

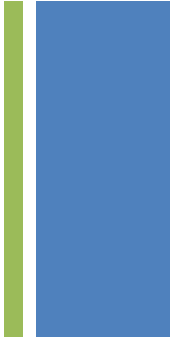
Corso Java

Da Hello World ad accendere una lampada con Android

Giornata 3°

7 Novembre 2015

+ Programma di oggi



■ **Giornata 3 – Sabato 7 novembre 2015 – dalle 9 alle 18**

Java

- Funzioni per le stringhe e per le liste in Java
- Programma libretto
- Strumenti di grafica in Java, attraverso un gioco

Android

- Che cos'è Android e perché è bello e “facile” programmare applicazioni per smarthphone.
- Spiegazione di un progetto per Android.
- Interazione tra utente e Android, attraverso gli strumenti grafici disponibili.
- Esempi di applicazioni domotiche con Arduino: accendere/spegnere una lampada da remoto con un tap.



+ Parte Java



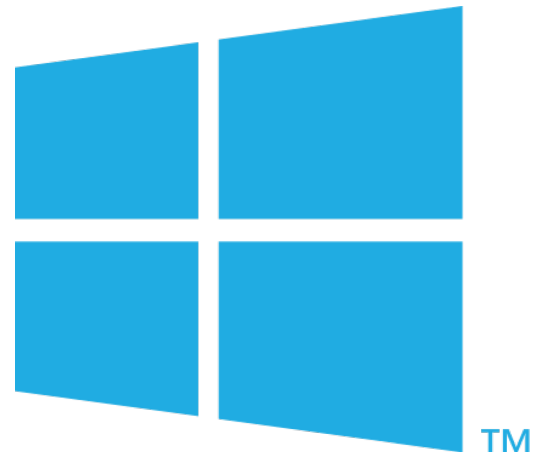
- **Programma Stringhe:** vedremo un po' di funzionalità presenti in Java per gestire queste variabili. Vedremo anche le liste di oggetti, che sono molto utili
- **Programma Libretto:** faremo un programma in grado di calcolare la media dei voti di scuola
- **Programma Gioco:** ci divertiremo in un piccolo gioco, ma molto divertente, che permette di mettere un po' di colori nelle nostre applicazioni
- **Programma Lampada:** faremo un programma in grado di accedere una lampada



+ Smartphone



- Oggi parleremo di un argomento molto interessante: la programmazione di applicazioni per dispositivi mobile
- Ormai lo smartphone è dispositivo che tutti hanno, con il quale è possibile di tutto
- Le prestazioni diventano sempre più simili a quelle di un computer





Linguaggi di programmazione



- Android: Java
- iOS: Swift
- Windows Phone: C#
- Perché questo corso ha scelto di usare Android come piattaforma ?
 1. È il linguaggio più supportato in rete (tantissime guide)
 2. È gratuito
 3. Permette di raggiungere il maggior numero di consumatori





Il linguaggio più supportato in rete



- Google mette a disposizione tantissime guide che spiegano come usare le librerie già presenti, per poter sfruttare i componenti hardware dei vari smartphone
- Ci sono tantissime guide/materiale in rete, basta digitare sui motori di ricerca le funzioni richieste
- Molti sviluppatori mettono a disposizione dei tutorial step-by-step, su come realizzare un'applicazione



+ Android è “gratuito”



- Eclipse permette di esportare in file .apk il programma che viene realizzato e può essere condiviso sul web per poter essere installato su tutti i dispositivi
- Per pubblicarlo invece sullo store e avere quindi tutti i servizi/vantaggi di tale struttura, è necessario pagare 25 \$ una sola volta (un account iOs costerebbe più di 70 Euro all'anno)





Il maggior numero di utenti



- Android è un sistema operativo che è presente in tantissimi smartphone di diverse aziende: Samsung, LG, Motorola etc...
- Il bacino di utenti è davvero ampio
- Non è necessario scrivere del codice per ogni dispositivo, ma viene fatto tutto in automatico





Come si sviluppa in Android ?



