INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

FORMATION

E-Blocks wifi controlled by an Android device

Benoit Pierret – benoit.pierret@insa-lyon.fr
Nicolas Stouls – nicolas.stouls@insa-lyon.fr
Yann Ricotti – yann.ricotti@insa-lyon.fr

membre de
UNIVERSITÉ DE LYON

INSA LYON

Architecture

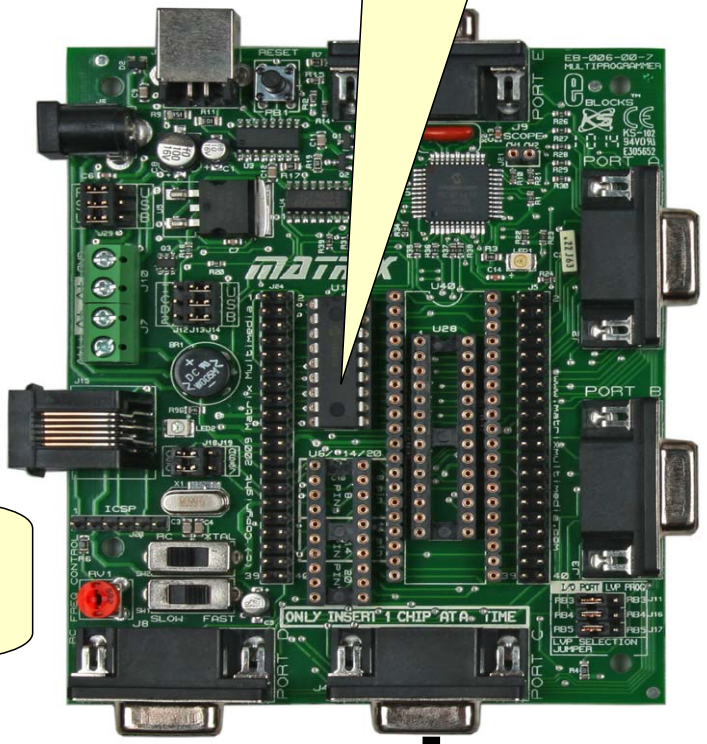
This solution is simple because:

- Nothing to know about the wifi protocol. The data are simply copied from wifi to RS232 with EB069 E-Block
- The EB069 is a server: nothing to configure in your network
- RS232 is easy to program and simulate with Flowcode
- Wifi is natively supported by Android. No library to import. A packet can be send in a single function

Android program

WIFI

Wifi configuration



Flowcode program

RS232

What do you need ?

- Hardware:
  - Android device
  - Matrix E-Block USB PICmicro multiprogrammer board (EB006)
  - Matrix E-Block Wireless (EB069)
  - Anything else you want

- Software:
  - Free Android development tool chain:

    http://developer.android.com/sdk/installing.html
  - Flowcode:

    http://www.matrixmultimedia.com/flowcode.php

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

FORMATION

A Full Example : Remote Controled LED

membre de
UNIVERSITÉ DE LYON

INSA LYON

What do we want to do ?

10 minutes to have your own

- Part 1: E-Blocks side - 3 steps / 5 minutes
  - E-blocks assembly
  - Flowcode program
  - Setting up the Wi-Fi E-Block
- Part 2: Android device side - 4 steps / 5 minutes
  - Starting a new Eclipse Android project
  - Building the Human Machine Interface (HMI)
  - Java part : Binding an action to the on/off button
  - Requesting for network permissions

