

Termux, Node-RED e BLE

Steven Salazar

July 2018

Abstract

L'obiettivo del progetto è consentire l'avvio di un processo Node-RED in modo veloce e semplice attraverso l'installazione di un apk su di un dispositivo Android, l'unico prerequisito per far partire il processo sarà quello di riavviare il dispositivo dopo l'installazione. Inoltre, Node-RED dovrà offrire la possibilità di scannerizzare i dispositivi bluetooth (e.g. attraverso un nodo [9] che utilizza la libreria noble).

1 Introduzione

L'applicazione realizzata è principalmente una modifica di Termux-app in modo che permetta di:

- Installare Termux-app, Termux-boot e Termux-api attraverso un unico apk.
- Installare e avviare Node-RED senza essere a conoscenza dei comandi necessari.
- Scannerizzare i dispositivi Bluetooth Low Energy.

1.1 Termux

Termux è un software open-source [1] che permette di avere un emulatore del terminale Linux su Android. Tra le caratteristiche principali che differenziano Termux dalle altre applicazioni sta il fatto che Termux non ha bisogno del root o di configurazioni aggiuntive sul dispositivo. Una volta installata la applicazione sarà messo a disposizione un sistema di base che ci permetterà di utilizzare alcuni tool Linux e di installarne altri attraverso il package manager APT.

Un'altra caratteristica importante è il fatto che gli sviluppatori abbiano messo a disposizione un insieme di add-on che permettono di estendere le funzionalità di Termux-app. Gli add-on che si utilizzeranno per questo progetto sono: Termux-boot e Termux-api.

1.2 Termux-boot

Termux-boot è un add-on di Termux [2] che permette di eseguire degli script automaticamente al boot del telefono, in questo modo è possibile configurare il terminale, come ad esempio aggiornarlo attraverso i comandi 'apt update' e 'apt upgrade') e/o avviare un server in automatico.

N.B. Per utilizzare Termux-boot è necessario che l'apk di boot sia stato firmato con la stessa chiave di Termux-app in modo da avere i permessi per eseguire gli script.

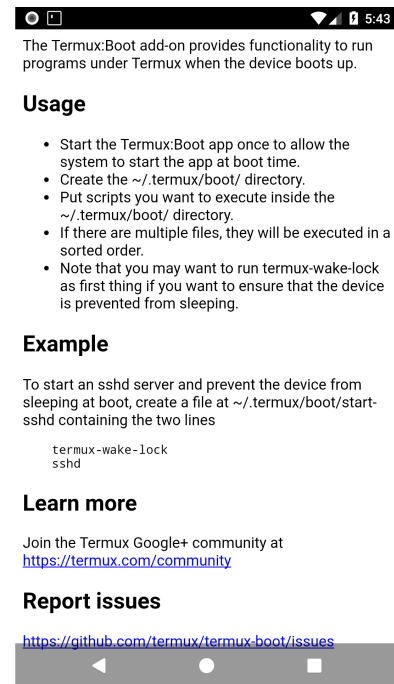
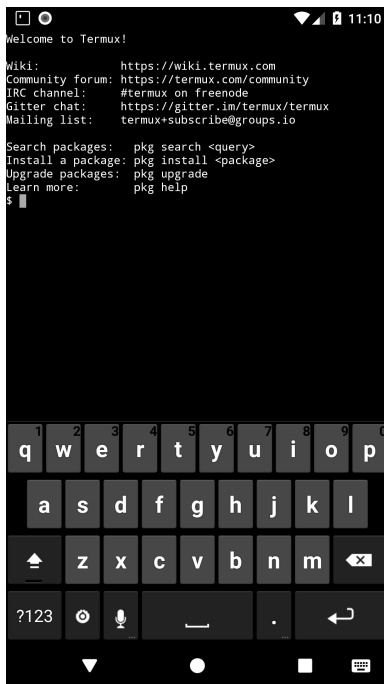


Figure 1: Termux-app e Termux-boot

1.2.1 Come utilizzare Termux-boot

1. Installare Termux-boot (deve essere stato firmato con la stessa chiave di Termux-app)
2. Avviare Termux-boot almeno una volta cliccando sull'icona che è stata aggiunta dopo l'installazione
3. Creare la cartella `~/.termux/boot/`
4. Scrivere degli script nella cartella `~/.termux/boot/`, se ce ne sono più di uno verranno eseguiti in ordine alfabetico
5. Usare `termux-wake-lock` come il primo script line se si vuole essere sicuri che l'esecuzione del file non venga sospesa dal sistema operativo.

1.3 Termux-api

Termux-api è un add-on di Termux [3] che espone una serie di funzioni che operano sul hardware del dispositivo. Queste funzioni possono essere chiamate dal command line di Termux, da degli script che poi possono essere seguiti da Termux-boot o da altri programmi (come ad esempio Node-RED su Termux-app).

N.B. Per utilizzare Termux-api è necessario che l'apk sia stato firmato con la stessa chiave di Termux-app in modo da avere i permessi per funzionare.

1.3.1 Come utilizzare Termux-api

1. Installare Termux-api dal PlayStore, FDroid o l'apk generato dal codice su github.
2. Aprire Termux-app
3. Eseguire il seguente comando in modo che venga aggiunto il pacchetto contenente una lista dei possibili comandi da chiamare
`$apt install termux-api`
4. Chiamare una qualunque funzione di Termux-api:

