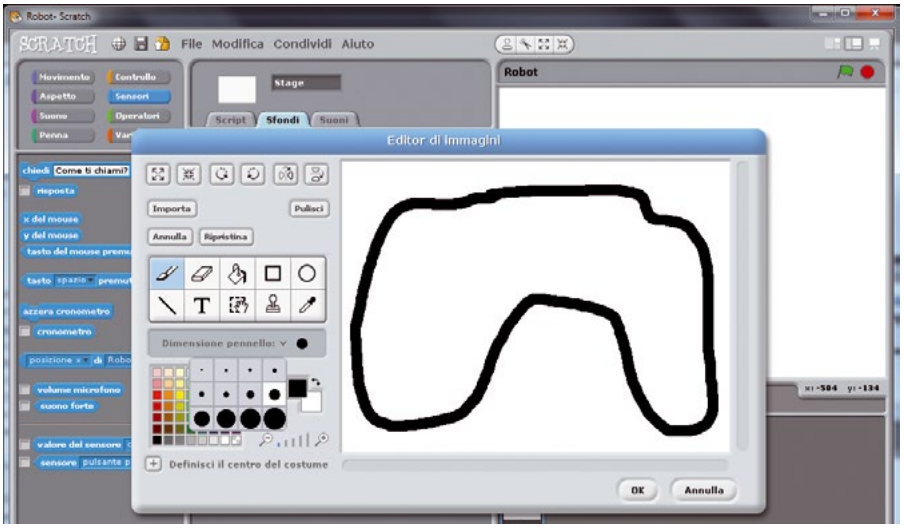



***Fig. 32***

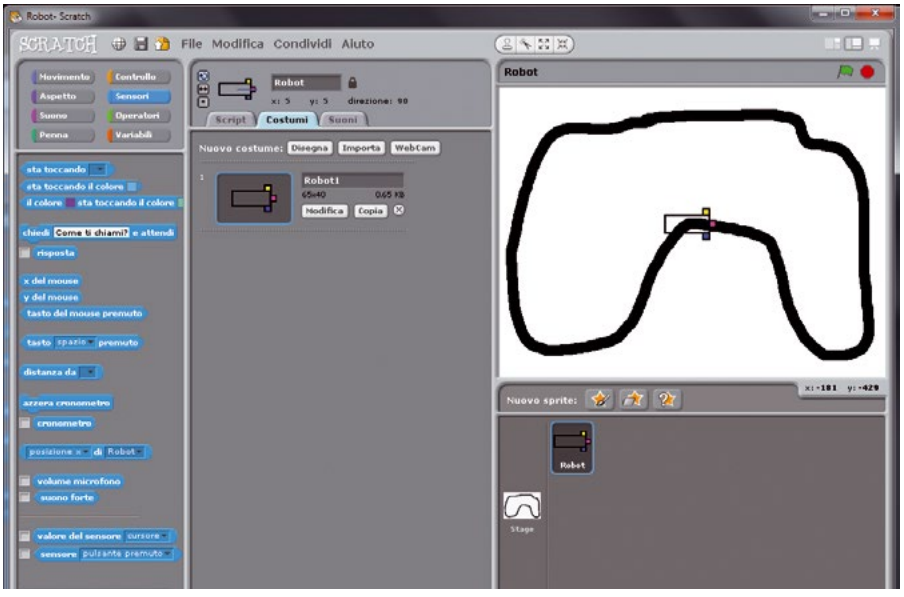
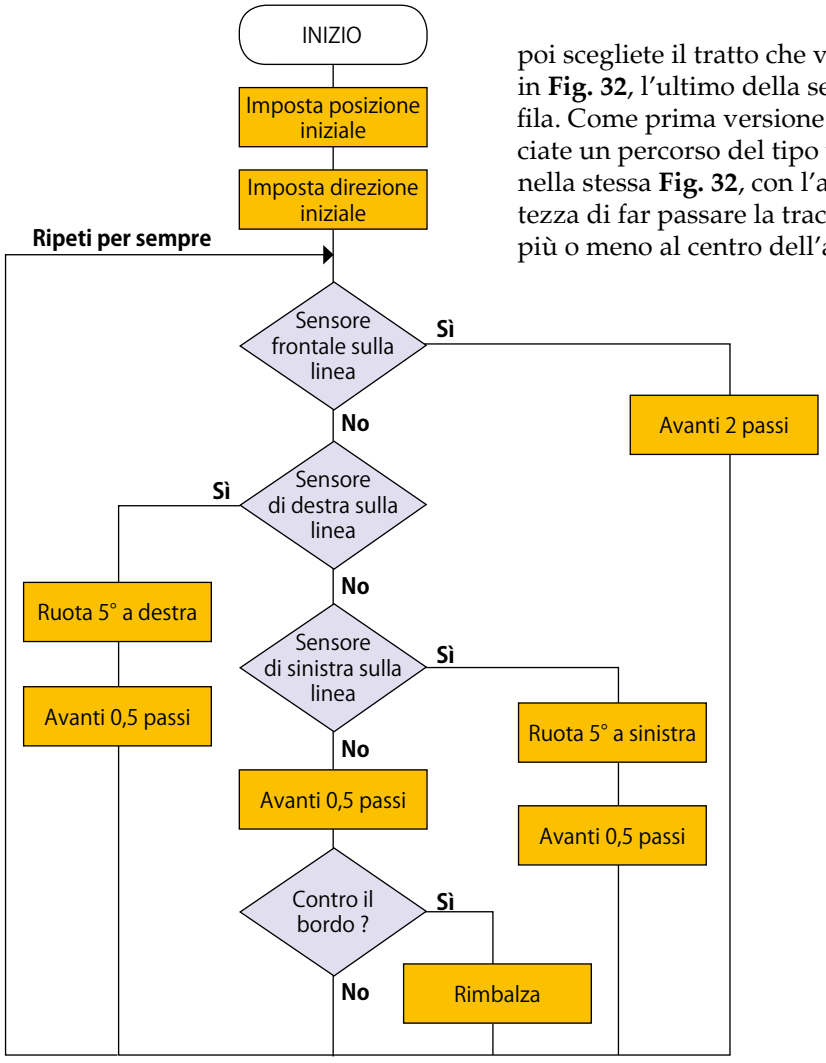



***Fig. 33***

disponibile), e tracciate un rettangolo dalle proporzioni come quello in figura. Poi disegnate un quadrato, in un qualunque punto del palcoscenico. Selezionate lo strumento "selezione". Selezionate il quadrato, cliccate col tasto destro del mouse e selezionate "Copia", trascinate la copia nella posizione di uno dei sensori. Riselezionate, copiate di nuovo l'originale e trascinatela nella posizione del secondo sensore. Infine selezionate il quadrato originale e trascinatelo nella posizione del terzo sensore. Ora dovete "personalizzare" ciascun sensore con un colore diverso. Allargate il disegno con lo strumento "zoom" ovvero le tacchette in basso a destra della tavolozza. Cliccando su una o l'altra tacchetta si allarga o restringe il disegno. Ora colorate i tre sensori con colori diversi, noi abbiamo scelto il giallo, il blu ed il viola. Selezionate il colore giallo cliccandovi sopra nella tavolozza. Poi selezionate lo strumento "secchiello" e riempite il sensore di sinistra con il colore giallo. Vi consigliamo di seguire lo schema di colori che abbiamo adottato in modo da avere riscontro con le istruzioni e la logica complessiva dello script. Ripetete l'operazione per i colori blu e viola fino ad ottenere il risultato finale visibile in **Fig. 29**. Confermate cliccando sul pulsante "OK" e date un nome allo sprite ed al costume appena realizzati, per esempio "Robot" e "Robot1", come visibile in **Fig. 30**. Ora dobbiamo preparare il "percorso" che il nostro robot dovrà seguire. Cliccate sull'icona "Stage" nell'area in basso a destra e nel pannello date un nome allo "sfondo" e cliccate sul pulsante "Modifica" (**Fig. 31**).

