Relazione LAM 2015/2016

ThingSpeak4Android

Natale Vadalà
Matr. 691364

1. Introduzione

L' applicazione, disponibile per dispositivi Android, nasce con l'obiettivo di monitare i sensori di uno smartphone, raccogliere questi risultati su un server e riuscire a visualizzarli in un secondo momento direttamente sul cellulare.
Per ottenere ciò (e al meglio), l'utente dovrà consentire all'app di ottenere vari permessi:

- `android.permission.CHANGE_WIFI_STATE`
- `android.permission.ACCESS_WIFI_STATE`
- `android.permission.INTERNET`
- `android.permission.ACCESS_NETWORK_STATE`
- `android.permission.ACCESS_COARSE_LOCATION`
- `android.permission.CHANGE_NETWORK_STATE`
- `android.permission.BODY_SENSORS`
- `android.permission.RECEIVE_BOOT_COMPLETED`
- `com.android.alarm.permission.SET_ALARM`
- `android.permission.ACCESS_COARSE_LOCATION`
- `android.permission.ACCESS_FINE_LOCATION`
- `android.permission.ACCESS_LOCATION_EXTRA_COMMANDS`

2. Scopo e funzionalità

Le features dell'applicazione, fondamentalmente, sono tre (come descritto dalla specifica):

(1) Sensor Configuration
(2) Send Data
(3) Get Statistics

La (1) è stata implementata in toto: vengono visualizzati tutti i sensori disponibili (inclusi dati relativi a connessione Wifi e Mobile) ed è possibile selezionare alcuni sensori per iniziare l'attività di reporting.
La (2) è stata implementata quasi da specifica (Problematiche incontrate, vedi punto 6.e): è possibile iniziare l'attività di monitoring e di comunicazione con il server, attraverso un servizio che risveglia l'attività principale anche dopo l'uscita dall'applicazione, e che parte al boot del cellulare. É possibile, inoltre, modificare la frequenza di reporting dei sensori, e scegliere se usare la rete Wifi, la connessione dati, o qualsiasi connessione, per comunicare i propri risultati al sito.
La (3) è stata implementata in toto: è possibile scaricare i grafici del proprio reporting dal server,

selezionando i dati dell'ultimo giorno/settimana/mese.

3. Requisiti per la progettazione ed ambiente di sviluppo

Per progettare l'app, è stato necessario (se non fondamentale), appoggiarsi ad alcuni servizi esterni per: a) mantenere i dati; b) disegnare i grafici.
a)" ThingSpeak"[1] è il nome del servizio utilizzato per conservare i dati: gratuito, ben dotato di metodi per la creazione, l'aggiornamento e l'eliminazione di canali direttamente con richieste Get/Post/Delete, semplice da utilizzare e da interrogare (poiché i dati vengono mantenuti sul server in JSON). Dà la possibilità, inoltre, di accedere ai propri grafici dal sito web e esportare grafici ben più complessi.

b)"HelloChart"[2], libreria per Android, che permette di rappresentare grafici su app Android in modo semplice e, devo dire, molto pulito.

A proposito di ambiente di sviluppo, è stato utilizzato l'IDE "Android Studio"[3] per la progettazione, diversi dispositivi Android collegati alla console per la fase di testing.

4.Progettazione

L'app è stata progettata con Android 5.0 e Api level 21.

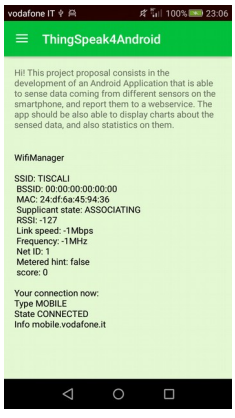
E' composta da:

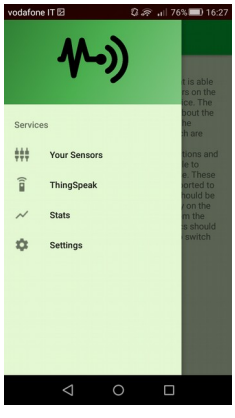
A) Activity
1. MainActivity
2. SensorListActivity
3. SensorDetailActivity
4. SpeakActivity
5. StatsActivity
6. Stat2Activity
7. SettingsActivity

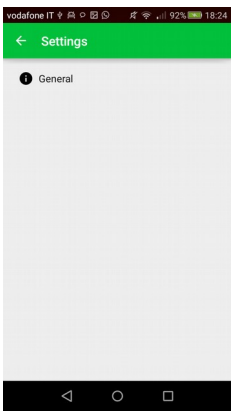
B) Service
1. MainService

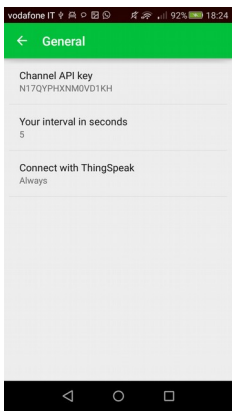
C) BroadcastReceiver
1. MyBroadcastReceiver

MainActivity: è una NavigationDrawerActivity, fornisce l'accesso a tutte le altre Activities.