<code>\$termux-audio-info</code>	<code>\$termux-battery-status</code>
<code>\$termux-brightness</code>	<code>\$termux-call-log</code>
<code>\$termux-camera-info</code>	<code>\$termux-camera-photo</code>
<code>\$termux-clipboard-get</code>	<code>\$termux-clipboard-set</code>
<code>\$termux-contact-list</code>	<code>\$termux-dialog</code>
<code>\$termux-download</code>	<code>\$termux-call-log</code>
<code>\$termux-camera-info</code>	<code>\$termux-camera-photo</code>
<code>\$termux-clipboard-get</code>	<code>\$termux-clipboard-set</code>
<code>\$termux-contact-list</code>	<code>\$termux-dialog</code>
<code>\$termux-download</code>	<code>\$termux-enable-buttons</code>
<code>\$termux-fingerprint</code>	<code>\$termux-infrared-frequencies</code>
<code>\$termux-infrared-transmit</code>	<code>\$termux-location</code>
<code>\$termux-media-player</code>	<code>\$termux-media-scan</code>
<code>\$termux-microphone-record</code>	<code>\$termux-notification</code>
<code>\$termux-notification-remove</code>	<code>\$termux-sensor</code>
<code>\$termux-share</code>	<code>\$termux-sms-inbox</code>
<code>\$termux-sms-send</code>	<code>\$termux-speech-to-text</code>
<code>\$termux-storage-get</code>	<code>\$termux-telephony-call</code>
<code>\$termux-telephony-cellinfo</code>	<code>\$termux-telephony-deviceinfo</code>
<code>\$termux-toast</code>	<code>\$termux-torch</code>
<code>\$termux-tts-engines</code>	<code>\$termux-tts-speak</code>
<code>\$termux-vibrate</code>	<code>\$termux-volume</code>
<code>\$termux-wallpaper</code>	<code>\$termux-wifi-connectioninfo</code>
<code>\$termux-wifi-enable</code>	<code>\$termux-wifi-scaninfo</code>

5. Ad esempio per conoscere lo stato della batteria è sufficiente inserire il comando
`$termux-battery-status`.

1.4 Node-RED

Node-RED è un tool di programmazione basato sul flusso per l'Internet of Things. L'obiettivo è quello di dare a tutti, anche a chi non è esperto di programmazione, la possibilità di collegare tra loro diversi dispositivi, API e servizi online.

Node-RED mette a disposizione un editor di flusso visibile tramite browser che rende semplice il collegamento tra i flussi utilizzando una ampia varietà di nodi offerti in una palette. Inoltre, l'editor permette di scrivere funzioni javascript intermedie.

1.4.1 Come installare Node-RED

Node-RED può essere installato su diversi dispositivi e sistemi operativi, dato che Node-RED è costruito su nodejs è necessario aver installato nodejs prima di procedere con l'installazione.

Dato che l'obiettivo di questo progetto è quello di generare un apk, che permetta di utilizzare Node-RED, ci concentreremmo solo sull'installazione sui dispositivi android [4]:

1. Installare Termux-app in modo da avere a disposizione una console linux sul dispositivo android.
2. Aggiornare Termux-app attraverso i comandi
`$apt update` e `$apt upgrade`.
3. Installare Nodejs attraverso il comando
`$apt install coreutils nodejs`
4. Installare Node-RED con il comando
`$npm i -g --usersafe-perm node-red`.

Sugli emulatori android di AndroidStudio si creerà un problema dato che l'emulatore dice a nodejs che non possiede una CPU vera e propria e quindi non permetterà di installare ulteriori tool (e.g. Node-RED), per risolvere il problema è necessario seguire il *Paragrafo 1.4.2*.

Una volta installato Node-RED, per utilizzarlo sarà sufficiente scrivere il comando `node-red` e andare su `http://localhost:1880/` nel browser.

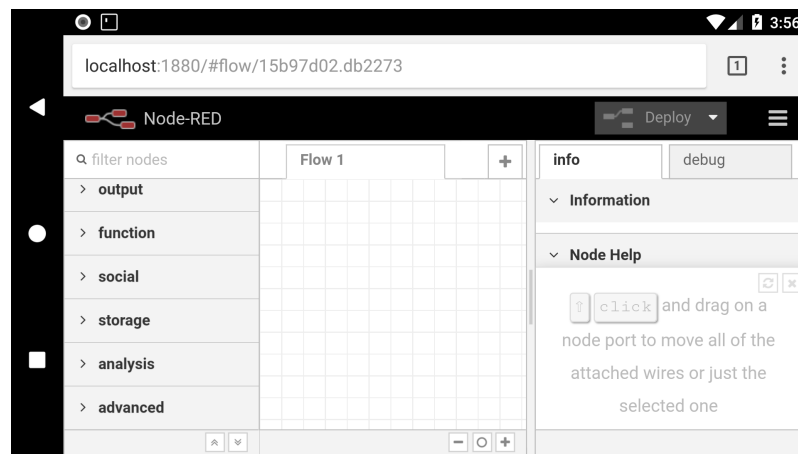


Figure 2: Node-RED

1.4.2 Node-RED su un emulatore di AndroidStudio

Com'è stato anticipato nel paragrafo *Paragrafo 1.4.1*, l'emulatore di AndroidStudio informa nodejs che non ha una CPU vera e propria, per risolvere questo problema è necessario seguire i seguenti passi:

1. Seguire i primi 3 passi nel *Paragrafo 1.4.1*
2. Installare un editor come vim o nano con il comando
`$apt install vim`
3. Aprire la cartella `$PREFIX/lib/node_modules/npm/node_modules/worker-farm/lib/`
4. Aprire il file `farm.js`
`$vim farm.js`
5. Modificare la riga

```
maxConcurrentWorkers      : require('os').cpus().length
con
maxConcurrentWorkers      : (require('os').cpus() || {length: 1}).length
```

6. Riprovare ad installare Node-RED
`$npm i -g --usersafe-perm node-red`

1.5 Noble

Noble è una libreria di nodejs che permette di scannerizzare, connettersi e comunicare con qualsiasi dispositivo tramite il protocollo bluetooth.

1.5.1 Come installare Noble

Per installare noble è necessario aver già installato nodejs e i prerequisiti mostrati in [5], dopo aver installato i prerequisiti **dovrebbe essere sufficiente** scrivere il comando `'npm install noble'` sulla console di Termux¹.