Nella finestra di disegno selezionate lo strumento "pennello",

*Fig. 34*

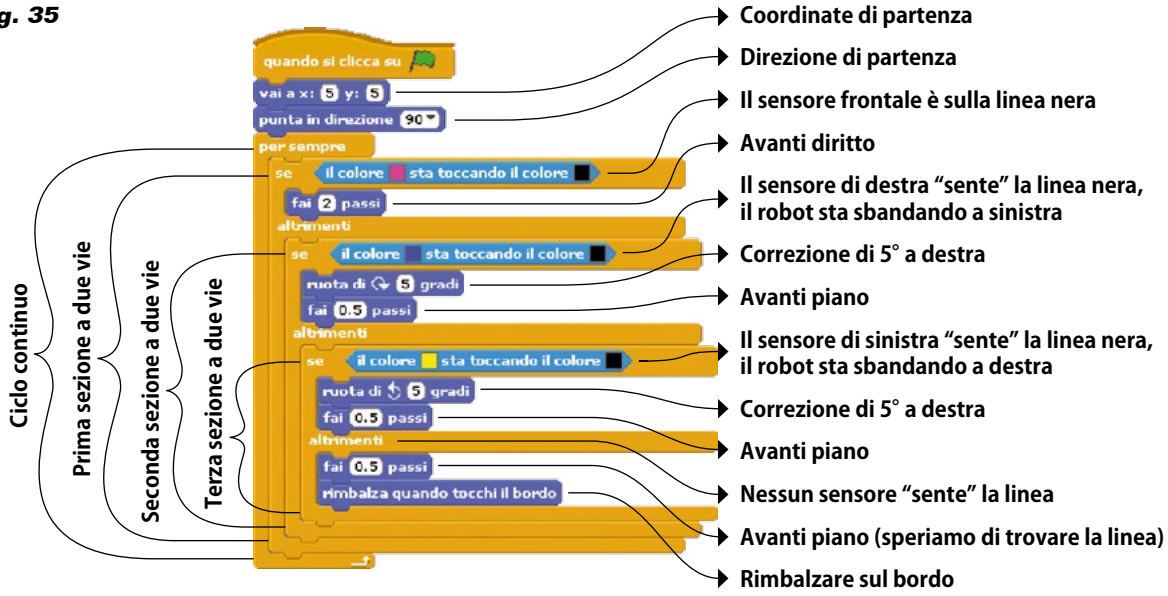



poi scegliete il tratto che vedete in **Fig. 32**, l'ultimo della seconda fila. Come prima versione tracciate un percorso del tipo visibile nella stessa **Fig. 32**, con l'accortezza di far passare la traccia più o meno al centro dell'area di disegno. Confermate con "OK". Il risultato finale è visibile in **Fig. 33**.

Ora dobbiamo istruire a dovere il nostro Robot per fargli seguire la linea nera. È a questo punto che entra in gioco la capacità di ragionare in modo logico e creativo per trovare una soluzione al problema. Le variabili in gioco sono molte, la tipologia del percorso con curve più o meno accentuate, il posizionamento dei sensori, la forma del robot, eccetera. Una prima semplice strategia, adatta al robot ed al tipo di percorso che avete disegnato può essere la seguente:

- Se il sensore frontale, quello viola sta "toccando" la linea nera, il robot prosegue diritto;
- Se il sensore di "destra", di colore blu, sta toccando la linea nera, significa che il robot sta "sbandando" a sinistra, oppure che la linea sta incurvandosi verso destra. In questo caso il robot deve "girare" verso destra di un certo angolo e, prudenzialmente, avanzare più lentamente;
- Se il sensore di "sinistra", di

*Fig. 35*



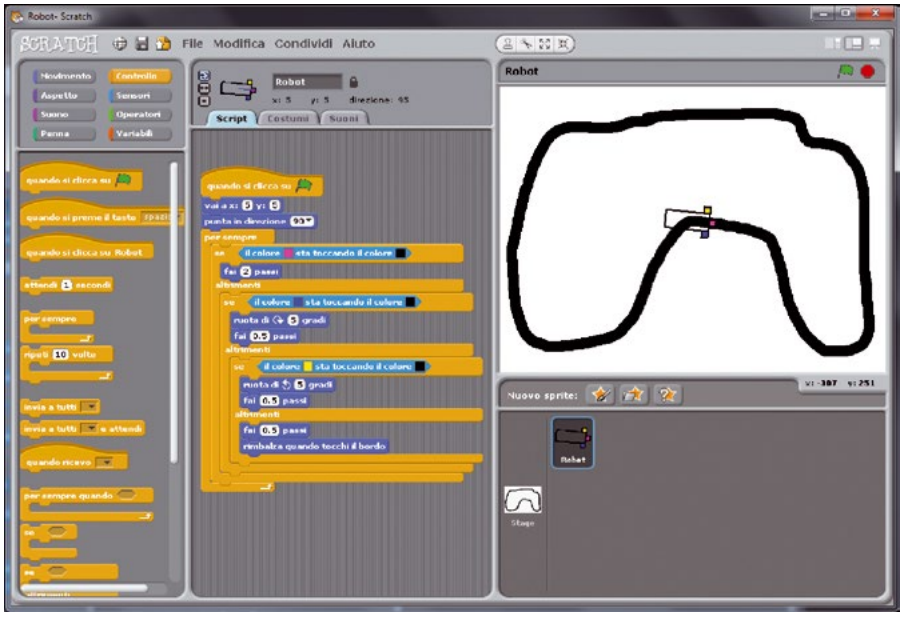


***Fig. 36***

colore giallo, sta toccando la linea nera significa che il robot sta sbandando verso destra oppure che la linea nera sta piegando a destra. Il robot deve girare verso sinistra di un certo angolo ed avanzare più lentamente;

- Infine, se nessuno dei sensori tocca la linea nera, significa che il robot si è perso, oppure la linea è interrotta (provate, cancellando un pezzo di linea). Il robot rallenta la velocità e prosegue dritto, sperando di incontrare la linea nera di nuovo, prima o poi.
- Nel caso vada a sbattere contro un bordo, deve rimbalzare, rientrando in campo e proseguendo come prima.

Abbiamo tradotto questa strategia nel diagramma a blocchi di **Fig. 34**, ed in base a questo abbiamo realizzato lo script Scratch visibile in **Fig. 35**.
Lo script prevede un ciclo "per sempre" e tre blocchi di selezione a due vie "annidati" in modo da coprire tutti i casi previsti nella strategia. Potete ovviamente adottare tutte le soluzioni alternative che esprimano lo stesso comportamento "logico", per esempio utilizzando una sequenza di blocchi di selezione ad una via. Diverse soluzioni possono differire per il comportamento e per le prestazioni, provate più che potete per fare esperienza.
Componete lo script di **Fig. 35** attivando lo sprite "Robot", selezionando il pannello "Script" e poi "scovando" i blocchi nelle diverse collezioni e trascinandoli sull'area dello script, collegandoli assieme nel modo corretto.
Un'unica nota relativamente alla condizione di selezione "se il colore <X> sta toccando il colore <Y>". Per impostare correttamente i colori, cliccate sul primo quadratino della condizone, poi andate a selezionare il colore cliccando sul sensore desiderato. Controllate bene il risultato, bisogna operare con una certa precisione e "centrare" con precisione il colore del sensore. Nel caso ripetete l'operazione. Il colore del secondo quadratino è più facile da impostare, in quanto la linea nera è decisamente più semplice da "centrare". Il blocco di selezione ritorna il valore vero, se si verifica la condizione nel "blocco" verde, ovvero il sensore del colore impostato sta toccando la linea nera. In questo caso viene eseguito il blocco di istruzioni contenuto nella prima "bocca" della selezione a due vie. Se la condizione non è soddisfatta viene eseguito il blocco di istruzione contenuto nel ramo "else" della selezione, ovvero la seconda "bocca" del blocco di selezione. Una volta terminato il lavoro dovreste trovarvi nella situazione visibile in **Fig. 36**.
Salvate il progetto con un nome, se non l'avete ancora fatto.
Ora potete provare il tutto cliccando sulla bandierina verde e mettendo in moto il robot. Se per caso funziona male e non segue la linea, il problema potrebbe essere la linea di spessore troppo grande, che impegna più di un sensore alla volta, oppure la distanza tra i sensori troppo piccola. Rifate il disegno del percorso con una linea meno spessa oppure disegnate un robot leggermente più largo. Carino?
A questo punto potete sperimentare moltissime opzioni alternative. Modificate il percorso rendendolo più difficile. Spostate i sensori o cambiate la forma del robot. Modificate i parametri di sterzata e di velocità per tentare il record del percorso. Introducete incroci a "T" e a forma di "+" come avviene per le prove della RoboCup. Ovviamente dovrete modificare la strategia, e magari la composizione del robot con un numero maggiore di sensori in posizioni diverse. Nelle nuove strategie potete sfruttare le condizioni composte "AND" e "OR", aumentando il numero dei blocchi di selezione utilizzati.
Altra possibilità e avvalersi della condizione "distanza da <X>" inserendo nel palcoscenico degli sprite "ostacoli" e provando ad evitarli, oppure a passare tra due sprite "barriera". Fatevi aiutare da amici e bambini.
Il divertimento è assicurato.