Part 1: E-Block

E-Block assembly

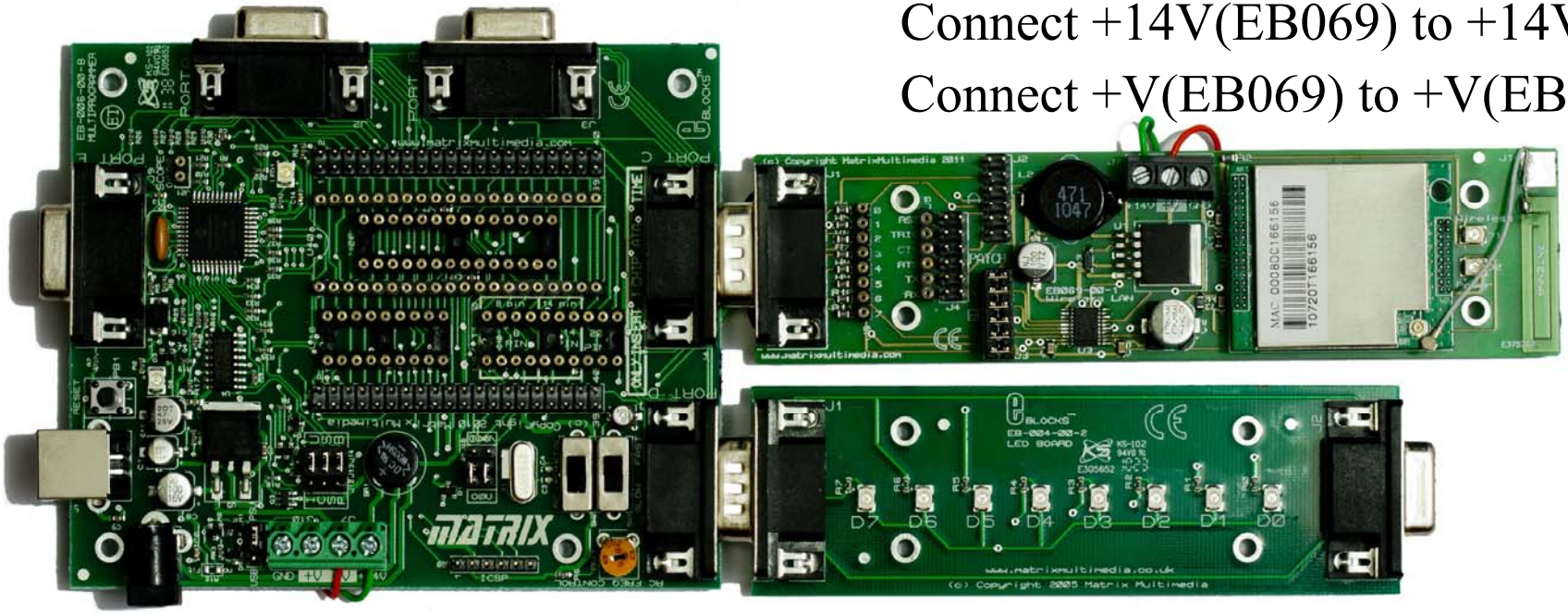
Multiprogrammer EB006 with :

- PICmicro : 16F887 or 16F877
- Port D : LED EB004
- Port C : WLAN EB069

Jumper B position

Connect +14V(EB069) to +14V(EB006)

Connect +V(EB069) to +V(EB006)