2 Termux-app, Termux-api e Termux-boot in un unico apk

Di seguito saranno riportati due metodi per mettere insieme due o più progetti in AndroidStudio². In entrambi è necessario sceglierne uno come *main project* e gli altri come due progetti da essere aggiunti al main, in questo caso è stato scelto come *main project* Termux-App.

¹Attualmente Noble non è disponibile per Android, ma come verrà mostrato nel *Paragrafo 4.2* si può forzare l'installazione.

²Come è stato mostrato nel *Paragrafo 1.3.1* per utilizzare Termux-api è necessario installare i comandi attraverso `$apt install termux-api`, questo creerà un problema che sarà risolto nel *Paragrafo 2.3*

2.1 Fusione attraverso il merge dei manifest

Una possibile soluzione per mettere insieme Termux-app, Termux-api e Termux-boot è quella di copiare le classi e risorse di Termux-api e Termux-boot per poi incollarle nel progetto Termux-app.

In questo modo sarà possibile l'accesso alle classi di api e boot dentro il progetto Termux-app, una volta incollati i files è necessario modificare il manifest di Termux-app.

Nel manifest di Termux-app oltre ad aggiungere i permessi di Api e Boot, bisogna aggiungere anche i loro receiver, providers e services in modo che questi continuino a funzionare come prima e non come semplici oggetti.

termux/app/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.termux"
    android:installLocation="internalOnly"
    android:sharedUserId="com.termux"
    android:sharedUserLabel="@string/shared_user_label">
    <!-- PERMESSI e FEATURE DI TERMUX-APP, TERMUX-API e TERMUX-BOOT -->
    ...
    <!-- FINE PERMESSI e FEATURE -->

    <!-- INIZIO PARTE DEL MANIFEST DI TERMUX-APP -->
    ...
    <!-- FINE PARTE DEL MANIFEST DI TERMUX-APP -->

    <!-- INIZIO PARTE DEL MANIFEST DI TERMUX-BOOT -->
    <receiver android:name=".boot.BootReceiver"
        android:exported="false">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED" />
        </intent-filter>
    </receiver>
    <service android:name=".boot.BootJobService"
        android:permission="android.permission.BIND_JOB_SERVICE" />
    <!-- FINE PARTE DEL MANIFEST DI TERMUX-BOOT -->

    <!-- MAIN ACTIVITY CON 4 BUTTONS --->
    <activity android:name=".MainActivity" />
    <activity android:name=".InfoActivity" />
```

```

<!-- INIZIO PARTE DEL MANIFEST DI TERMUX-API -->
<activity
    android:name=".api.DialogActivity"
    android:excludeFromRecents="true"
    android:exported="false"
    android:noHistory="true"
    android:theme="@style/DialogTheme" />
<activity
    android:name=".api.FingerprintAPI$FingerprintActivity"
    android:excludeFromRecents="true"
    android:exported="false"
    android:noHistory="true"
    android:theme="@android:style/Theme.NoDisplay" />
<activity
    android:name=".api.util.TermuxApiPermissionActivity"
    android:excludeFromRecents="true"
    android:exported="false"
    android:noHistory="true"
    android:theme="@android:style/Theme.NoDisplay" />
<activity
    android:name=".api.StorageGetAPI$StorageActivity"
    android:excludeFromRecents="true"
    android:exported="false"
    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
<provider android:name=".api.ShareAPI$ContentProvider"
    android:authorities="com.termux.sharedfiles"
    android:exported="true"
    android:grantUriPermissions="true"
    android:readPermission="com.termux.sharedfiles.READ" />
<service android:name=".api.SpeechToTextAPI$SpeechToTextService" />
<service android:name=".api.TextToSpeechAPI$TextToSpeechService" />
<service android:name=".api.SensorAPI$SensorReaderService" />
<service android:name=".api.MediaPlayerAPI$PlayerService"
    android:exported="false" />
<service android:name=".api.MicRecorderAPI$MicRecorderService"
    android:exported="false" />
<service android:name=".api.WallpaperAPI$WallpaperService" />
<receiver android:name=".api.TermuxApiReceiver"
    android:enabled="true"
    android:exported="true" />
<!-- FINE PARTE DEL MANIFEST DI TERMUX-API -->

</application>
</manifest>

```

2.2 Fusione attraverso librerie

In alternativa al metodo proposto nel *Paragrafo 2.1* è possibile implementare Termux-api e Termux-boot come librerie aar [6].

2.2.1 Come generare i file aar

Dal momento che Termux-api e Termux-boot sono due progetti android veri e propri (possiedono un manifest) la proposta di AndroidStudio è di creare due file **aar** invece di due **jar**, per generare un modulo aar bisogna seguire i seguenti passi:

1. Aprire il File **build.gradle**
2. Cambiare **apply plugin: 'com.android.application'** con **apply plugin: 'com.android.library'**
3. Rimuovere le righe **applicationId** e **shrinkResources**
4. Menu **Build/Rebuild Project** per generare il file **aar**.

2.2.2 Come importare i file aar

Una volta generato il file **aar** sia per Termux-api che per Termux-boot, è necessario aggiungerli come moduli nel Progetto Termux-app, per fare questo bisogna seguire i seguenti passi:

1. Menu **File/New/New Module**
2. Selezionare *Import aar module*, selezionare un file aar e poi fare lo stesso per l'altro file.
3. Aggiungere le dependencies, andare sul menu **File/Project Structure**, cliccare su **app** e poi su **dependencies**, cliccando sul simbolo **+** e poi **module_dependencies** è possibile selezionare i due file aar da aggiungere come dependencies.

2.2.3 Risoluzione degli errori

Dopo aver completato i tre passi (o dopo aver aggiunto le dependencies come mostrato in *Figura 3*) verrà mostrato il seguente messaggio:

Manifest merger failed with multiple errors, see logs.

Per risolverlo è sufficiente aprire **termux/app/src/main/AndroidManifest.xml**, cliccare su 'merged Manifest' e poi cliccare su tutti i suggerimenti che AndroidStudio propone.

In particolare per il caso di Termux-api è necessario aggiungere tutte le sue risorse nella cartella

termux/app/src/main/res in modo che queste possano essere trovate al momento della compilazione.

Inoltre dato che Termux-api utilizza la dependency *com.android.support:design:27.1.0* bisogna aggiungere:

```
implementation 'com.android.support:design:27.1.0'
```

come dependency nel file **termux/app/build.gradle**.

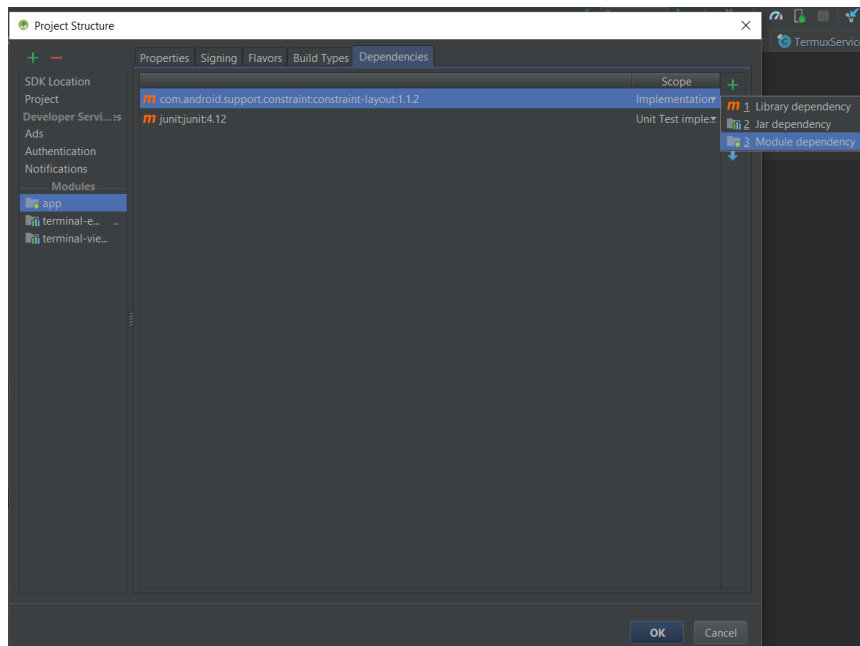


Figure 3: Come aggiungere un modulo aar in un progetto AndroidStudio

2.3 Come utilizzare le funzioni di Termux-api nell'unico apk

Come è già stato anticipato nel punto 3 del *Paragrafo 1.3.1*, dopo aver scelto uno dei due metodi descritti in precedenza per utilizzare Termux-api su Termux-app dobbiamo installare il pacchetto *Termux-api* dal terminale, provando a fare semplicemente:

```
$apt install termux-api
$termux-battery-status
```

Ci si accorge che Termux-App si blocca, questo è dovuto al fatto che non riesce a trovare la funzione java BatteryStatus, o peggio, non riesce a trovare tutte le funzioni java che abbiamo preso dal progetto Termux-api.

Questo problema è dovuto al fatto che il pacchetto installato utilizzando **apt** presenta un problema per il nostro caso, la funzione (nel file `termux-api.c`) che associa i comandi da console con le funzioni *java* di Termux-api, cerca come Receiver `com.termux.api/.TermuxApiReceiver`, quando in realtà avendo modificato l'apk il package è diventato `com.termux` e il Receiver si trova in `com.termux/.api.TermuxApiReceiver`.

Per risolvere questo problema ci sono due alternative:

- Installare Termux-api-package manualmente e non attraverso il comando **apt**, per poi modificarlo dalla console.
- Caricare online una versione già modificata e poi installarla direttamente dalla console.

Per il primo caso sarà necessario eseguire i seguenti comandi dalla console³:

```
1 $apt install curl make clang -y
2 $curl -sL https://github.com/termux/termux-api-package/archive/v0.32.tar.gz | tar xz
3 $cd termux-api-package-0.32
4 $sed -i "s/com.termux.api\/.TermuxApiReceiver/com.termux\/.api.TermuxApiReceiver/g" termux-api.c
5 $make
6 $make install
```

Nel *Paragrafo 4.3* verrà mostrato che oltre a modificare quella riga di codice, per offrire la possibilità di scannerizzare i dispositivi bluetooth si dovrà modificare ulteriormente il pacchetto termux-api-package, per questo motivo la seconda soluzione sarà quella adottata. Per installare termux-api-package già modificato è sufficiente eseguire i seguenti comandi:

```
1 $apt install curl make clang -y
2 $curl -sL https://github.com/StevenSalazarM/Termux-api-bluetooth/archive/v0.39.tar.gz | tar xz
3 $cd Termux-api-bluetooth-0.39
4 $make
5 $make install
```

3 Installare e avviare Node-RED in automatico

Com'è stato anticipato nel *Paragrafo 1.4.1* per installare Node-RED è necessario eseguire dei comandi su Termux. L'obiettivo del progetto però è quello di permettere l'utilizzo facile e semplice di Node-RED senza la necessità di aprire la console di Termux-app, per fare questo, utilizzeremmo Termux-boot che permette di eseguire dei comandi in automatico all'avvio del telefono (*Paragrafo 1.2*).

Per permettere a Termux-boot di eseguire i comandi in un file (e.g. `start`) dobbiamo creare questo file nella directory `~/.termux/boot/`, così per permettere l'esecuzione dei comandi è necessario creare questa cartella e file in automatico all'installazione dell'applicazione.

3.1 Creazione della cartella e lo script da eseguire in automatico

Dato che l'utente potrebbe, per qualche motivo, cancellare i dati dell'applicazione il codice seguente che crea la cartella e il file deve essere messo dentro un servizio, in modo che venga sempre verificata la loro esistenza.

³ 'curl' permette di scaricare il pacchetto, tramite l'opzione `-s` si evita una progress bar e si entra in modo 'silenzioso', con `L` invece, si dice che siamo pronti ad essere reindirizzati se l'indirizzo è stato spostato e se ce lo comunica nel suo header. 'sed' attraverso l'opzione `-i` permette di modificare una stringa di un file con un'altra.

termux/app/src/main/java/com/termux/app/TermuxService.java