# SCRATCH E RASPBERRY PI

di MARCO MAGAGNIN



Sempre più strumenti stanno adottando l'interfaccia grafica di Scratch. Uno di questi, ScratchGPIO7, ci permette di utilizzare il linguaggio grafico di Scratch per gestire il GPIO di Raspberry Pi. Terza parte.

Quando abbiamo deciso di proporre questa serie di articoli, con l'obiettivo di facilitare l'apprendimento della programmazione dei calcolatori elettronici per mezzo di uno strumento didattico dall'aspetto ludico ma dal contenuto professionale, non pensavamo alla reale portata di questo approccio. Pur essendo disponibile da diverso tempo, questa tipologia di strumenti ha conosciuto negli ultimi mesi una diffusione inaspettata. Oltre alla versione di Scratch tradizionale, quella che vi abbiamo presentato fino ad ora, sono apparse nuove soluzioni che ricalcano la stessa

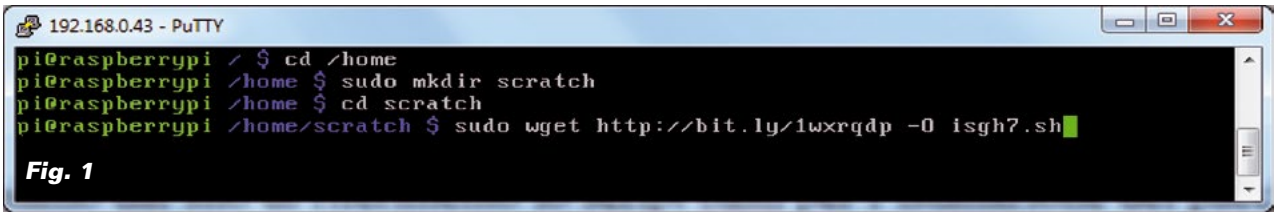



*Fig. 1*

filosofia, soprattutto per quanto riguarda l'impostazione grafica delle applicazioni. Si va dagli ambienti di sviluppo totalmente utilizzabili all'interno di un browser web, come la versione 2.0 di Scratch o l'ambiente Snap (messo a disposizione dall'Università di Berkeley), a quelli dedicati a specifici hardware, fino alle soluzioni, per così dire, proprietarie.
Nel caso delle applicazioni destinate ad uno specifico hardware citiamo ScratchGPIO7 che include le "istruzioni" che permettono di interfacciare il GPIO di Raspberry Pi e che, ovviamente, "gira" su Raspberry Pi; allo stesso modo S4A che sta per "Scratch For Arduino", permette di scrivere gli sketch per Arduino utilizzando elementi simili a quelli utilizzati dal linguaggio grafico Scratch.
Tra le soluzioni "proprietarie" citiamo Wyliodrin, un ambiente che permette di utilizzare i sensori di diversi produttori anche da remoto, in modo da poter realizzare una sorta di sistema di automazione distribuito e gestibile in remoto.
La soluzione prevede un utilizzo gratuito fino ad un numero limitato di schede e di applicazioni, per poi proporre diverse tipologie di abbonamento, in base alle esigenze degli utilizzatori. Ma torniamo alla implementazione di Scratch che permette di interfacciare il GPIO di Raspberry Pi, ovvero il pacchetto ScratchGPIO7. L'ultimo numero rappresenta la versione del software ed incrementa molto velocemente, probabilmente al tempo dell'uscita dell'articolo potrebbe già essere disponibile una nuova versione.
Procediamo con l'installazione. Partiamo da una immagine fresca del sistema operativo Raspbian su Raspberry Pi.
ScratchGPIO7 richiede di essere lanciato dall'interfaccia grafica di Raspberry Pi, quindi dobbiamo "allestire" il nostro Raspberry Pi in "stile" desktop, dotato di monitor HDMI, tastiera e mou-

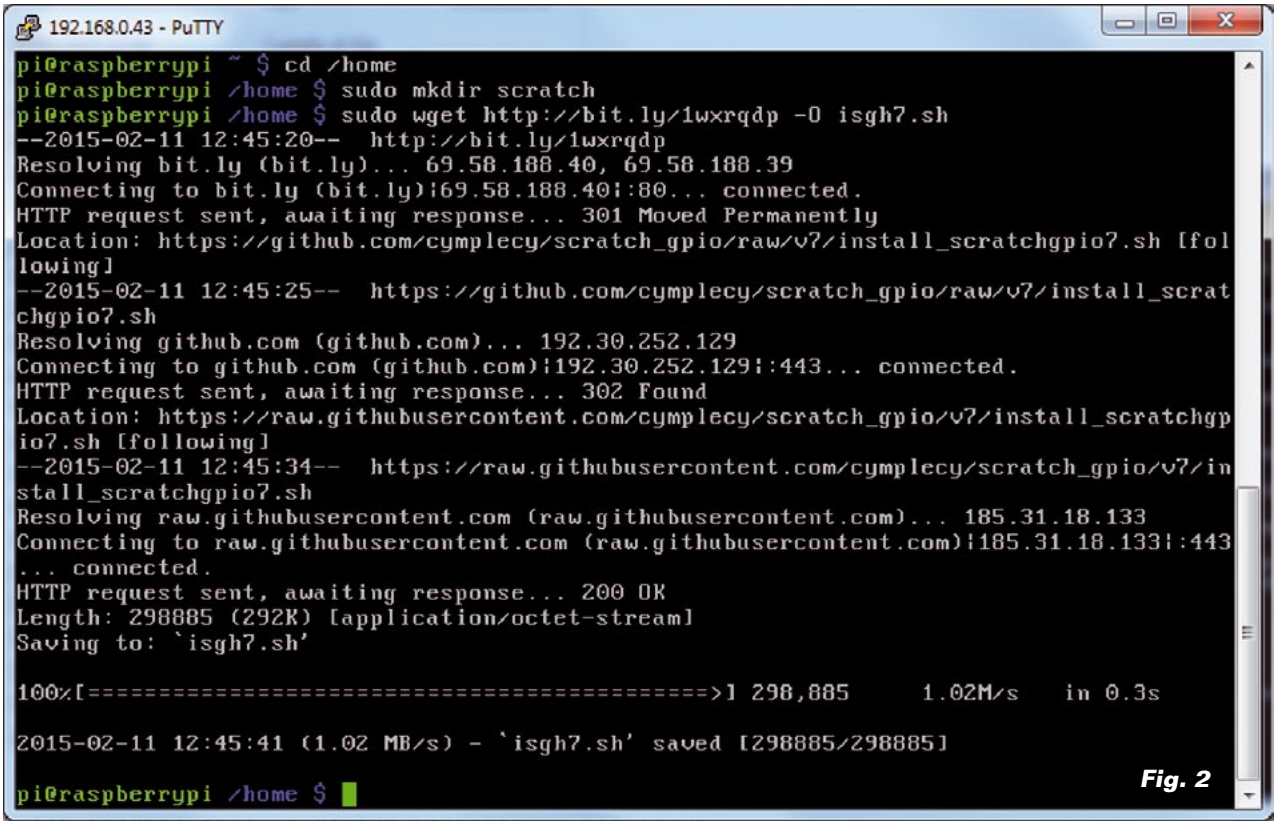



*Fig. 2*

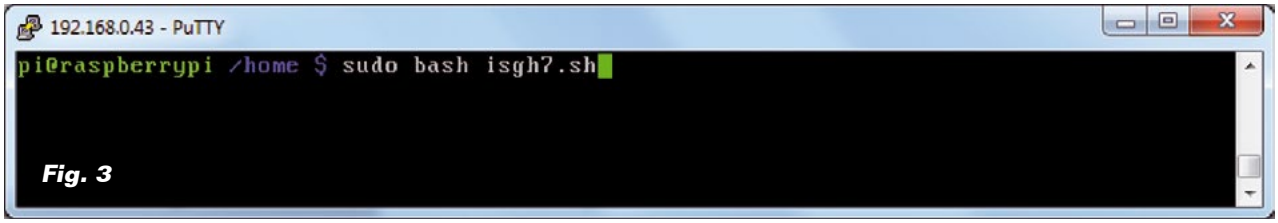



*Fig. 3*

se. Una possibile alternativa è utilizzare, in ambiente Windows, lo strumento di accesso remoto Mobaxterm, come indicato nell'articolo dedicato alla scheda RandA pubblicato nel numero 190 della rivista. Ovviamente, come è di prassi nei sistemi GNU/Linux, Raspberry Pi deve essere collegato alla rete internet, per poter scaricare i pacchetti necessari all'installazione. Apriamo un terminale, in locale od in remoto, ci logghiamo con l'utente "pi" (password "raspberry") ed aggiorniamo il sistema operativo con i comandi consueti, preceduti dal comando "sudo" in quanto non siamo utente "root":