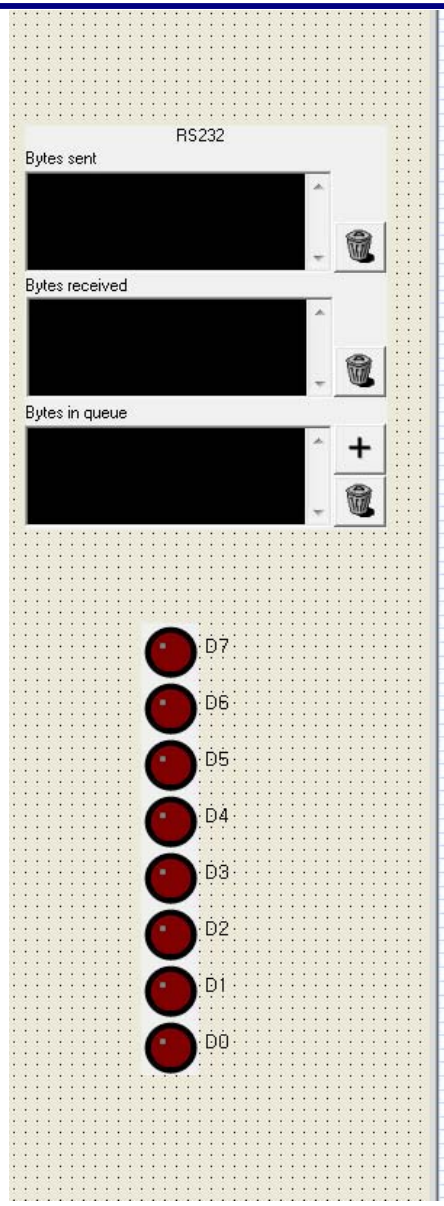


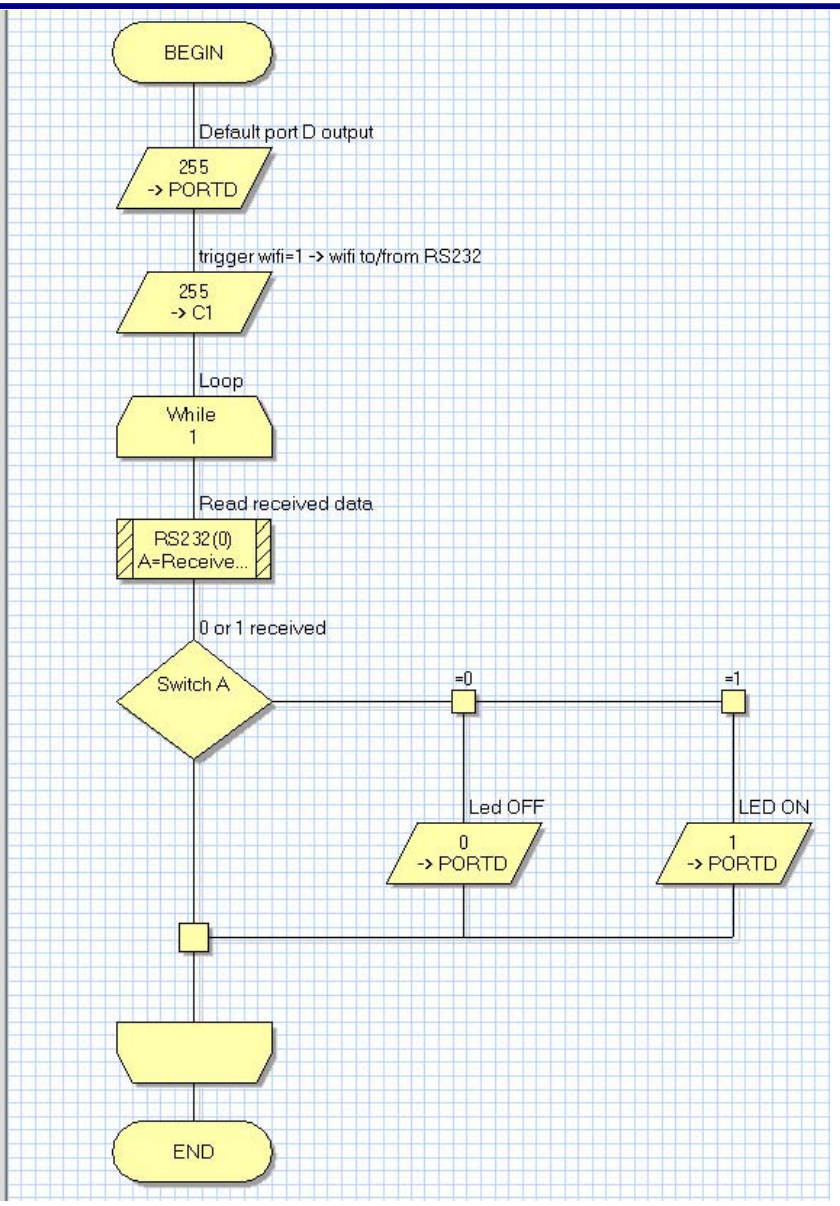
Flowcode program

- How it works :
  - Wifi data are reading from the RS232
  - The LED is switched on/off with 1/0 data

- RS232 properties :
  - BAUD rate 38400
  - Display bytes for simulation with numbers

It is possible to do it with less steps but it would be ridiculous...