```
/*
...
*/

@Override
public void onCreate() {
    setupNotificationChannel();
    startForeground(NOTIFICATION_ID, buildNotification());
    // code to make the file ~/.termux/boot/start
    SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
    // se non esiste 'firstTime' creiamo la cartella, così se il file viene cancellato verrà
    ↪ ricreato
    if (!prefs.getBoolean("firstTime", false)) {
        File folder = new File(HOME_PATH+"/.termux/boot/");
        if(!folder.exists()){
            boolean result = folder.mkdirs();
        } // string che contiene i comandi da scrivere nel file
        String string = "#!/data/data/com.termux/files/usr/bin/sh\n" +
            "termux-wake-lock\n"+
            "[ -f $PREFIX/bin/node-red ] || termux-toast \"Updating repository\" &&
            ↪ termux-vibrate\n"+
            "[ -f $PREFIX/bin/node-red ] || apt update\n"+
            "[ -f $PREFIX/bin/node-red ] || apt upgrade -y\n"+
            "[ -f $PREFIX/bin/node ] || termux-toast \"Installing packages\" && termux-vibrate\n"+
            "[ -f $PREFIX/bin/node ] || apt -y install coreutils nano nodejs openssh git\n"+
            "[ -f $PREFIX/bin/node-red ] || termux-toast \"Installing node-red\" &&
            ↪ termux-vibrate\n"+
            "[ -f $PREFIX/bin/node-red ] || npm i -g --unsafe-perm node-red\n" +
            "[ -d $PREFIX/lib/node_modules/node-red/node_modules/node-red-contrib-termux-api ] ||
            ↪ cd $PREFIX/lib/node_modules/node-red/\n"+
            "[ -d $PREFIX/lib/node_modules/node-red/node_modules/node-red-contrib-termux-api ] ||
            ↪ termux-toast \"Installing termux-api nodes\" && termux-vibrate \n"+
            "[ -d $PREFIX/lib/node_modules/node-red/node_modules/node-red-contrib-termux-api ] ||
            ↪ npm install node-red-contrib-termux-api\n"+
            "[ -d $PREFIX/lib/node_modules/node-red/node_modules/node-red-contrib-snap4city-user ]
            ↪ || termux-toast \"Installing node-red-contrib-snap4city-user nodes\" &&
            ↪ termux-vibrate\n"+
            "[ -d $PREFIX/lib/node_modules/node-red/node_modules/node-red-contrib-snap4city-user ]
            ↪ || npm install
            ↪ git+https://github.com/disit/node-red-contrib-snap4city-user.git\n"+
            "[ -d $PREFIX/lib/node_modules/node-red/node_modules/node-red-dashboard ] ||
            ↪ termux-toast \"Installing node-red-dashboard\" && termux-vibrate \n"+
            "[ -d $PREFIX/lib/node_modules/node-red/node_modules/node-red-dashboard ] || npm
            ↪ install node-red-dashboard\n"+
            "termux-enable-buttons\n"+
            "termux-toast \"starting node-red\" \n\n"+
            "node $PREFIX/bin/node-red\n";
        try { FileOutputStream fos = new FileOutputStream(new
            ↪ File(HOME_PATH+"/.termux/boot/start"));
            fos.write(string.getBytes());
            fos.close();
        } catch (Exception e){
            Log.d("termux","exception creating a file "+e.getMessage());
        } // mark first time has runned.
        SharedPreferences.Editor editor = prefs.edit();
        editor.putBoolean("firstTime", true);
        editor.commit();
    } }
```

`[-f $PREFIX/bin/node-red] ||` comando significa che se non esiste il file `$PREFIX/bin/node-red` dobbiamo eseguire comando.

Per le directory si utilizza `'-d'`.

È utile notare che la cartella `/.node-red` viene creata solo quando Node-RED è stato avviato almeno una volta, per cui l'installazione di `node-red-contrib-snap4city-user`, `node-red-contrib-termux-api` e `node-red-dashboard` verrà fatta su `$PREFIX/lib/node_modules/node-red/`.

Termux-enable-buttons abilita i pulsanti "Node-RED" e "Dashboard" del MainActivity.
`node $PREFIX/bin/node-red` avvia Node-RED.

4 Scanner dei dispositivi Bluetooth

Per scannerizzare i dispositivi Bluetooth Low Energy sono state considerate due opzioni, una attraverso la libreria noble [5] e l'altra attraverso codice nativo java che implicherà poi dover aggiungere manualmente un nodo in Node-RED [7] che interagisce con la funzione `BluetoothScanInfo` della classe `termux/app/src/main/java/com/termux/api/BluetoothAPI.java`.

4.1 Noble su Android

Com'è stato anticipato nel *footnote di Pagina 5* attualmente noble non è disponibile sui dispositivi Android, compreso anche il fatto di utilizzare la console linux (debian) di Termux-App. Esiste però, la possibilità di installare noble manualmente e fargli accettare anche i sistemi operativi android, questo è possibile scaricando il pacchetto e modificando due file in cui noble verifica il sistema operativo.

4.2 Modifica di noble per farlo funzionare su Android

Una volta installato Termux-app e nodejs i passi da fare ([5] e [8]) per far funzionare noble sono questi :

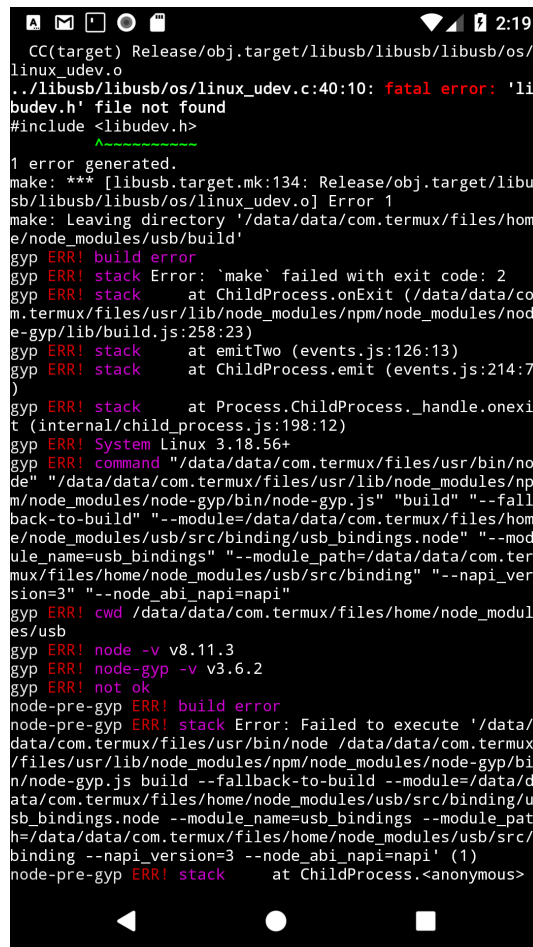
```
1      # andare nella cartella HOME in modo da scaricare e modificare i pacchetti lì
2      $cd $HOME
3      # installare python2
4      $apt install python2 -y
5      $alias python=python2
6      # installare node-gyp come è stato suggerito in [8]
7      $npm install -g node-gyp
8      # scaricare noble e decomprimerlo in modo da poter modificarlo
9      $npm pack noble
10     $tar -xvzf noble-1.9.1.tgz
11     # aprire il package di noble scaricato
12     $cd package
13     # modificare i 2 file in modo da aggiungere 'android' come una delle piattaforme supportate
14     $vim lib/resolve-bindings.js
15     $vim package.json
16     $cd ..
17     # prima di installare il pacchetto già modificato è necessario installare le dependencies
```