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get dist-upgrade
```

Per l'utilizzo di base di Raspberry Pi e del sistema operativo Raspbian, distribuzione di GNU/Linux basata su Debian, rimandiamo agli argomenti contenuti nel libro "Raspberry Pi – Il mio primo Linux embedded". Terminato l'aggiornamento, che può richiedere un riavvio del sistema con il comando:

```
sudo reboot
```

possiamo procedere all'istallazione del pacchetto ScratchGPIO7. Il sito di riferimento, in inglese, per il pacchetto ScratchGPIO7 è "*http://simplesi.net/scratchgpio/scratch-raspberrypi-gpio/*". Per prepararci all'installazione entriamo nella cartella /home e creiamo al suo interno una cartella di nome "scratch", dove scaricheremo lo script di installazione, come visibile in **Fig. 1**, con i comandi:

```
cd /home
sudo mkdir scratch
cd scratch
```

Scarichiamo dal sito di riferimento lo script bash per l'istallazione del pacchetto con il comando (**Fig. 2**):

```
sudo wget http://bit.ly/1wxrqdp
-O isgh7.sh
```

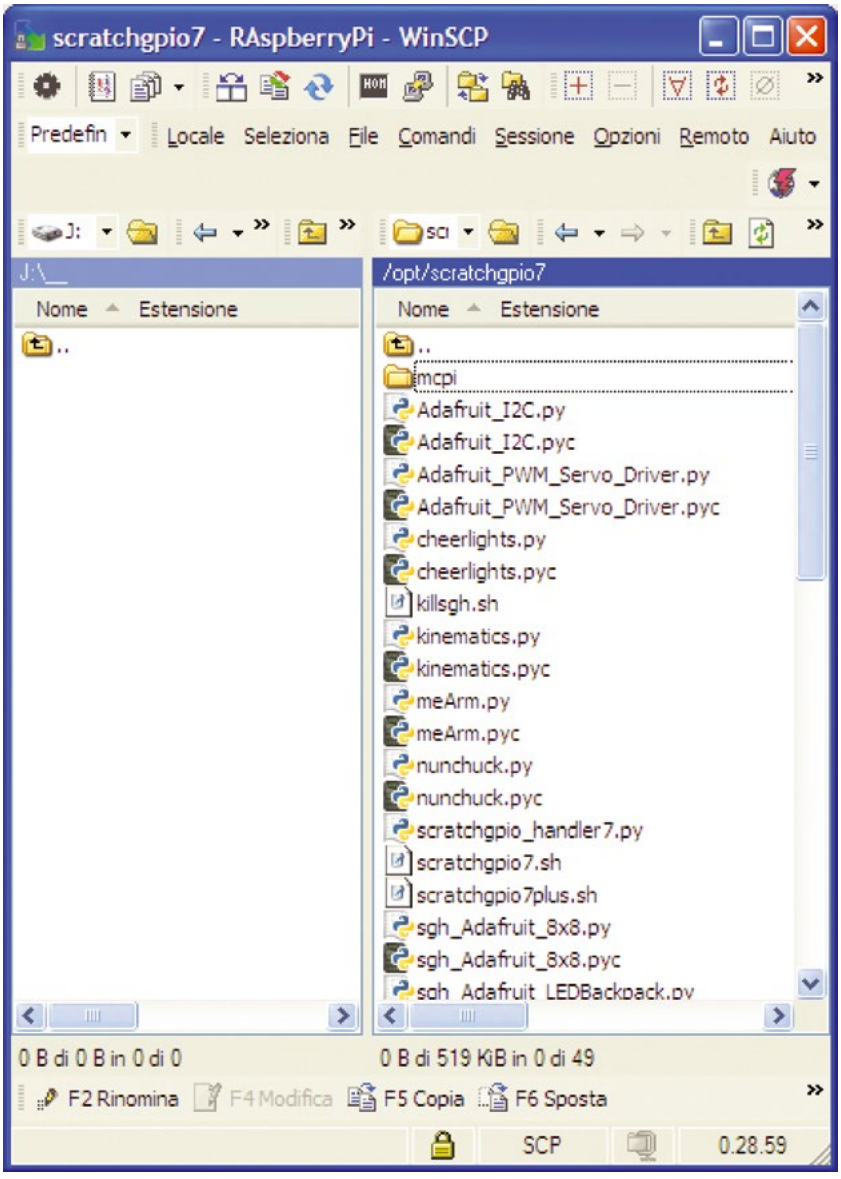
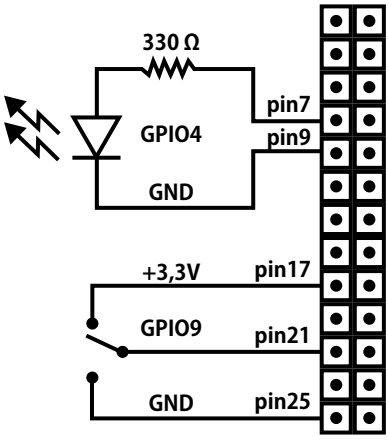



*Fig. 4*

***Fig. 5***




Una volta scaricato correttamente lo script di installazione lo possiamo eseguire con il comando (**Fig. 3**):