Setting up the Wi-Fi EB069 E-Block

- Step 1 : Factory reset the EB069

Hardware : Reset pin at 5V more than 3s

- Step 2: Configure the EB069

A-Connect to the wifi WLAN

B-With your best explorer

go to 192.168.1.254

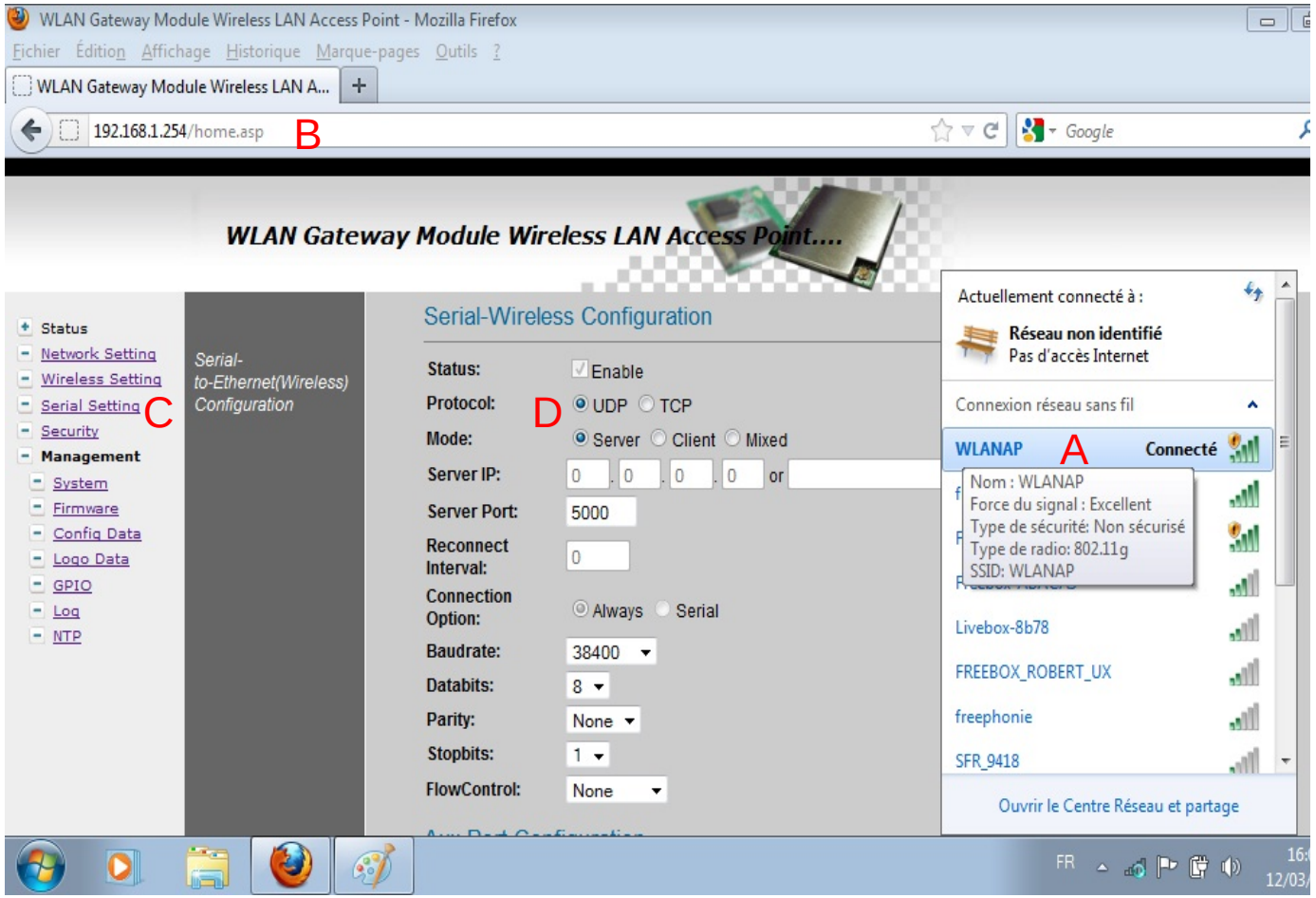
User:admin

Pass:admin

C-Serial settings tab

D-Choose on UDP

Save

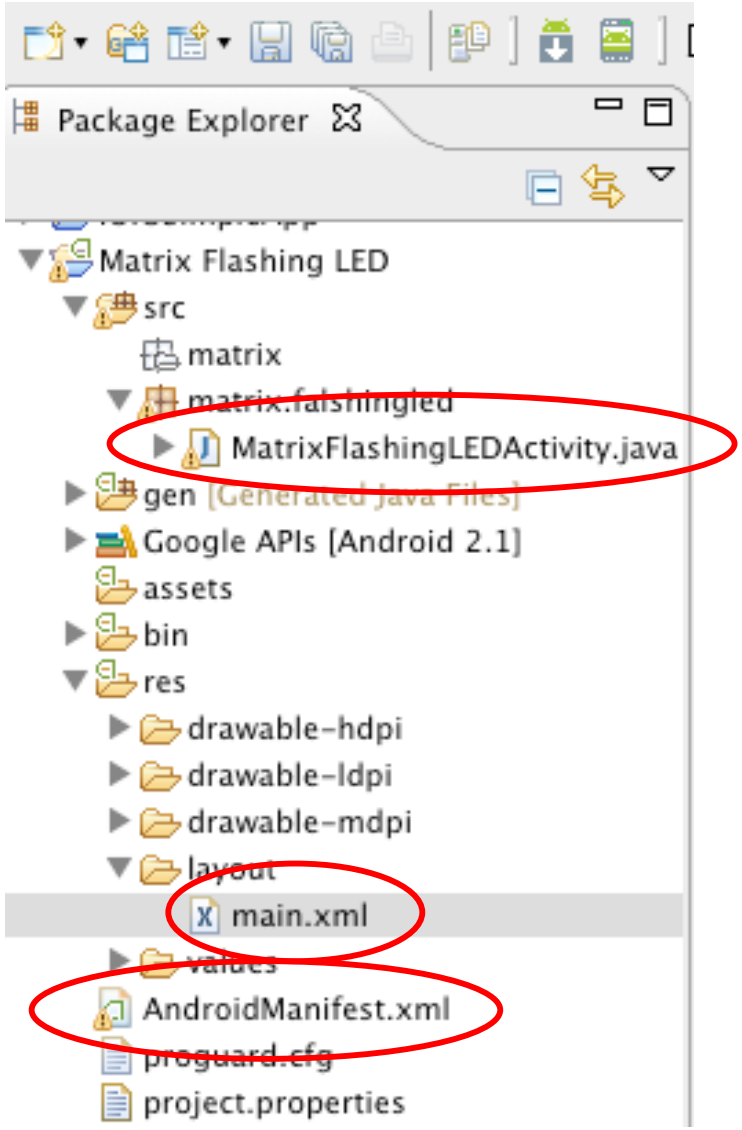


Part 2: Starting a new Eclipse Android Project

Quick starting guide

- **New → Android project**
  - **Project name** -> Matrix Flashing LED *(Eclipse project name)*
  - **API** → 7 *(Android version of your device)*
  - **Application Name** → Matrix Flashing LED
  - **Package Name** → matrix.flashingled
  - **Create Activity** → MatrixFlashingLEDActivity
  - **Min SDK Version** → 7 *(to make your appli backward compatible)*
  - **Run**
    - *Compilation and deployment into an .apk file*
    - *Load/run of the .apk file in the emulator*