```

18 # di cui noble ha bisogno (e.g. bluetooth-hci-socket) dato che in caso di
19 # provare ad installarlo
20 # npm install package
21 # ci darà l'errore che si vede in Figura 4
22 # il problema però è che bluetooth-hci-socket ha bisogno di libusb che al suo interno
23 # ha bisogno di libudev (che non si riesce a trovare perché attualmente
24 # non è disponibile su Termux).
25 # Gitter Termux/Termux -> http://prntscr.com/kd0yz1

```

Così, per il nostro caso non è stato possibile l'utilizzo noble per scannerizzare i dispositivi bluetooth.



```

CC(target) Release/obj.target/libusb/libusb/libusb/os/
linux_udev.o
../libusb/libusb/os/linux_udev.c:40:10: fatal error: 'li
budev.h' file not found
#include <libudev.h>
^
1 error generated.
make: *** [libusb.target.mk:134: Release/obj.target/libu
sb/libusb/libusb/os/linux_udev.o] Error 1
make: Leaving directory '/data/data/com.termux/files/hom
e/node_modules/usb/build'
gyp ERR! build error
gyp ERR! stack Error: 'make' failed with exit code: 2
gyp ERR! stack     at ChildProcess.onExit (/data/data/co
m.termux/files/usr/lib/node_modules/npm/node_modules/nod
e-gyp/lib/build.js:258:23)
gyp ERR! stack     at emitTwo (events.js:126:13)
gyp ERR! stack     at ChildProcess.emit (events.js:214:7
)
gyp ERR! stack     at Process.ChildProcess._handle.onexi
t (internal/child_process.js:198:12)
gyp ERR! System Linux 3.18.56+
gyp ERR! command "/data/data/com.termux/files/usr/bin/no
de" "/data/data/com.termux/files/usr/lib/node_modules/np
m/node_modules/node-gyp/bin/node-gyp.js" "build" "--fall
back-to-build" "--module=/data/data/com.termux/files/hom
e/node_modules/usb/src/binding/usb_bindings.node" "--mod
ule_name=usb_bindings" "--module_path=/data/data/com.ter
mux/files/home/node_modules/usb/src/binding" "--napi_ver
sion=3" "--node_abi_napi=napi"
gyp ERR! cwd /data/data/com.termux/files/home/node_modul
es/usb
gyp ERR! node -v v8.11.3
gyp ERR! node-gyp -v v3.6.2
gyp ERR! not ok
node-pre-gyp ERR! build error
node-pre-gyp ERR! stack Error: Failed to execute '/data/
data/com.termux/files/usr/bin/node /data/data/com.termux
/files/usr/lib/node_modules/npm/node_modules/node-gyp/bi
n/node-gyp.js build --fallback-to-build --module=/data/d
ata/com.termux/files/home/node_modules/usb/src/binding/u
sb_bindings.node --module_name=usb_bindings --module_pat
h=/data/data/com.termux/files/home/node_modules/usb/src/
binding --napi_version=3 --node_abi_napi=napi' (1)
node-pre-gyp ERR! stack     at ChildProcess.<anonymous>

```

Figure 4: Libudev-dev not found

4.3 Scanner Bluetooth con Termux-api

Dopo aver visto che non è stato possibile l'utilizzo di noble, nasce dunque la necessità di trovare un metodo alternativo per scannerizzare i dispositivi bluetooth con Node-RED, di seguito verrà mostrato come questo è possibile tramite un'ulteriore modifica del progetto con l'unico apk e una modifica del pacchetto che viene installato tramite console (per permettere la chiamata delle funzioni java).

creare il file `termux/app/src/main/java/com/termux/api/BluetoothAPI.java`