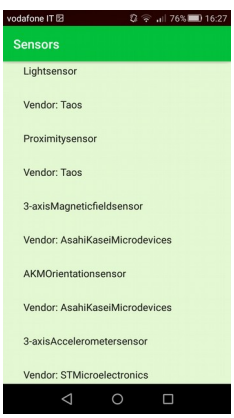


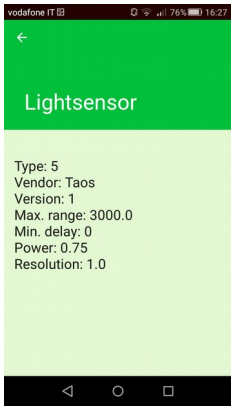






SettingsActivity, è una PreferenceActivity.
Permette di cambiare account (cambiando la Master API-Key di Thingspeak), cambiare la frequenza di reporting e selezionare che tipo di connessione utilizzare.



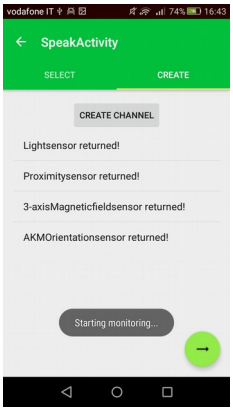


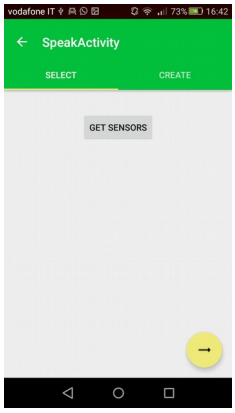
SensorListActivity: è una MasterDetailActivity, con la possibilità (su tablet) di visualizzare tutto su una sola tab, infatti è costituita dalla SensorDetailActivity e dal SensorDetailFragment.

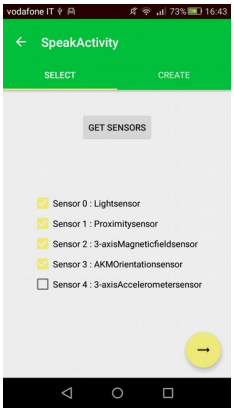
All'avvio inizializza tutti i sensori disponibili e permette di visualizzarli in una lista espandibile, che mostra i dettagli di ciascun sensore rilevato.

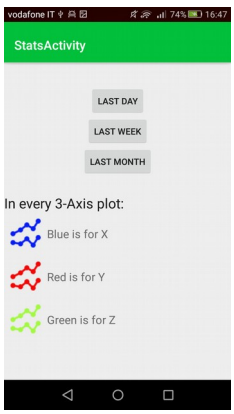
SpeakActivity è una TabbedActivity, ed è quella che ci permette di fare un mini "setup" della nostra applicazione, rilevando i sensori, dandoci la possibilità di scegliere quale tracciare e di creare i canali per Thingspeak.

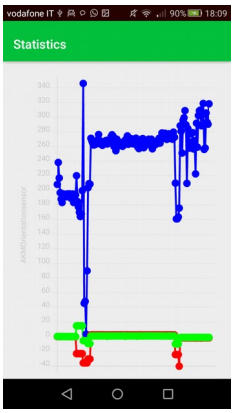
Alla fine di tutti gli step, se tutto è andato a buon fine (fondamentalmente, se c'è abbastanza linea per non fare andare in timeout le post), sarà stato creato un canale per ogni sensore.
Il FloatingButton permette di iniziare l'attività principale: quella di monitoring e reporting al server.











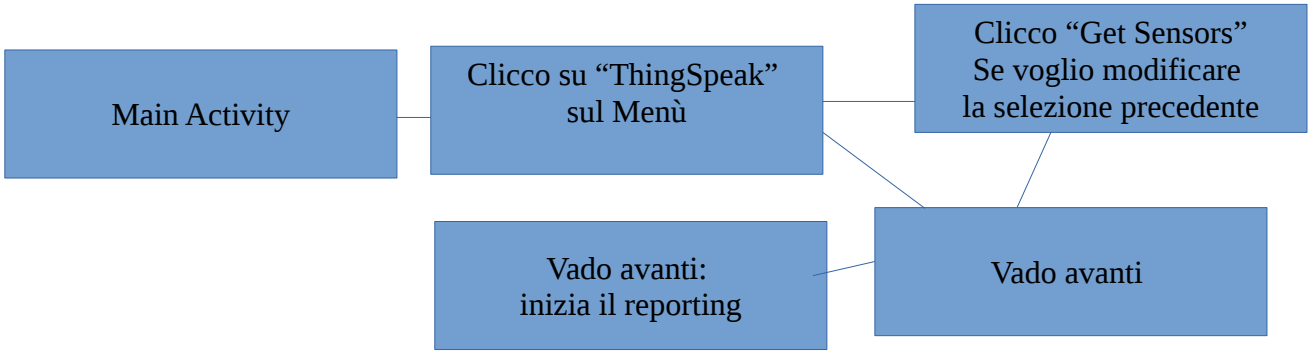
StatsActivity e **Stats2Activity** sono, rispettivamente, quella che chiede i feed dei canali al server e disegna il grafico, e quella che li rappresenta su una TabbedActivity.