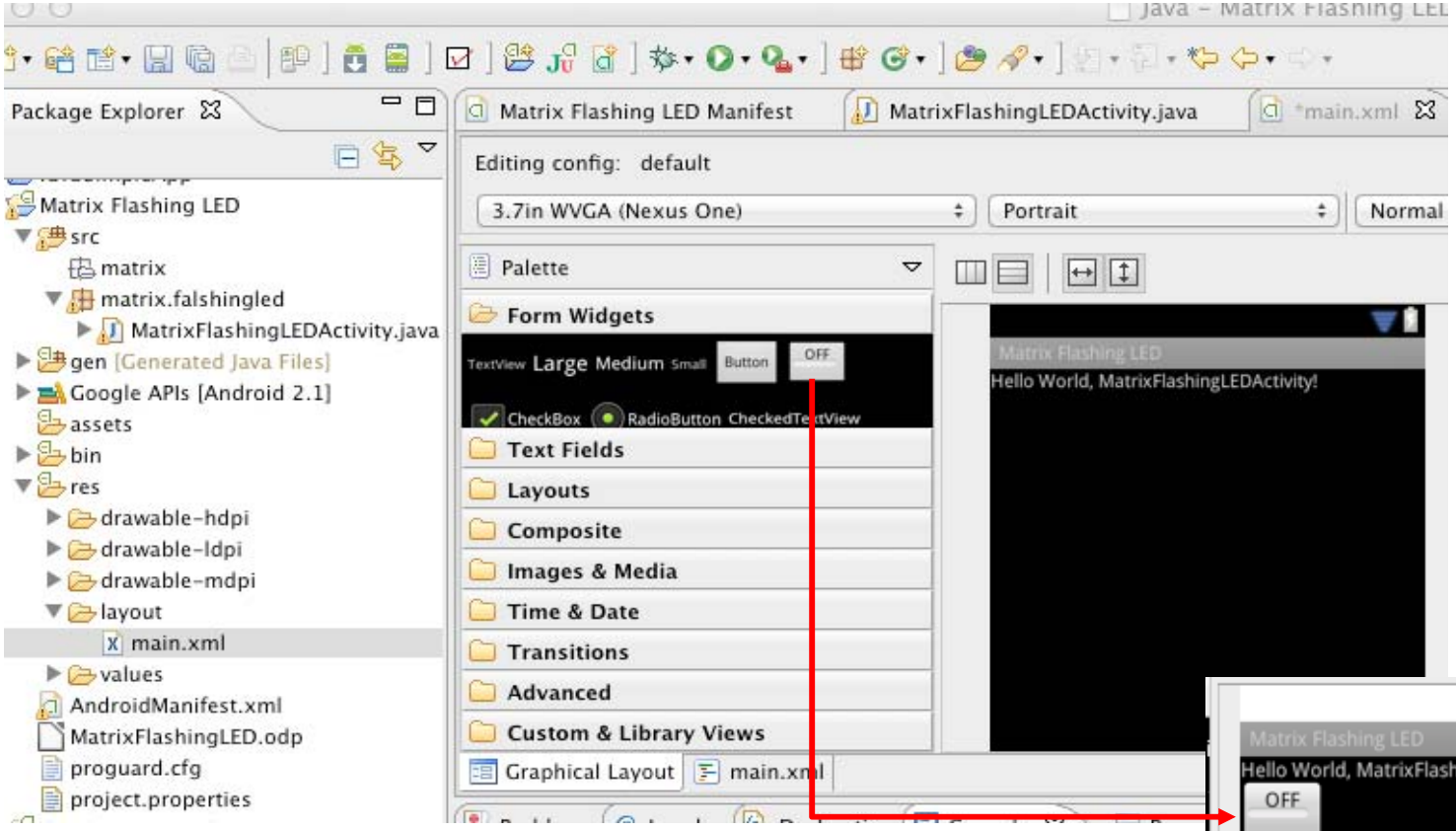
Parts of a new Android Project



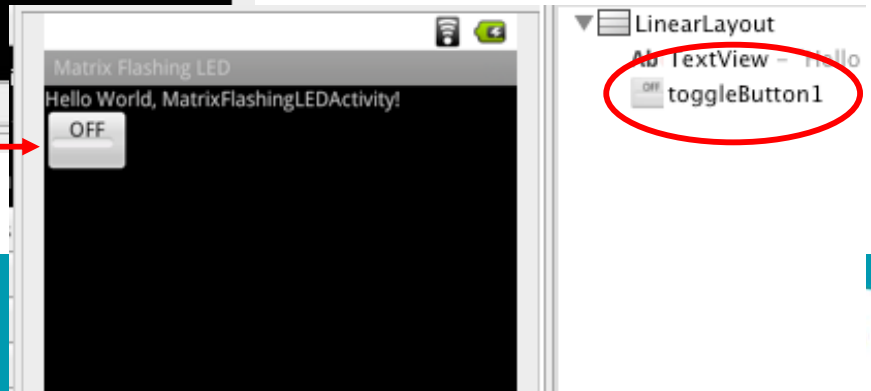
- Mainly 3 kinds of files :
  - In "src": Java files.
    - *The code of the program*
  - In "res/layout": XML files
    - *Definition of the graphical interface*
  - AndroidManifest: program descriptor
    - *Describe how to run the program, its requirements, etc.*