- Si può usare Eclipse con il plugin di Android Development Kit
- In alternativa Google mette a disposizione Android Studio, che è un'applicazione molto simile ad Eclipse, con il supporto nativo alla sviluppo per Android
- Per comodità, durante il corso useremo la prima soluzione
- A casa potrete provare ad usare l'altra versione, per provare le differenze che ci sono





Sommario Android intro

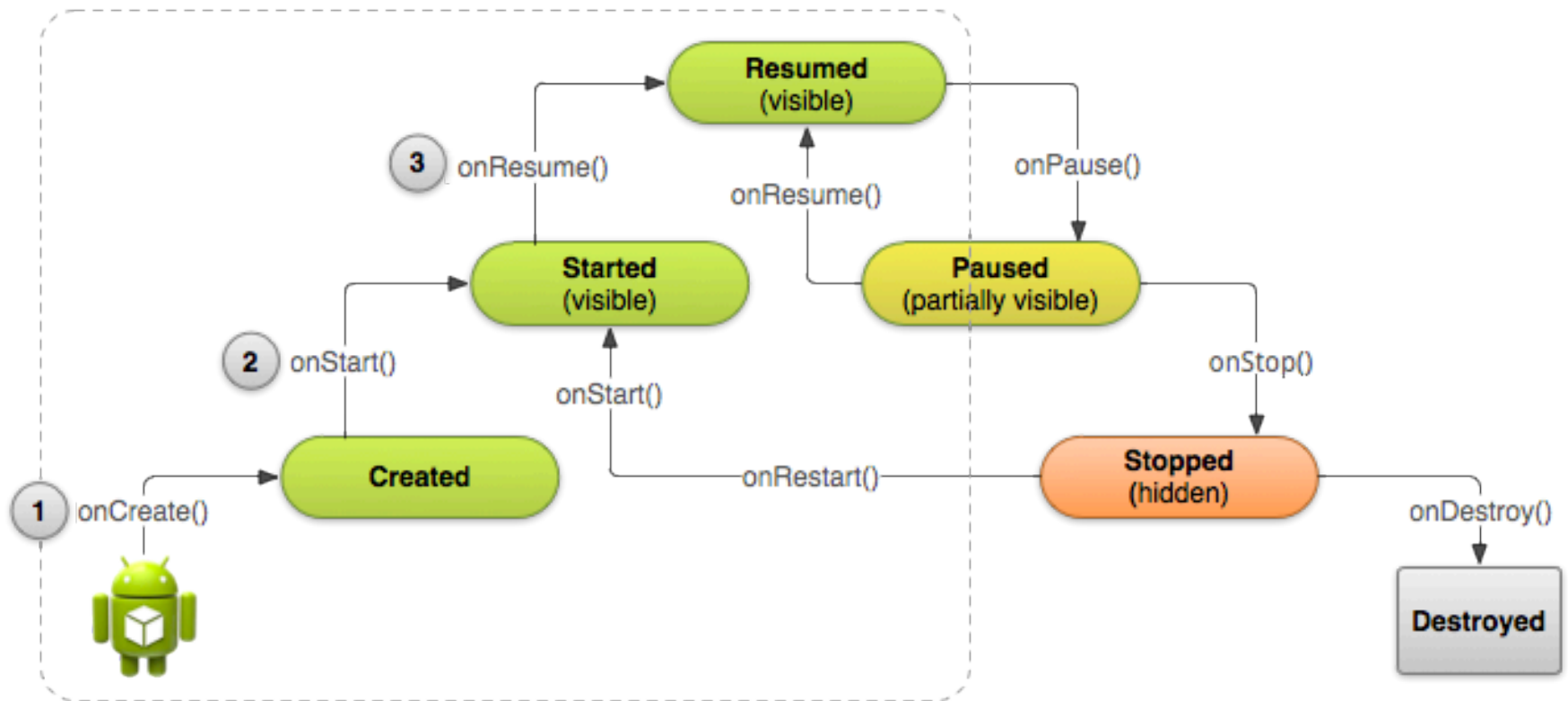


- Android è quindi un linguaggio di programmazione veramente all'avanguardia
- Permette di realizzare applicazioni di ottimo livello
- È supportato con tantissimo materiale

Ha qualche difetto ?

- *Si, ma per fortuna sono davvero pochi*
- *Alcune funzionalità richiede di scrivere un po' troppo codice*
- *Per poter essere testata correttamente un'applicazione, richiede un dispositivo Android, in quando l'emulatore presente nel PC è davvero lento e privo di molto funzionalità*

+ Thread in Android



+ HelloWorld in Android



- In questo programma verrà mostrata la scritta *Ciao Mondo !* sulla schermata del nostro smartphone
- Siete pronti a programmare ?
- In realtà Eclipse è troppo gentile e ci farà lui il lavoro per noi
- Adesso vedremo...