```
public class BluetoothAPI {

    static void onReceiveBluetoothScanInfo(TermuxApiReceiver apiReceiver, final Context context,
    ↪ final Intent intent) {
        ResultReturner.returnData(apiReceiver, intent, new ResultReturner.ResultJsonWriter() {

            @Override
            public void writeJson(final JsonWriter out) throws Exception {
                // Metodo da implementare, nel file del progetto è stato aggiunto uno sketch di come
                ↪ dovrebbe funzionare
                // chiamato da 'termux-bluetooth-scaninfo' nella console
                out.beginObject().name("message:").value("BluetoothScanInfo da
                ↪ implementare").endObject();
            }
        });
    }

    static void onReceiveBluetoothConnect(TermuxApiReceiver apiReceiver, final Context context,
    ↪ final Intent intent) {
        // Metodo da implementare
        // chiamato da 'termux-bluetooth-connect device_uuid' nella console
        ResultReturner.returnData(apiReceiver, intent, new ResultReturner.WithStringInput(){

            @Override
            public void writeResult(PrintWriter out) throws Exception {
                try {
                    JsonWriter writer = new JsonWriter(out);
                    writer.setIndent("  ");
                    writer.beginObject().name("message:").value("BluetoothConnect da implementare,
                    ↪ dovrebbe connettersi a " + inputString);
                    out.println(); // To add trailing newline.
                } catch (Exception e){
                    Log.d("termux", "errore bluetooth connect "+e.getMessage());
                }
            }
        });
    }
}
```

Creazione dello script termux-bluetooth-scaninfo

```
# codice da https://github.com/StevenSalazarM/Termux-api-bluetooth/blob/master/scripts/
#!/data/data/com.termux/files/usr/bin/sh
set -e -u

SCRIPTNAME=termux-bluetooth-scaninfo

show_usage () {
    echo "Usage: $SCRIPTNAME"
    echo "Get information about the last bluetooth scan."
    exit 0
}

while getopts :h option
do
    case "$option" in
        h) show_usage;;
        ?) echo "$SCRIPTNAME: illegal option -$OPTARG"; exit 1;
    esac
done
shift $((OPTIND-1))

if [ $# != 0 ]; then echo "$SCRIPTNAME: too many arguments"; exit 1; fi

/data/data/com.termux/files/usr/libexec/termux-api BluetoothScanInfo
```

Creazione dello script termux-bluetooth-connect

```
# codice da https://github.com/StevenSalazarM/Termux-api-bluetooth/blob/master/scripts/
#!/data/data/com.termux/files/usr/bin/bash
set -e -u
SCRIPTNAME=termux-bluetooth-connect

show_usage () {
    echo "Usage: termux-bluetooth-connect [device_uuid]"
    echo "Connects to a given device and uuid"
    exit 0
}

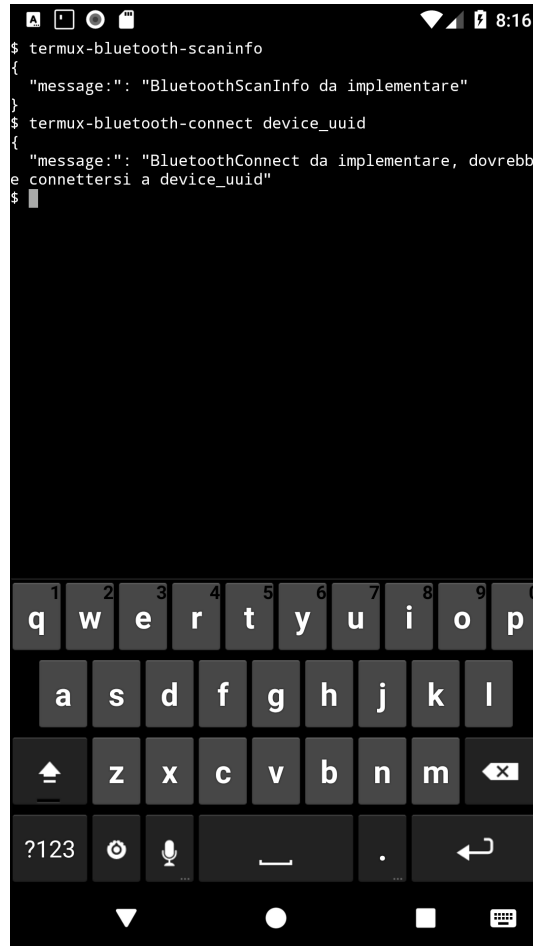
PARAMS=""
shift $((OPTIND-1))

CMD="/data/data/com.termux/files/usr/libexec/termux-api BluetoothConnect $PARAMS"

if [ $# = 0 ]; then
    $CMD
else
    echo $@ | $CMD
fi
```

Dopo aver creato questi 3 file, l'unica cosa che rimane è aggiungere i due case nello switch di `termux/app/src/main/java/com/termux/api/TermuxApiReceiver.java`, uno che invoca il metodo `onReceiveBluetoothScanInfo` quando l'`api-method` chiamato è `BluetoothScanInfo`, e l'altro che invoca `onReceiveBluetoothConnect` nel caso di `api-method=BluetoothConnect`.

Come si può osservare nella *Figura 5* ora è possibile chiamare due funzioni, una farà lo scan dei dispositivi e l'altra permetterà di connettersi ad un servizio di un device.