Building the Human Machine Interface

- Open the "res/layout/main.xml" file
- Add a toggle Button



Drag and drop a toggle button



Button's Id

Flashing LED : Java part

- **Step 1:** Open the Java file generated by Eclipse
- **Step 2:** Add 4 variables
  - *Network description*
  - *ToggleButton address*
- **Step 3:** adapt onCreate method
  - *Initializing the Network*
  - *Making it possible to use the XML declared toggle Button from Java*
  - *Associating an action to the button*
- **Step 4:** Add a new method, called when switching the button
  - *Computing a message to send*
  - *Sending the message*

```
package matrix.falshingled;
import … ;
public class MatrixFlashingLEDActivity extends Activity {
    // Global variables declaration
    private ToggleButton btn;       // Java pointer to the button
    private DatagramSocket socket;  // Object for UDP network access
    private DatagramPacket packet;  // Transmission unit for UDP
network
    private InetAddress address;    // Network host descriptor

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // Network initialization
        try {
            address = InetAddress.getByName("192.168.1.254");
            socket = new DatagramSocket();
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Binding between button and its action
        btn = (ToggleButton)findViewById(R.id.toggleButton1);
        btn.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                swapLedState();
            }
        });
    }

    public void swapLedState(){
        // Computation of the message to send (value 1 or 0)
        byte[] msg=new byte[1];
        if(btn.isChecked()){
            msg[0]=1;
        } else {
            msg[0]=0;
        }

        // Wraping message into a Datagram and sending it
        try {
            packet = new DatagramPacket(msg,msg.length,address,5000);
            socket.send(packet);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

2

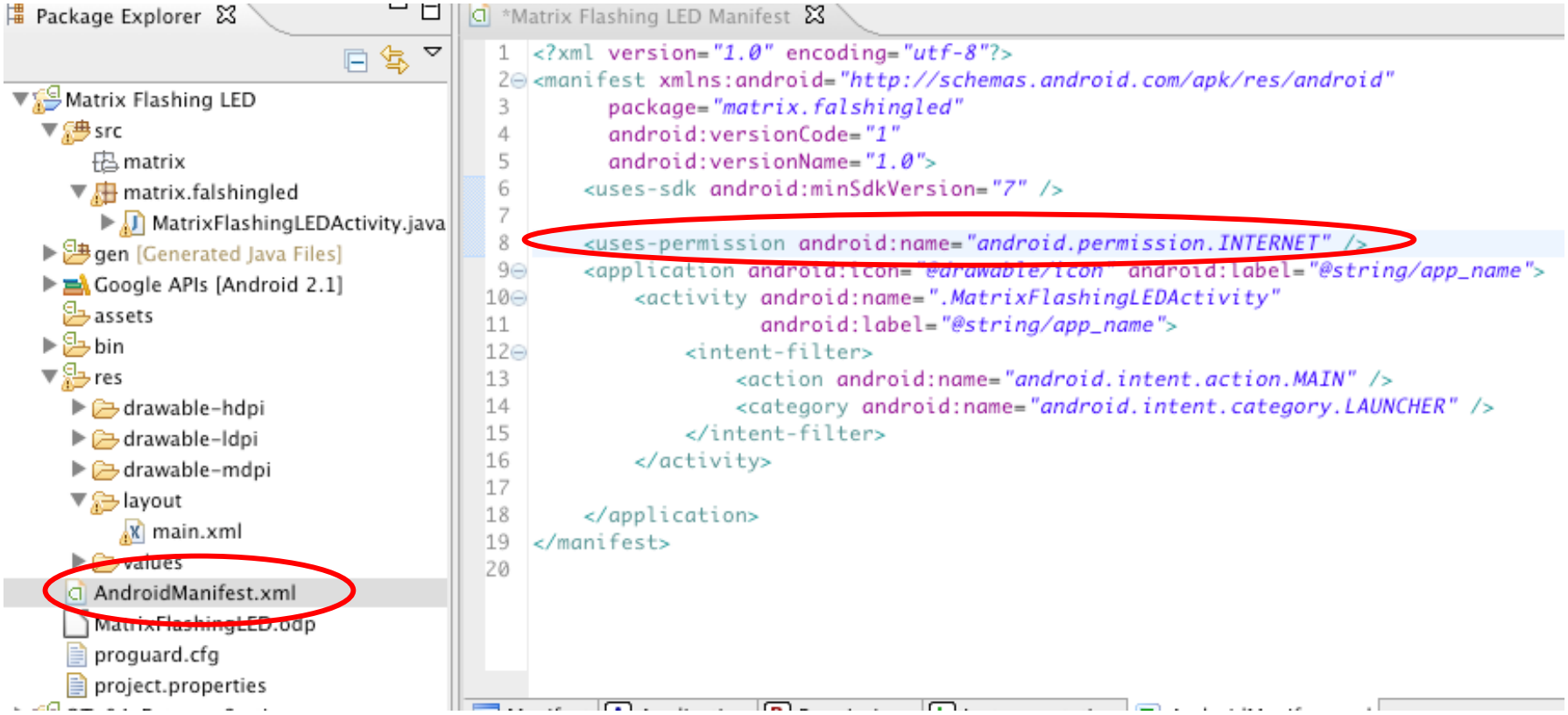
3

4

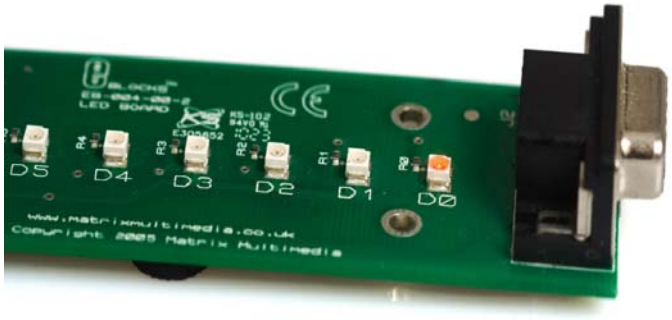
Requesting for Network Permissions

- Open the "AndroidManifest" file
- Add one line to get Network use permission:

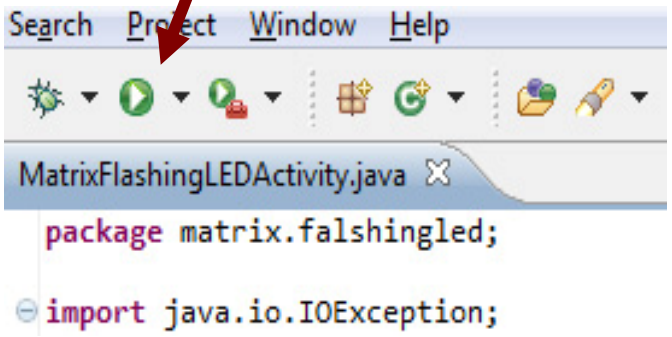
```
<uses-permission android:name="android.permission.INTERNET" />
```



That's all folks !



- Connect your Andoid device to WLANAP
- USB Plug your Android device in the computer
- Press on play in eclipse
- Touch the button



INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

FORMATION

Everything starts with a dream

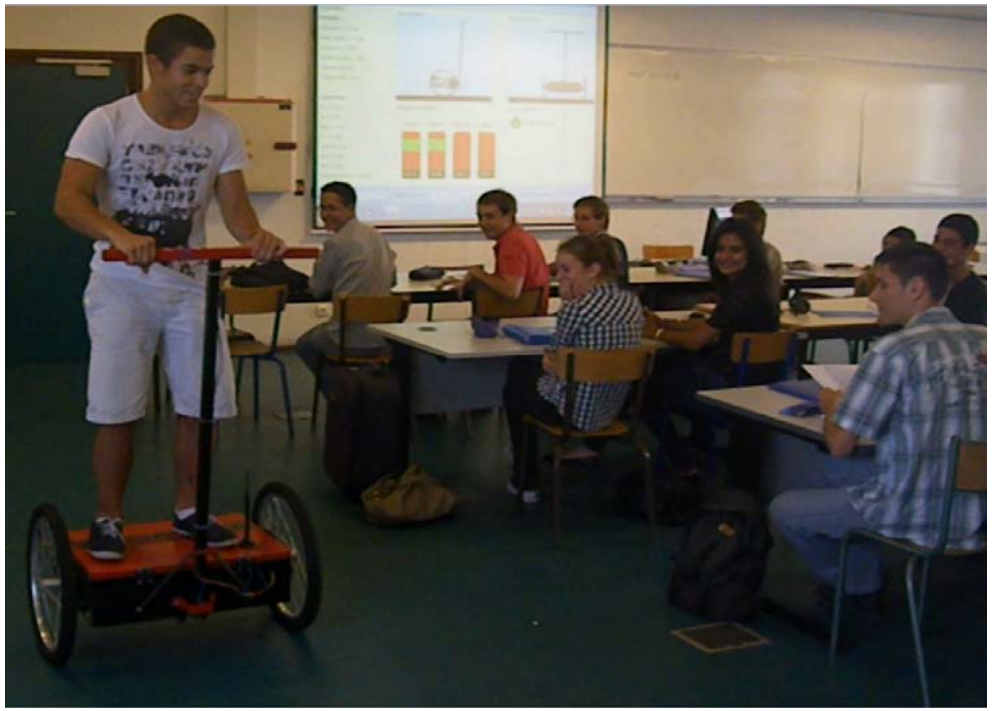
membre de
UNIVERSITÉ DE LYON

INSA LYON

What you can do...

video

video

What you can do...

- Segway like vehicle:
  - Real time parameters viewing
  - Real time stabilization parameters adjustment