Come creare un progetto Android



New Android Application

New Android Application
Creates a new Android Application



Application Name:

Project Name:


Package Name:


Minimum Required SDK:

Target SDK:

Compile With:

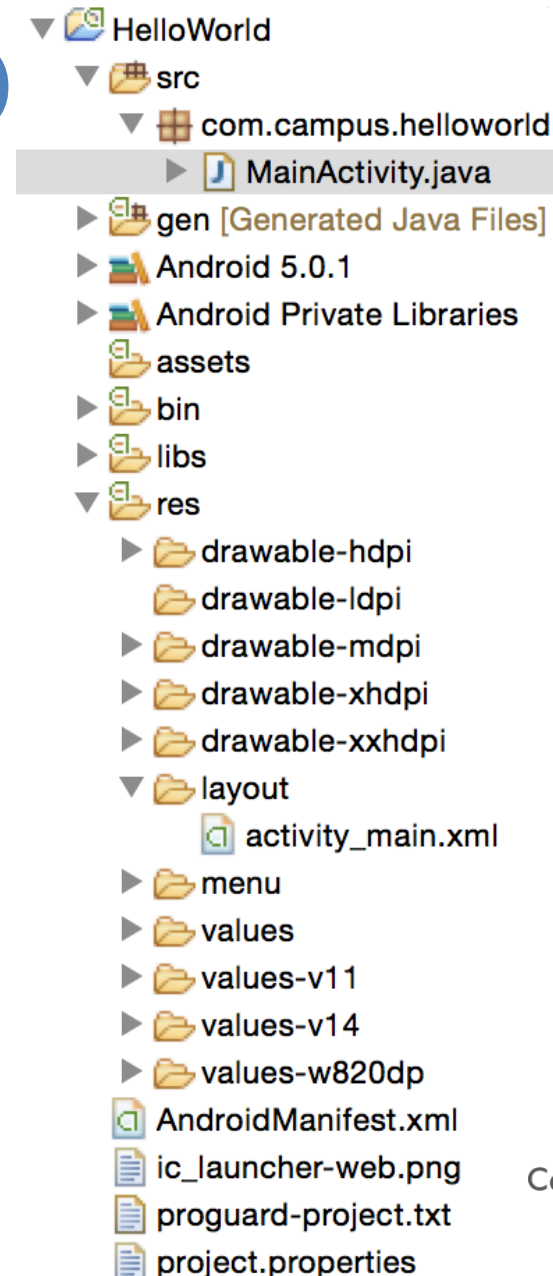
Theme:

 The package name must be a unique identifier for your application. It is typically not shown to users, but it *must* stay the same for the lifetime of your application; it is how multiple versions of the same application are considered the "same app". This is typically the reverse domain name of your organization plus one or more application identifiers, and it must be a valid Java package name.