MainService è il nucleo dell'applicazione, la classe che implementa un SensorEventListener per ottenere i valori dei sensori e genera gli update da mandare al server (con chiamate asincrone ottenute attraverso la classe *AsyncTask*).
Il service è in background, ha la capacità di rimanere vivo anche dopo la distruzione di tutte le Activities grazie al **MyBroadcastReceiver** che lo risveglia, anche dopo il reboot dello smartphone.

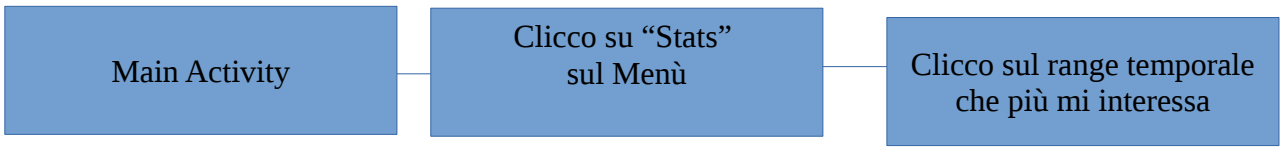
5. Caso d'uso tipico

Primo avvio.

“Setup” già realizzato



Statistiche



6. Problematiche incontrate

Le problematiche incontrate sono state di diverso tipo:
a) Scrittura del Service, la parte riguardante i sensori e la frequenza di aggiornamento, risolto

implementando un SensorEventListener che, al momento del cambio di valore di un sensore (la prima volta), questo viene “unregistrato” dal Listener, per essere riregistrato al tempo T richiesto dall'utente attraverso un Handler ed una richiesta PostDelayed.

b) Scrittura del Service, la parte riguardante l'avvio al boot e l'avvio alla distruzione dell'app, risolto banalmente cambiando telefono (Huawey P8 Light) e leggendo su Internet: installando app da debug usb, Huawey non dà la possibilità di acconsentire all'esecuzione in background (bisogna buildare l'APK e installarlo da telefono).
c) Scrittura della CreateChannel, poiché teoricamente dovrebbe essere una chiamata sincrona, risolto simulando il sincronismo attraverso chiamate asincrone.
d) Gestione Preference “Only Data” per fare update (in caso di conflitti l'utente dovrebbe poter chiedere di attivare i Dati se ha scelto “Only Data”), poiché su android >=5.0 sono state eliminate funzioni dal ConnectionManager (come la *setMobileDataEnabled)* e le uniche possibili funzioni da utilizzare (appartenenti al TelephonyManager) funzionano solo su smartphone con permessi di root.
e) Salvataggio file in locale, non implementata poiché, se quei dati venissero postati in differita dal reale tempo di monitoring, il reporting (e le statistiche) non sarebbe né realistiche né veritiere (Thingspeak salva l'ora di post del valore)

7. Possibili estensioni

Possibili estensioni potrebbero essere quelle in fatto di portabilità (es. implementazione per iOS), piuttosto che ai fini ambientali (usare server che effettivamente analizzino questi dati) o concentrarsi sull'aumento dello spettro dei sensori utilizzabili (gestendo anche sensori esterni al telefono e registrati dall'app in un qualche modo semplice per l'utente), potendo creare in pochi minuti una centralina meteorologia controllata da smartphone .

8. Commenti finali

Lo sviluppo di questa applicazione è stato molto interessante e stimolante, per vari motivi: è stata l'esperienza universitaria che più di tutte mi ha avvicinato (realmente) al paradigma della Programmazione Object Oriented, permettendomi anche di imparare un nuovo linguaggio di programmazione (nella pratica) quale il Java, sebbene sono consapevole che devo migliorare su più aspetti.
Credo comunque di aver acquisito una buona conoscenza dell'ambiente e delle tecniche di sviluppo.

Inoltre, mi è piaciuto dover lavorare con i sensori: è stato così stimolante che dopo lo sviluppo di questa applicazione mi sono avvicinato al mondo del “Crowdsense Data” con Arduino, e al mondo dei sensori in generale.

9. Riferimenti

[1] github.com/lecho/hellocharts-android
[2] it.mathworks.com/help/thingspeak
[3] developer.android.com/studio/index.html