```
sudo bash isgh7.sh
```

Una volta terminato il processo di installazione troveremo tutti i file dell'applicazione installati nella cartella /opt/scratchgpio7 come visibile in **Fig. 4**, dove possiamo notare che l'estensione di Scratch per Raspberry PI è costituita da una gerarchia di programmi scritti in linguaggio Python. Sul desktop di Raspberry Pi troveremo due icone aggiuntive. Un'icona di nome ScratchGPIO7 che permette di fare esperimenti con semplici circuiti ed un'icona di nome ScratchGPIO7 Plus che contiene le configurazioni predefinite per poter interfacciare numerose schede add-on disponibili in commercio. Visto che parliamo di icone e di entrate nel menu di "start", ne approfittiamo per approfondire uno dei principali cambiamenti avvenuti nell'interfaccia grafica Lxde (Lightweight X11 Desktop Environment) di Raspberry Pi, con il rilascio della versione di Raspian datata 22-12-2014. Il rilascio successivo, 31-1-2015 è finalizzato al supporto della nuova versione 2 di Raspberry Pi, con il nuovo processore Cortex A7 quadricore ed un GB di memoria RAM, che la rende del tutto equivalente ad un piccolo PC desktop. A questa nuova versione di Raspberry Pi abbiamo dedicato un articolo di presentazione in questo stesso numero della rivista. Per inciso, tenete presente che al momento ScratchGPIO7 non "gira" sul nuovo Raspberry Pi 2 a causa della necessità di aggiornare alcune librerie dal quale dipende. Al momento dell'uscita della rivista il problema potrebbe essere già risolto. Tornando all'interfaccia grafica, la nuova versione prevede la barra delle applicazioni in alto sullo schermo, con il menu start in alto a sinistra. Questa soluzione è adottata da numerose distribuzioni GNU/Linux e sembra essere ispirata dalla somiglianza con il modo di utilizzare i libri tradizionali, che si scorrono dall'alto in basso e da sinistra a destra, almeno nella cultura occidentale, ed anche per uniformarsi alle applicazioni mobili, che hanno la barra delle notifiche in alto. La "scrivania" viene proposta vuota, con il solo "cestino" visibile in alto a sinistra. Tutte le altre icone possono essere posizionate a piacere sulla scrivania a partire dal menu start. Una volta individuata un'applicazione, è possibile portarne l'icona sulla scrivania, semplicemente, cliccandovi sopra col tasto destro del mouse e selezionando l'opzione "Add to desktop". Per spiegare il funzionamento delle nuove istruzioni presenti nel pacchetto ScratchGPIO7 usiamo il "solito" metodo del LED e dell'interruttore. Dunque realizziamo il piccolo circuito, lo schema del quale è visibile in **Fig. 5**.
Noi lo abbiamo realizzato semplicemente saldando la resistenza direttamente sul terminale positivo del LED e "allungando" i due reofori con due spezzoni di cavetto terminati con un connettore femmina ciascuno. Lo stesso per l'interruttore, nel nostro caso abbiamo trovato un modello con i terminali distanti 5 mm tra loro. Abbiamo saldato sui terminali uno spezzone di strip femmina a cinque poli, in modo da lasciare liberi i poli pari. Il componente così realizzato trova perfetta corrispondenza con i pin +3,3V, GPIO9, e GND di Raspberry Pi, corrispondenti rispettivamente ai pin fisici 17, 21 e 25. Collegheremo il terminale di massa del LED al pin fisico 9 ed il positivo al pin fisico 7, che corrisponde all'uscita GPIO4, anch'essi "vicini" tra di loro, come si può notare consultando lo schema del GPIO in **Fig. 6**. La foto dell'insieme è visibile in **Fig. 7**.
Per collegare i due componenti così realizzati è raccomanda-

| | | | P1 | | | | |
|---|---|---|---|---|---|---|---|
| <50mA | 3V3 | | 1 | 2 | | 5V | |
| BCM GPIO00/02 | SDA0/1 | 8 | 3 | 4 | | 5V | |
| BCM GPIO01/03 | SCL0/1 | 9 | 5 | 6 | | GND | |
| BCM GPIO04 | | 7 | 7 | 8 | 15 | TX | BCM GPIO14 |
| | GND | | 9 | 10 | 16 | RX | BCM GPIO15 |
| BCM GPIO17 | | 0 | 11 | 12 | 1 | PWM0 | BCM GPIO18 |
| BCM GPIO21/27 | | 2 | 13 | 14 | | GND | |
| BCM GPIO22 | | 3 | 15 | 16 | 4 | | BCM GPIO23 |
| <50mA | 3v3 | | 17 | 18 | 5 | | BCM GPIO24 |
| BCM GPIO10 | SPIMOSI | 12 | 19 | 20 | | GND | |
| BCM GPIO9 | SPIMOSO | 13 | 21 | 22 | 6 | | BCM GPIO25 |
| BCM GPIO11 | SPI SCLK | 14 | 23 | 24 | 10 | SPI CE0 N | BCM GPIO08 |
| | GND | | 25 | 26 | 11 | SPI CE1 N | BCM GPIO07 |

***Fig. 6***


*Fig. 7*

bile, per prima cosa, spegnere Raspberry Pi, poi collegare i due componenti, controllare attentamente i collegamenti e infine riaccendere il tutto. Lanciamo il desktop con il comando:

```
startx
```

che lancia il desktop grafico Lxde dove sono presenti le due icone di ScratchGPIO7 citate in precedenza (**Fig. 8**).
Apriamo l'ambiente di sviluppo ScatchGPIO7 e vediamo come è stata gestita l'interfaccia con i pin del GPIO di Raspberry Pi. Appena parte l'ambiente di sviluppo la configurazione predefinita prevede alcuni pin configurati come input ed altri come output. In particolare sono impostati come input i pin 22, 7, 3, 5, 24, 26, 19, 21, 23, 8 e 10 e come output i pin 11, 12, 13, 15, 16 e 18. Tenete presente che la numerazione utilizzata in ScratchGPIO7 fa corrispondere i pin alla loro posizione fisica sul connettore GPIO e non al nome di "sistema" del pin. Potete personalizzare la lingua nella quale sono presentate le istruzioni di ScratchGPIO7 cliccando sull'icona in alto a sinistra, a forma di piccolo "mondo" e scegliendo la lingua desiderata, come visibile in **Fig. 9**. Una delle istruzioni che permettono di interagire con i pin è la "invia a tutti" ("broadcast") visibile in **Fig. 10**. "Invia a tutti", in Scratch, è l'istruzione utilizzata per inviare messaggi ad altri script o, come in questo caso, a periferiche di sistema. Nello spazio "messaggio" dell'istruzione possiamo impostare lo stato dei pin inserendo "messaggi" costruiti con una sintassi particolare. Per portare

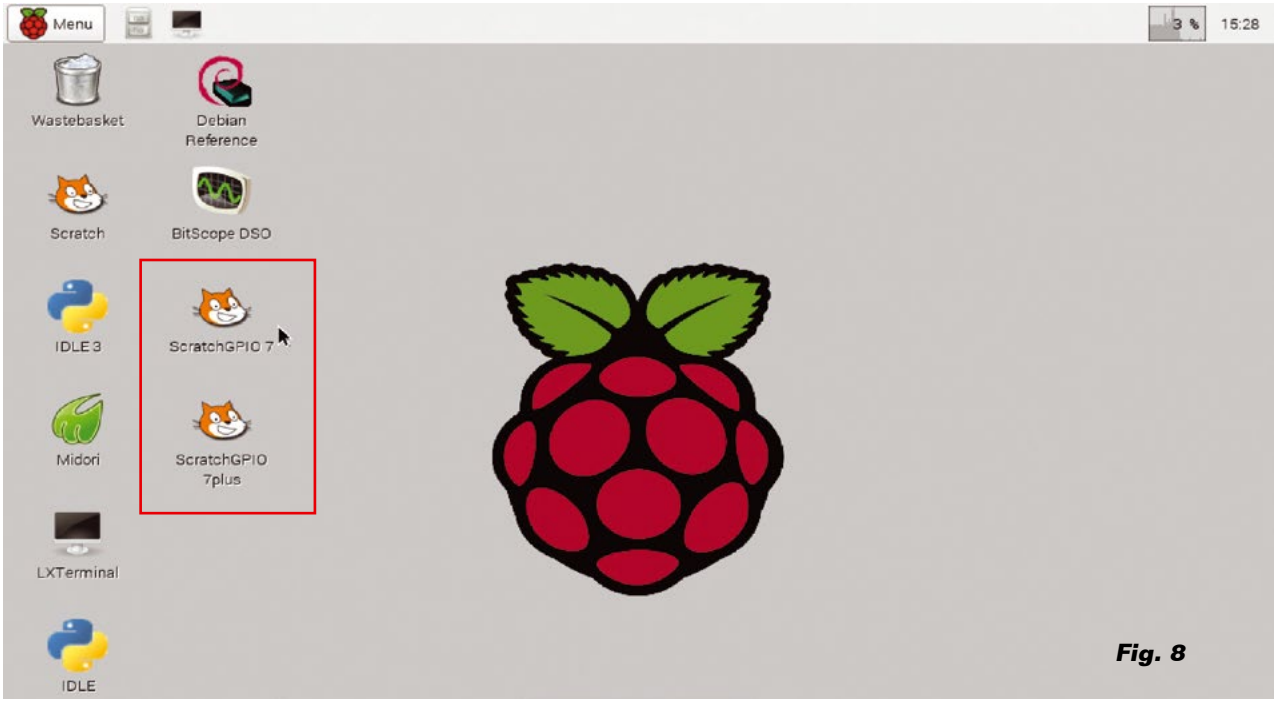



*Fig. 8*


***Fig. 9***

il pin 7 a livello alto utilizziamo il messaggio "pin7high" mentre per portarlo a livello basso utilizziamo il messaggio "pin7low". Non esiste controllo formale per i messaggi, quindi è necessario prestare la massima attenzione alla loro scrittura. In caso di errore, durante la composizione dello script, appare un messaggio di errore generico e, una volta lanciata l'esecuzione, il contorno dello script appare rosso invece che bianco.
In questi casi è molto utile la funzione di "aiuto", che si attiva "cliccando" sull'istruzione desiderata. È possibile anche alzare o abbassare i livelli di tutti i pin contemporaneamente con i messaggi "allon" e "alloff". Nello stesso messaggio possono essere concatenati più "ordini" come "pin7on pin5off".
In questo modo possiamo scrivere il nostro primo programma Scratch in grado di spegnere ed accendere un LED, il "listato" del quale è visibile in **Fig. 11**.
Sono validi anche i messaggi "pin7on" e "pin7off", come visibile nel "listato" di **Fig. 12**.
Con questo metodo potremmo riprodurre realmente il semaforo presentato nel primo articolo di questa serie, utilizzando tre LED, uno rosso, uno giallo ed uno verde. Piccola digressione. Un altro modo di interfacciare i pin è di creare delle variabili, nominarle in modo standard con i nomi dei pin come "pin26" o "pin19". In questo modo avremo a disposizione (**Fig. 13**):

- una variabile utilizzabile nel campo "variabile" delle istruzioni che lo prevedono;
- le istruzioni predefinite per l'impostazione dei valori da assegnare alle variabili;
- la possibilità di rendere visibile la variabile sul "palcoscenico".

Un piccolo accorgimento ma importante per evitare di perdere tempo a capire le cause di quello che sembra un malfunzionamento. Scratch, almeno fino alla versione attuale, "ricorda" lo stato di una variabile ed imposta un nuovo valore solo se questo è diverso da quello corrente. Quando si "clicca" sulla bandierina verde, Scratch invia solo i valori delle variabili che sono stati modificati rispetto alle esecuzioni precedenti, in pratica, le variabili che non hanno cambiato valore non vengono inviate e lo stato dei pin può apparire incongruente con ciò che ci aspettiamo.