```
$ termux-bluetooth-scaninfo
{
  "message:": "BluetoothScanInfo da implementare"
}
$ termux-bluetooth-connect device_uuid
{
  "message:": "BluetoothConnect da implementare, dovrebbe
e connettersi a device_uuid"
}
$
```

Figure 5: Termux-bluetooth-scaninfo and Termux-bluetooth-connect

4.3.1 Node-RED con le api di termux

Così come con noble è possibile installare un nodo 'SCAN BLE' su Node-RED [9], con la soluzione di uno scanner in *java* è anche possibile.

Per fare questo, sarà sufficiente modificare leggermente la palette che offre le api di Termux-api, *node-red-contrib-termux-api* [10], l'unica cosa da dover fare è aggiungere un ulteriore nodo che chiamerà 'termux-bluetooth-scaninfo'.

5 Una applicazione per tutti

Arrivati a questo punto abbiamo già raggiunto l'obiettivo del progetto, l'unica cosa che rimane è quella di offrire tutto in una applicazione più 'friendly', ovvero una applicazione che offre una interfaccia grafica semplice, con la possibilità di aprire direttamente Node-RED, la sua **dashboard** o la console di **Termux-app** (per i più esperti).

Per raggiungere l'obiettivo è sufficiente aggiungere un Activity nel progetto, l'activity sarà come quello in *Figura 6*.

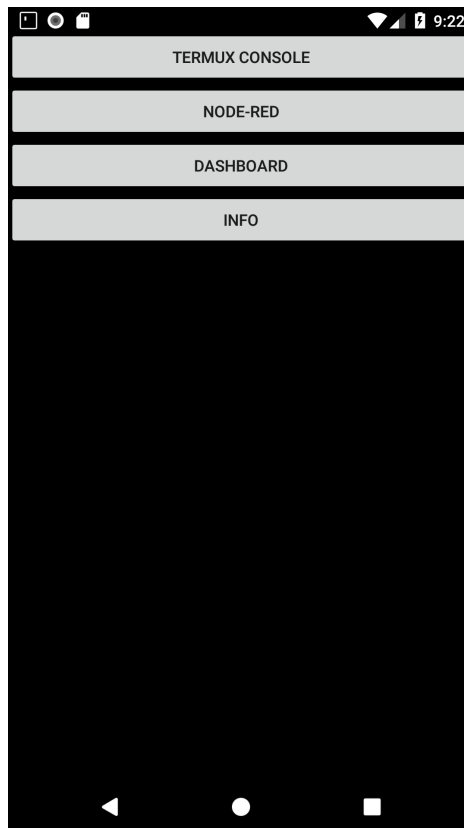


Figure 6: MainActivity

5.1 Gestione del MainActivity

Per aggiungere il MainActivity come quello principale (ovvero, quello che deve essere mostrato nell'apertura dell'app) è possibile aggiungerlo come `main_activity` nel manifest. Questo però, implica che bisognerà installare tutto, ovvero copiare tutto il codice di `TermuxActivity` e metterlo in questo activity, codice che permetterà di avviare i servizi, bisognerà gestire anche l'apertura di Termux dalla barra delle notifiche, etc.

Per semplicità, la gestione del MainActivity è stata fatta all'interno di `TermuxActivity`, così non c'è bisogno di modificare né il manifest né `TermuxInstaller`, non c'è bisogno neanche di spostare il codice che fa partire i servizi, né la gestione dell'apertura dalla barra delle notifiche.

Le uniche cose che bisogna fare in `TermuxActivity` sono:

- Aggiungere un'attributo che fa riferimento ad un `intent` (che servirà per far partire MainActivity), questo attributo verrà inizializzato alla fine del metodo `onCreate()`.
- Fare l'override del metodo `onResume`
- Fare l'override del metodo `onBackPressed`

termux/app/src/main/java/com/termux/app/TermuxActivity.java

```
private Intent MainOptions;
@Override
protected void onCreate(Bundle bundle) {
    /*
        ...
    */
    MainOptions = new Intent(this, MainActivity.class);
}
@Override
protected void onResume() {
    super.onResume();
    if(MainActivity.ActualActivity == 1 ){
        //ActualActivity = 1 significa che bisogna mostrare MainActivity
        //ActualActivity = 2 significa che bisogna mostrare Termux-Console
        startActivity(MainOptions);
    }
}
@Override
public void onBackPressed() {
    // così si crea l'illusione che TermuxConsole non è l'activity principale ma che quando si torna
    ↪ indietro si torna nel main_activity
    MainActivity.ActualActivity=1;
    startActivity(MainOptions);
}
```

Seguendo la stessa logica, su MainActivity bisogna fare l'override degli stessi metodi

termux/app/src/main/java/com/termux/app/MainActivity.java

```
@Override
protected void onResume() {
    super.onResume();
    // permette all'api_method enablebuttons di abilitare i bottoni node-red e dashboard
    activity = this;
    if (enableNodeRed) {
        btnNodeRed.setEnabled(true);
        btnDashBoard.setEnabled(true);
    }
}

@Override
public void onBackPressed() {
    // bisogna ricordare che l'ultimo Activity visibile era MainActivity
    ActualActivity = 1;
    // minimize App serve per far finta di chiudere l'app dopo il BackPressed della schermata
    ↪ principale.
    minimizeApp();
}
```

In questo modo il MainActivity si mostra come il main_activity del progetto, quando in realtà il main_activity rimane sempre TermuxActivity.

Riferimenti

- [1] Termux-app è disponibile nel PlayStore, F-Droid e come open-source in <https://github.com/termux/termux-app>
- [2] Termux-boot è disponibile a pagamento nel PlayStore, gratis su F-Droid o come open-source in <https://github.com/termux/termux-boot>
- [3] Termux-api è disponibile nel PlayStore, F-Droid o come open-source in <https://github.com/termux/termux-api>
- [4] Installazione di Node-RED su Android <https://nodered.org/docs/platforms/android>
- [5] Installazione di noble su linux <https://github.com/noble/noble#linux>
- [6] Come combinare due progetti android studio in uno solo <https://stackoverflow.com/questions/33528400/>
- [7] Palette che permette di utilizzare le funzioni di Termux-api <https://flows.nodered.org/node/node-red-contrib-termux-api>
- [8] Noble su android <https://github.com/noble/noble/issues/571>
- [9] Nodo che permette di scannerizzare i dispositivi bluetooth tramite noble <https://flows.nodered.org/node/node-red-contrib-noble>
- [10] Termux-api su Node-RED <https://github.com/chameleonbr/node-red-contrib-termux-api>