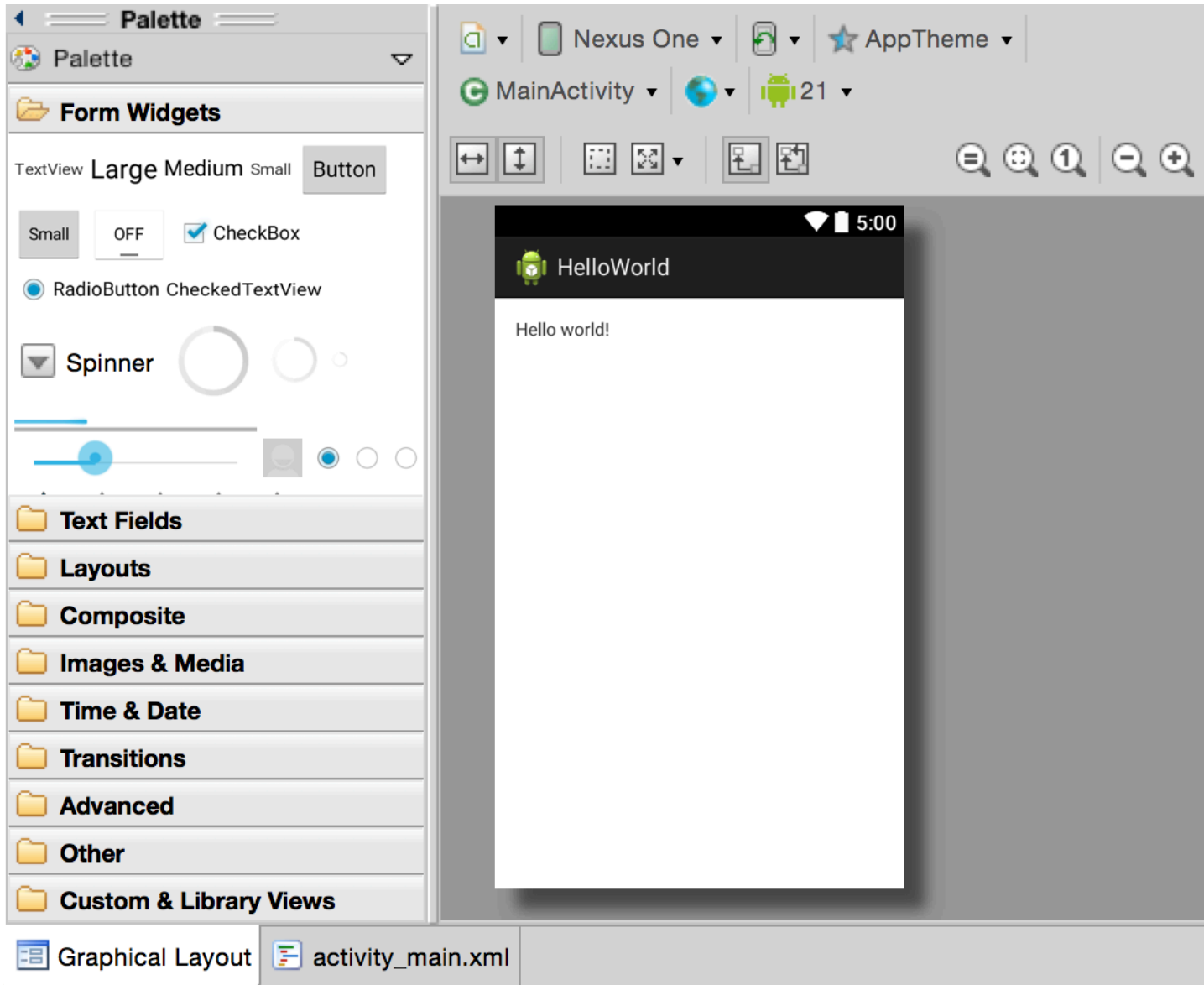


+ Progetto Android(1)

- Quando viene creato un progetto di Android, Eclipse genera tanto materiale in modo automatico



+ Progetto Android(2)



+ Progetto Android(3)

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```



+ What's your name ?



- Android viene utilizzato in tutto il mondo e quindi in diversi paesi, con le lingue diverse
- Esiste un modo semplice per realizzare applicazioni con diverse lingue ?
- La risposta è sì e in Android è davvero semplice !
- Il trucco è davvero semplice: al posto di scrivere nel nostro programma “Hello World”, scriviamo un tag-id, in modo da essere univoco.