Un modo per "girare attorno" a questo problema è quello di impostare le variabili all'inizio del programma, prima del primo ciclo "loop" a valori simbolici, anche di tipo non corrispondente ai dati attesi dalla variabile,


***Fig. 10***


***Fig. 11***


***Fig. 12***

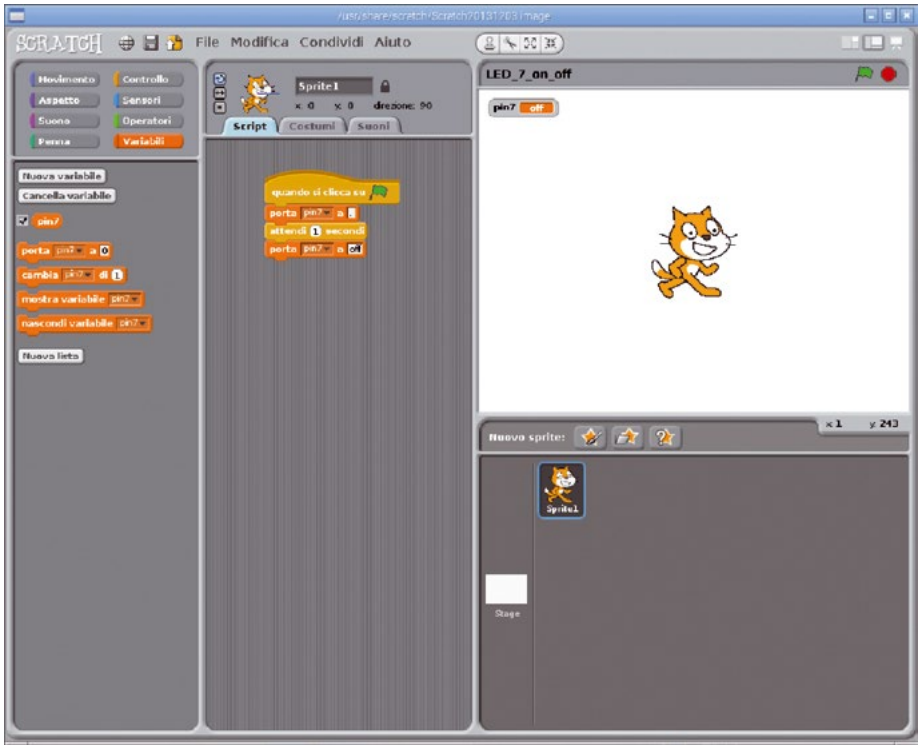
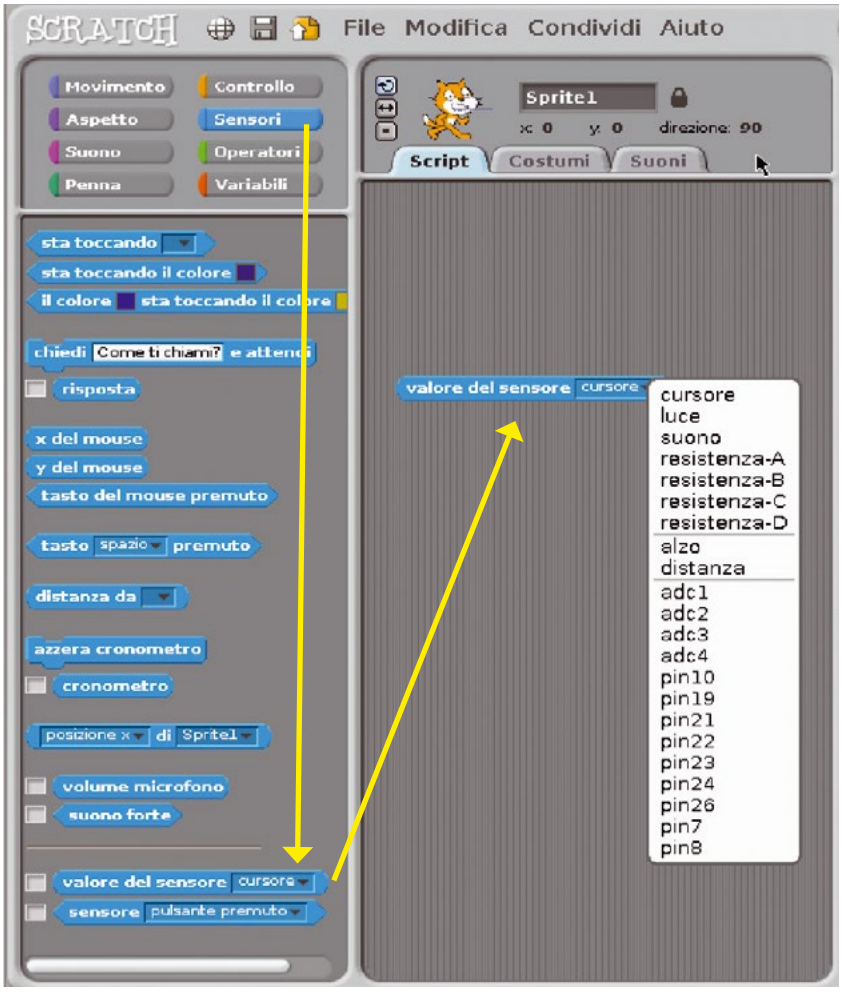



***Fig. 14***

dell'istruzione è di "intercettare" il cambiamento di stato, "0" quando il deviatore è collegato a massa e "1" quando collegato alla tensione di +3,3V.
Se per le vostre applicazioni avete bisogno di un numero di pin di output superiore a quello messo a disposizione nella configurazione predefinita, è sufficiente per esempio, eseguire un'istruzione "invia a tutti" ("broadcast") sul pin 10, impostato in modo predefinito come "input", per esempio "invia a tutti pin10on", per trasformarlo automaticamente in un pin di output. Il pacchetto ScratchGPIO7 comprende la possibilità di riconoscere e gestire un

per esempio con ".". Nel ciclo normale, poi, impostiamo i valori desiderati, come nell'esempio di **Fig. 14**.
Tornando al programma di accensione e spegnimento del LED, clicchiamo sulla bandierina per far partire l'esecuzione del programma. Inizialmente abbiamo collegato il terminale centrale del nostro deviatore al pin 21 e i terminali estremi rispettivamente a massa ed al pin +3,3V di Raspberry Pi (pin 25 e 17). Con questa configurazione possiamo realizzare un piccolo programma per verificare in pratica l'utilizzo delle istruzioni di gestione dei pin in ingresso. Andiamo nella sezione "Sensore" ed ispezioniamo le opzioni disponibili nell'istruzione "valore del sensore". Si può notare che sono indicati i pin 22, 7, 3, 5, 24, 26, 19, 21, 23, 8 e 10, ovvero quelli predefiniti come pin di ingresso, come mostrato in **Fig. 15**.
Creiamo un programma come quello visibile in **Fig. 17** dove l'istruzione centrale è quella indicata con la freccia. Scopo



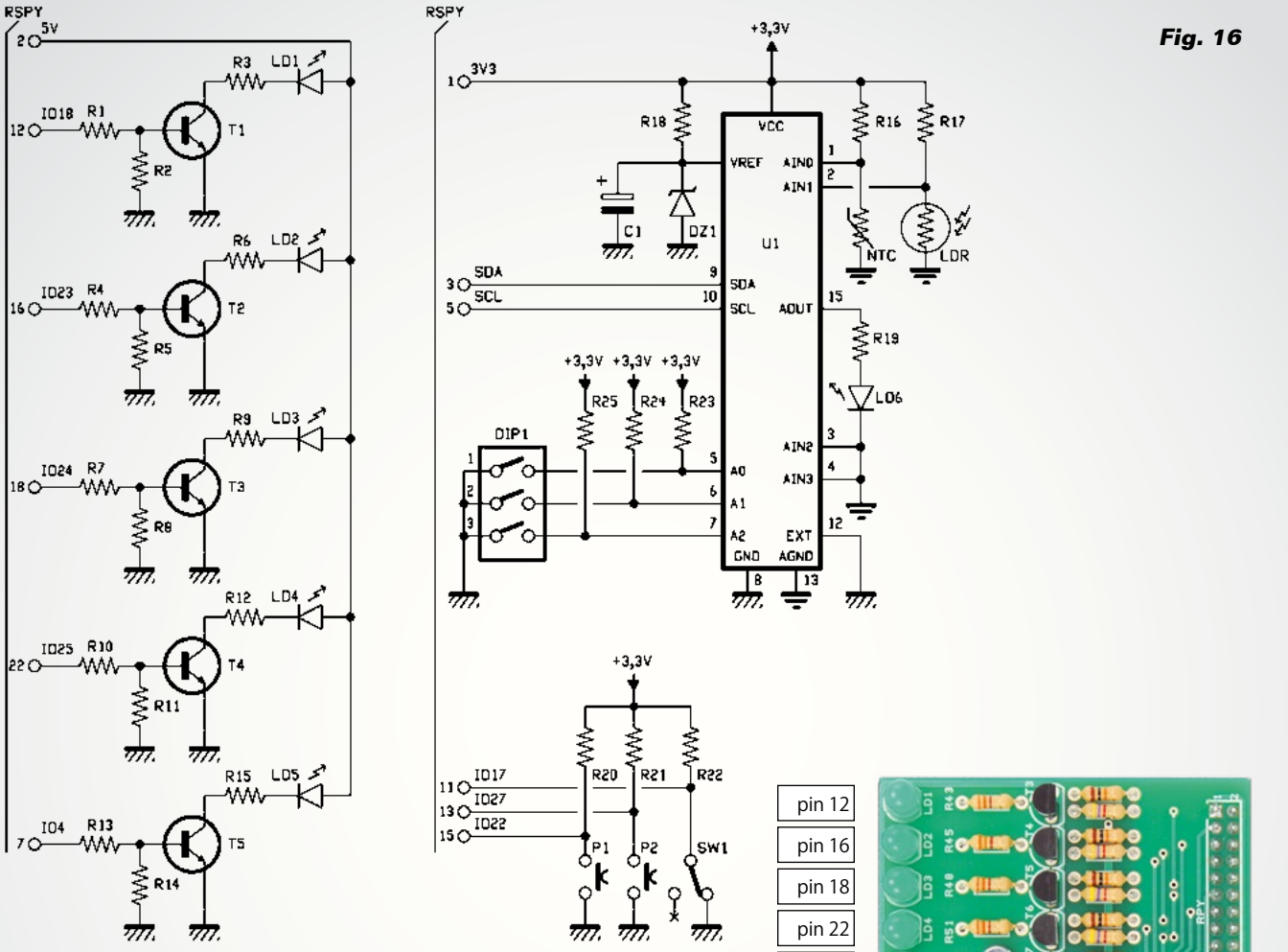



*Fig. 16*

## Elenco Componenti:

R1, R4, R7, R10, R13, R20÷R25: 4,7 kohm
R2, R5, R8, R11, R14: 10 kohm
R3, R6, R9, R12, R15, R19: 330 ohm
R16, R17: 15 kohm
R18: 100 ohm
C1: 4,7 $\mu$F 100 VL elettrolitico
DZ1: Zener 3,3V 400mW
T1÷T5: BC547
LD1÷LD5: LED 5 mm verde
LD6: LED 5 mm rosso
U1: PCF8591
P1, P2: Microswitch
SW1: Deviatore slitta verticale
DIP1: Dip-Switch 3 vie
NTC: NTC 10 kohm
LDR: Fotoresistenza 2-20 kohm

Varie:
- Zoccolo 8+8
- Strip Femmina 13x2 vie per Raspberry Pi
- Distanziale M/F 16mm
- Vite 8 mm 3 MA (2 pz.)
- Circuito stampato

certo numero di shield disponibili sul mercato. Una di queste permette a Raspberry Pi di acquisire dati analogici interfacciando l'integrato PCF8591 per mezzo del bus I2C, esattamente come avviene per la scheda FT1060, di corredo al libro "Raspberry Pi – Il mio primo Linux embedded". L'integrato PCF8591, come visibile nello schema di **Fig. 16** dispone di quattro ingressi analogico digitali a otto bit ed una uscita DAC, sempre ad otto bit. All'ingresso ADC1 dello shield è collegato un termistore (NTC nello schema) che possiamo utilizzare per misurare la temperatura ambiente. All'ingresso ADC2 è collegata una fotoresistenza (LDR nello schema) che possiamo


*Fig. 17*


*Fig. 18*

utilizzare per misurare la luminosità ambiente. Infine all'uscita DAC è collegato un LED rosso (LD6) che potremo far illuminare con un'intensità proporzionale al valore digitale impostato sull'uscita DAC stessa. Nel libro citato potete approfondire le caratteristiche dell'integrato PCF8591.
Per provare questa funzionalità, spegniamo Raspberry Pi con il comando:

```
sudo shutdown -h now
```

Smontiamo il LED ed il deviatore e posizioniamo la scheda FT1060 sul connettore GPIO di Raspberry Pi, facendo bene attenzione a far corrispondere i pin dello shield con quelli del microcomputer.
Dopo aver controllato tutto ancora una volta riaccendiamo Raspberry Pi ed attendiamo la

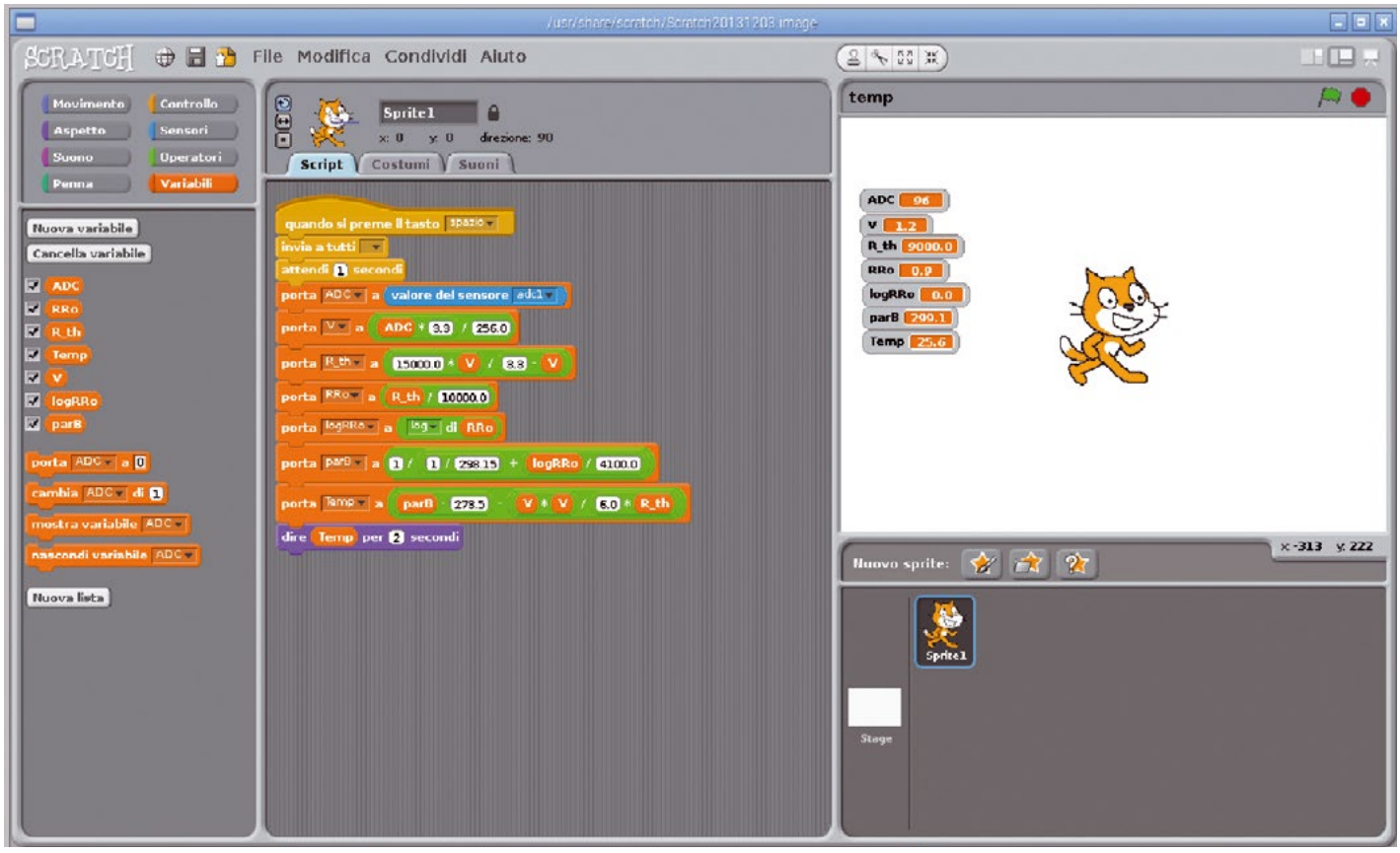
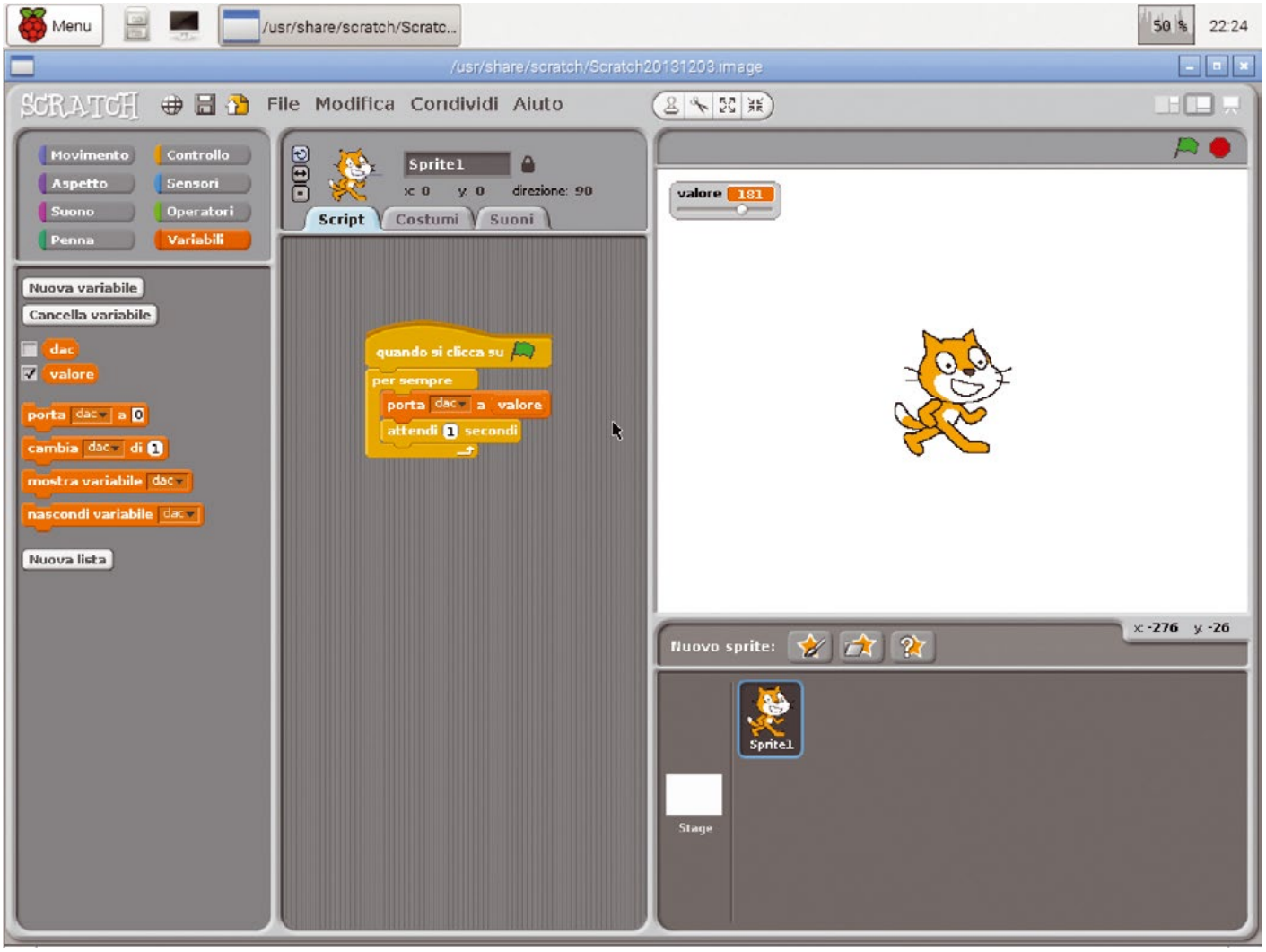


***Fig. 20***




fine del processo di boot. Ci connettiamo con l'utente generico "pi" e password "raspberry", lanciamo l'interfaccia grafica con il comando "startx" ed il programma ScratchGPIO7. Prima però verifichiamo se abbiamo attivato il driver di gestione del bus I2C.


Se così non fosse, e stiamo utilizzando le versioni di Raspbian più recenti, apriamo una finestra di terminale e lanciamo il configuratore raspi-config. Selezioniamo la voce di menu "I2C" e nelle pagine di configurazione successive rispondiamo "yes" a tutti i passaggi. Se ci viene chiesto di riavviare il microcomputer rispondiamo affermativamente e attendiamo il riavvio.
Nel nostro caso è stato necessario attivare il bus I2C anche nel modo tradizionale, commentando l'entrata "# blacklist i2c-bcm2708" nel file /etc/modprobe.d/raspi-blacklist.conf, dare il comando modprobe i2c-dev e installare il pacchetto "i2c-tools". Altro riavvio per poi riportarci nell'applicazione ScratchGPIO7. Ora proviamo a leggere il valore del sensore di temperatura collegato all'ingresso ADC1 dell'integrato PCF8591. Per eseguire la lettura utilizziamo l'istruzione "invia a tutti ADC1".
Per acquisire il risultato utilizziamo la "variabile" "valore del sensore adc1", come potete vedere nel programma di **Fig. 18** e nella vista d'insieme di **Fig. 19**, dove abbiamo anche riprodotto in Scratch le operazioni matematiche per calcolare la temperatura a partire dal valore ADC, con il metodo del "parametro B" descritto numerose volte nella

rivista e nel libro citato all'inizio.
Per impostare l'uscita DAC con un valore digitale da convertire in una tensione analogica proporzionale al valore impostato si utilizza l'istruzione "porta dac a <valore>" dove <valore> è un numero compreso tra "0" e "255".
Il programma visibile in **Fig. 21** permette di impostare la luminosità del LED rosso proporzionalmente alla posizione del cursore (**Fig. 20**).
Per i nostri esperimenti, della scheda FT1060, con riferimento allo schema di Fig. 16, possiamo utilizzare, oltre ai convertori ADC e DAC, anche i seguenti pin:

- Come Uscite possiamo utilizzare i pin 12 (GPIO18), pin 16 (GPIO23), pin 18 (GPIO24), pin 22 (GPIO25) e pin 7 (GPIO4). Ciascuna di queste uscite è provvista di un LED pilotato da un transistor configurato ad emettitore comune per evitare di sovraccaricare i pin del GPIO.
- In ingresso abbiamo a disposizione due pulsanti collegati ai pin 13 (GPIO27) e pin 15 (GPIO22).
  Un deviatore è collegato al pin 11 (GPIO17).
  Prima di utilizzare questi pin ricordiamo di impostarli come pin di ingresso con istruzioni del tipo "invia a tutti Config11In".

In ScratchGPIO7 l'istruzione "invert"permette di invertire la logica di comportamento di un pin, per esempio riconoscere il pin come On quando il livello di tensione è basso e viceversa.
L'istruzione "invia a tutti Config11In" permette di impostare in modo esplicito il pin 11 da output ad input. L'istruzione "invia a tutti GetIP" permette di "leggere" l'indirizzo IP di Raspberry Pi, che può poi essere evidenziato con l'istruzione "dire" e la variabile "valore del sensore ipaddress" come visibile in **Fig. 22**.