Esempio pratico



strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

it.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">CiaoMondo</string>
    <string name="hello_world">Ciao mondo!</string>
    <string name="action_settings">Impostazioni</string>
</resources>
```



+ Compito



- *Compito: Creare il supporto per le lingue Francese, Spagnolo*

(Suggerimento1: se non sapete le traduzioni, utilizzate Google Translate)

(Suggerimento2: Francia->fr, Spagna->es)



+ Programma HelloUser



- Nel programma precedente, non c'è troppa interazione
- In pratica il risultato mostrato è sempre lo stesso e l'utente non interagisce mai
- Nel prossimo programma, vedremo come utilizzare un po' di "controller" per creare interazioni tra l'utente e lo smartphone
- Ad esempio, l'utente inserisci il proprio nome e riceverà un saluto





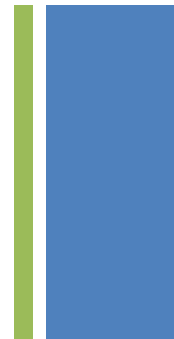
Componenti



TextView
EditText
Button



+ Compito



- *Compito: Creare il supporto per le lingue Italiano, Francese, Spagnolo*

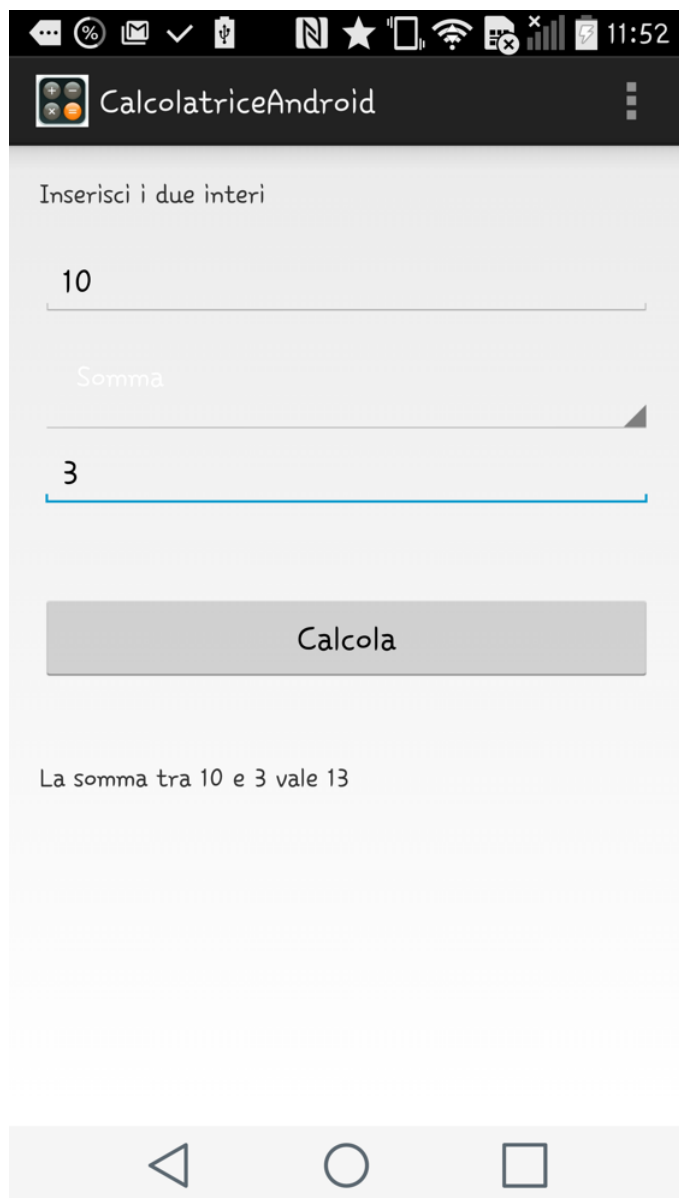
+ Calcolatrice per Android



- In questo programma, dovremmo realizzare una calcolatrice, che permette di effettuare le, ormai, solite operazioni
- Dovremmo cercare di sfruttare il codice che abbiamo già realizzato per il progetto della seconda giornata
- Per facilitare l'utente, verrà mostrata già una lista delle operazioni disponibili



+ Risultato del progetto



Inserisci i due interi

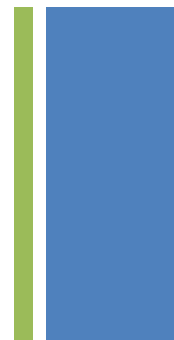
10

Somma

3

Calcola

La somma tra 10 e 3 vale 13



+ Compito



- *Compito: Creare il supporto per le lingue Italiano, Francese, Spagnolo*
- *Compito: aggiungere l'operazione che permette di trovare il minimo tra i due numeri inseriti*
- *Compito: cercare di far cambiare il colore bianco della lista delle operazioni (come vi ho già detto, il web è un'ottima risorsa)*





Introduzione ad un'applicazione



- Scommetto che sicuramente avrete già installato un'applicazione dallo store
- Quelle professionali hanno una intro all'applicazione, cioè un piccolo percorso guidato che permette all'utente di capire il funzionamento dell'applicazione ed effettuare le impostazioni iniziali richieste
- In tale modo l'utente è più contento e capisce meglio il funzionamento dell'applicazione



+ Progetto Introduzione



- Vedremo un semplice esempio di introduzione all'applicazione vera e propria, che ci serverà per il progetto della lampada
- Utilizzeremo per la prima volta i Fragment, che altro non sono che delle piccole Activity
- Con dei semplici swipe, sarà quindi possibile sfogliare le pagine della nostra introduzione



+ Progetto iLamp



- Finalmente siamo quasi arrivati all'obiettivo del corso, cioè riuscire a comandare una lampada da remoto dal nostro smartphone Android
- È stato un percorso dove, spero, abbiate imparato molto riguardo al mondo della programmazione e in particolare Java, Android
- Ora potremmo finalmente stupire i nostri amici, genitori
- In realtà, dal momento che sarà necessario lavorare a 220 V, il dispositivo hardware di Arduino dovrà essere realizzato da un adulto, in grado di poterlo fare !



+ Sistema



Parte hardware

1. Arduino Uno + Scheda di Rete
2. Un relè
3. Una lampada

Parte software

1. Programma iLamp

+ Arduino (1)



- Che cos'è Arduino ?
- Arduino è un piccolo PC, in grado di essere programmato per poter lavorare con dei sensori (ad esempio termostati, sensori di presenza etc...)
- In questo caso verrà utilizzato con un relay, che altro non è che un interruttore, in grado di far passare corrente solo quando vogliamo noi

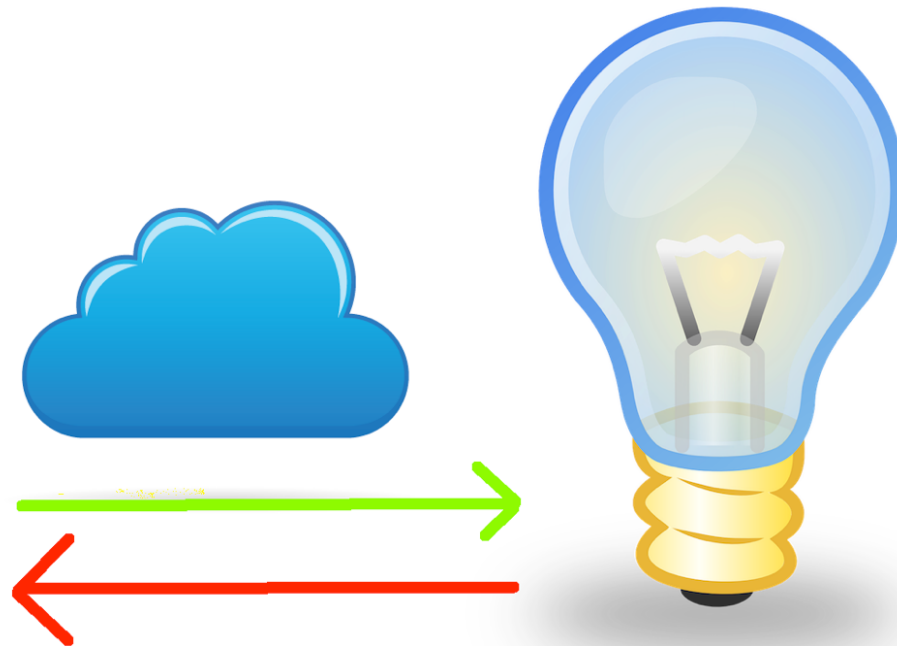


+ Arduino (2)



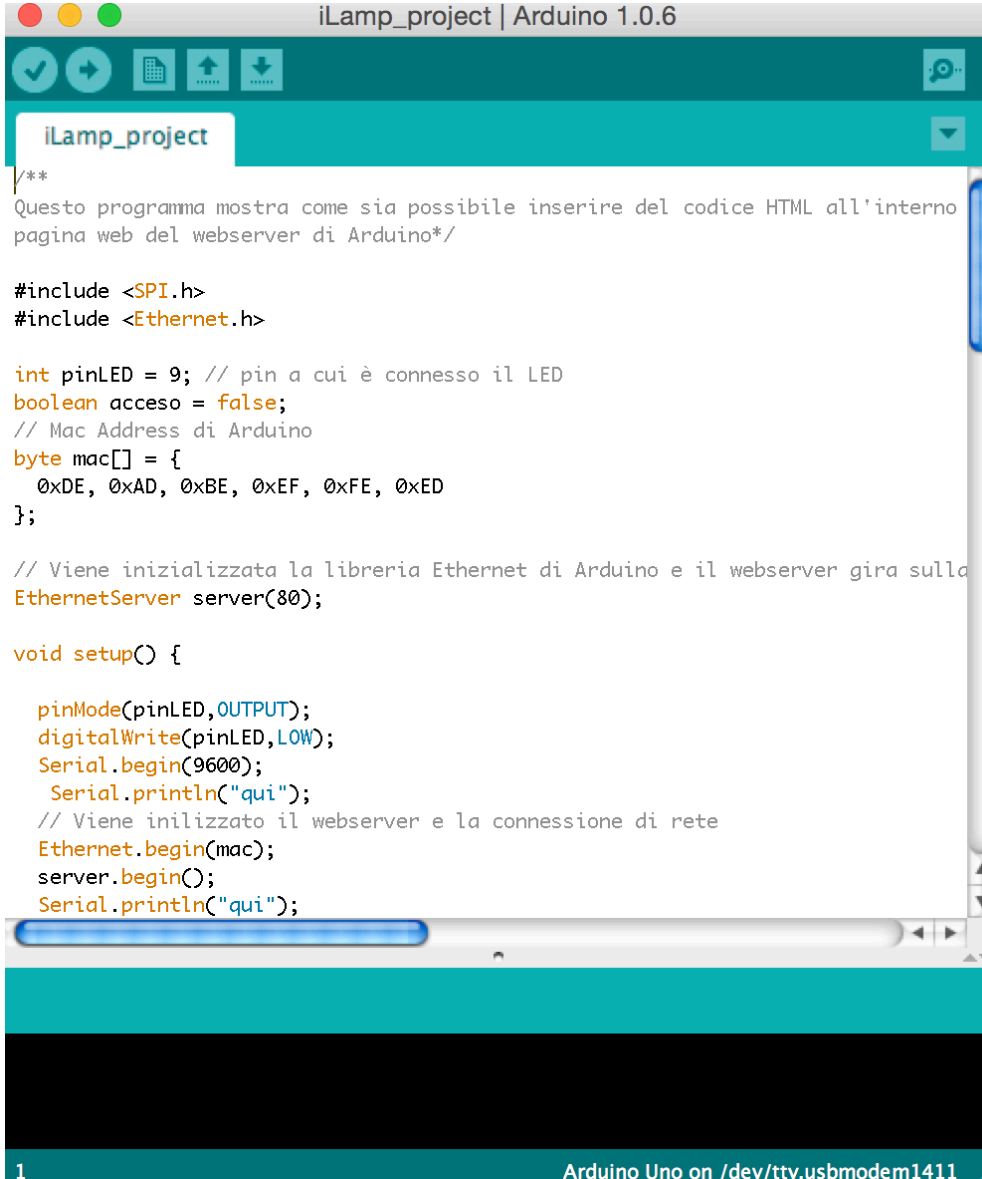
- La scheda di rete di Arduino, permette di farli ricevere comandi da remoto sfruttando la connessione ad Internet
- Ma come si programma Arduino ?
- Arduino si programma in C, uno dei linguaggi più famosi
- Non dovrete scrivere il codice, dal momento che ho già fatto io il lavoro per voi
- Ci manca da fare la parte legata alla comunicazione, utilizzando Java

+ Arduino (3)





Arduino (4)



The screenshot shows the Arduino IDE window titled "iLamp_project | Arduino 1.0.6". The code editor contains the following C++ code:

```
/**
Questo programma mostra come sia possibile inserire del codice HTML all'interno
pagina web del webserver di Arduino*/

#include <SPI.h>
#include <Ethernet.h>

int pinLED = 9; // pin a cui è connesso il LED
boolean acceso = false;
// Mac Address di Arduino
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};

// Viene inizializzata la libreria Ethernet di Arduino e il webserver gira sulla
EthernetServer server(80);

void setup() {

  pinMode(pinLED, OUTPUT);
  digitalWrite(pinLED, LOW);
  Serial.begin(9600);
  Serial.println("qui");
  // Viene inilizzato il webserver e la connessione di rete
  Ethernet.begin(mac);
  server.begin();
  Serial.println("qui");
}
```

The status bar at the bottom indicates "1" on the left and "Arduino Uno on /dev/tty.usbmodem1411" on the right.





Come comunicare con Arduino?(1)



- Abbiamo detto che utilizziamo la rete Internet
- Serve un protocollo di comunicazione
- TCP, HTTP

Che cosa sono ?

- Magari non lo sapete, ma ogni volta che navigate sul web li utilizzate





Come comunicare con Arduino?(2)



- Il programma che è stato realizzato, permette di comunicare in maniera facile
- **Accendere la lampada:** [http://IP DI ARDUINO/?on](http://IP_DI_ARDUINO/?on)
- **Spegnere la lampada:** [http://IP DI ARDUINO/?off](http://IP_DI_ARDUINO/?off)
- Dovremmo realizzare un programma in Java, che permetta di svolgere le medesime richieste che facciamo quando navighiamo nel web



+ Proviamo prima in Java

```
public String sendCommand(String ip, String command) {
    try {
        URL url = new URL("http://" + ip + "/" + command);
        connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("POST");
        connection.setDoInput(true);
        connection.setDoOutput(true);
        InputStream is = connection.getInputStream();
        BufferedReader rd = new BufferedReader(new InputStreamReader(is));
        String line;
        StringBuffer response = new StringBuffer();
        while ((line = rd.readLine()) != null) {
            response.append(line);
        }
        rd.close();
        return response.toString();
    } catch (Exception e) {
        return "Errore nella comunicazione";
    }
}
```



+ Passiamo ad Android



- Possiamo sfruttare la parte introduttiva che abbiamo realizzato precedentemente
- Ora dobbiamo focalizzare l'attenzione su due parti ben importanti:
 1. Il salvataggio dell'IP all'interno del nostro smartphone
 2. La comunicazione tra Android e Arduino



+ Salvataggio dei dati (1)



- In Android ci sono diverse soluzioni:
 1. All'interno delle impostazioni (semplice)
 2. Utilizzando un DB (un po' più complessa e inutile)
- Utilizzeremo la prima, dal momento che dovremmo salvare solamente l'indirizzo IP di Arduino, che è una variabile di tipo String

+ Salvataggio dei dati (2)



- **private SharedPreferences prefs;**
- **prefs=getApplicationContext().getSharedPreferences("introduzione", Context.*MODE_PRIVATE*);**
- **prefs.edit().putString("ip",ip).apply();;**
- **prefs.edit().putBoolean("intro", **false**).apply();**
- **String ip = prefs.getString("ip", "");**





Comunicazione con Arduino (1)

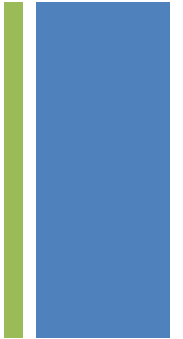


- Occorre utilizzare il protocollo HTTP e la sintassi vista precedentemente
- Possiamo sfruttare il codice che abbiamo già scritto in Java ed utilizzarlo in Android, ma con ulteriori precisazioni
- In Android, per utilizzare qualsiasi tipo di dispositivo (ad esempio fotocamera, scheda Wireless) è necessario chiedere l'autorizzazione all'utente, nel file `AndroidManifest.xml`
- In aggiunta, Android non permette di utilizzare un'operazione nello stesso Thread che viene utilizzato per gestire l'interazione tra utente e la grafica





Comunicazione con Arduino (2)



```
public class CommunicationTask extends AsyncTask<String, Void, String> {  
  
    private ProgressDialog pDialog;  
    private Context context;  
  
    public CommunicationTask(Context context) {  
        super();  
        this.context = context;  
    }  
  
    @Override
```

```
protected void onPreExecute() {  
    super.onPreExecute();  
    pDialog = new ProgressDialog(context);  
    pDialog.setMessage("Caricamento...");  
    pDialog.setIndeterminate(false);  
    pDialog.setCancelable(true);  
    pDialog.show();  
  
}
```





Comunicazione con Arduino (3)



```
@Override
protected String doInBackground(String... params) {
    if (Connection.getConnection().isNetworkAvailable(context)) {
        String ip = params[0];
        String command = params[1];
        return Connection.getConnection().sendCommand(ip, command);
    } else {
        return "Verifica di essere connesso and Internet";
    }
}

@Override
protected void onPostExecute(String result) {
    pDialog.dismiss();
    Toast.makeText(context, result, Toast.LENGTH_LONG).show();
    super.onPostExecute(result);
}
```





Comunicazione con Arduino (4)



```
private void actionButton(ImageButton button, final String command){  
    button.setOnClickListener(new OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            String ip = prefs.getString("ip", "");  
            CommunicationTask task = new CommunicationTask(HomeActivity.this);  
            task.execute(ip,command);  
        }  
    });  
}
```

