1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

Reti di Calcolatori

Bluetooth P2P Chat

Alessio Muzi mat. n°299329 | Informatica Applicata | A.S. 2021/2022

Introduzione

Per il progetto di esame di Reti di Calcolatori, l'argomento che è stato scelto come focus dell'approfondimento è il Bluetooth, principalmente per tre motivi:

1. Interesse personale verso questo protocollo wireless, economico, a basso consumo energetico, versatile e fondamentale per l'IT.

2. La volontà di costruire qualcosa di pratico, utilizzabile e che sfruttasse le conoscenze sia apprese nel corso di Reti ma anche interdisciplinari, integrando gli strumenti forniti dal corso di Laurea nella sua interezza.

3. La volontà di applicare le conoscenze del corso in un nuovo ambiente precedentemente sconosciuto come test delle proprie capacità.

Il **Bluetooth** (BT) è un protocollo WPAN nato nel 1994 in **Ericsson**, il quale sviluppo è poi stato allargato all'ente **Bluetooth SIG** (*Special Interest Group*). Lo scopo di questo protocollo era inizialmente di sostituire le funzioni dei cavi delle periferiche con soluzioni wireless. Nel tempo, sono stati definiti altri use cases.

L'obbiettivo posto per questo progetto è di sviluppare un'app Android semplice ma funzionale per la comunicazione tramite Bluetooth. Il nome scelto per l'app è **Bluetooth P2P Chat**, nome evocativo delle sue funzioni.

La relazione del progetto sarà così strutturata:

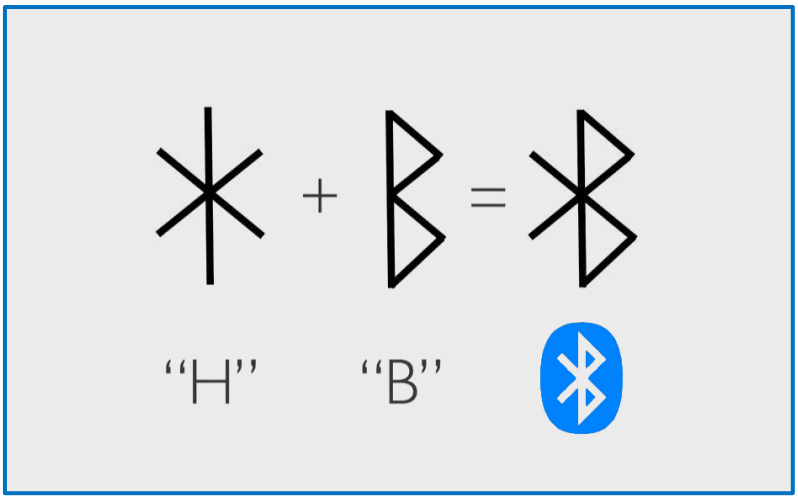
- Introduzione del Bluetooth: storia, funzioni, stack e usi comuni.
- Introduzione di altri concetti rilevanti per il progetto: P2P, MVVM, API.
- Breve introduzione agli strumenti utilizzati per lo sviluppo: linguaggio di programmazione Kotlin, ambiente Android Studio, device per il test.
- Illustrazione delle funzioni dell'applicazione.
- Esempi di uso.
- Considerazioni finali e conclusione.
- Bibliografia e sitografia

Bluetooth: storia e protocollo

Per parlare del Bluetooth è necessario prima introdurre la **storia** che ha portato alla sua ideazione, poiché ha **influenzato** in modo profondo, diretto e decisivo caratteristiche, specifiche tecniche, distribuzione e profili d'uso. Come la quasi totalità degli standard in campo industriale e informatico, anche il Bluetooth è strettamente legato ad un ente **no-profit** (poiché **open source**) che si occupa di ricerca, sviluppo, distribuzione del marchio, sezione amministrativa, specifiche minime e rilascio delle versioni al pubblico ma non della distribuzione e vendita in prima persona dei dispositivi. Il nome di questo ente è **Bluetooth SIG**, *Special Interest Group*.

La storia del Bluetooth inizia in Svezia, nell'azienda **Ericsson**, la quale si occupava di produrre hardware per la **telefonia**. La specifica iniziale del Bluetooth è stata sviluppata in Ericsson verso la fine degli **anni '90**, mentre venivano effettuate ricerche per uno standard che potesse **sostituire il cablaggio** per la trasmissione dei dati nell'uso di periferiche come mouse e tastiere. In seguito la specifica venne formalizzata dalla Bluetooth SIG, la cui costituzione è stata formalmente annunciata il 20 maggio 1999 da **Ericsson**, **Sony**, **IBM**, **Intel**, **Toshiba**, **Nokia**. Successivamente il numero di società che si sono aggiunte come associate o come membri (come per esempio **Microsoft** o **Lenovo**) è cresciuto rapidamente, fino ad arrivare a 30.000 aziende attuali. Il Bluetooth divenne quindi uno standard comune e accettato da tutti per la trasmissione dei dati **wireless a corto raggio**, portando alla nascita di aziende che producono device con il BT già integrato e modificando le catene di montaggio e i prodotti dell'epoca.

Il nome "Bluetooth" è ispirato al soprannome di **Harald Blåtand**, re di Danimarca (970-986) ed abile diplomatico che unì gli scandinavi introducendo nella regione il cristianesimo. Il suo soprannome era "***dente azzurro***", sulla cui origine gli storici non riescono a concordare: forse per via del fatto che andasse ghiotto di mirtilli oppure perché in battaglia si colorava i denti di blu per incutere paura al nemico, oppure ancora per il suo aspetto. Gli inventori della tecnologia hanno ritenuto che fosse un nome adatto per un protocollo **capace di mettere in comunicazione dispositivi diversi**. Il logo del Bluetooth unisce le rune nordiche *Hagall* (ᚼ) e *Berkanan* (ᛒ), analoghe alle moderne lettere H e B. Il colore ufficiale del logo Bluetooth è **#0082FC**.



Questo standard venne progettato con l'obiettivo di ottenere bassi consumi, un corto raggio d'azione e bassi costi di produzione. Lo standard doveva consentire il collegamento wireless tra periferiche come stampanti, tastiere, telefoni, microfoni, etc. a computer o PDA. Il 4 gennaio 2007 è stato superato il **miliardo** di dispositivi che utilizzano questa tecnologia. I dispositivi che integrano chip Bluetooth sono venduti in milioni di esemplari e sono abilitati a riconoscere e usare periferiche Bluetooth, svincolandosi dal collegamento cablato. Il primo produttore di automobili a introdurre il Bluetooth nei propri veicoli è stato **BMW**, permettendo ai guidatori di rispondere al telefono senza staccare le mani dal volante e permettere molte altre funzionalità con comando vocale.

Con le versioni **1.1** e **1.2** di Bluetooth sono gestiti trasferimenti di dati con velocità fino a 723,1 KBit/s. Nella versione **2.0** è stata aggiunta una modalità con velocità più elevata, che consente trasferimenti fino a 3 MBit/s. Tuttavia questa modalità comporta un incremento notevole dei consumi nei dispositivi, che prevalentemente sono alimentati a batteria. Con la versione **4.0** dello standard la velocità di trasferimento è stata incrementata fino a 4 Mbit/s. Nel frattempo è stato introdotto un nuovo criterio, la riduzione della durata dei segnali trasmessi, che a parità di quantità di dati trasferiti riesce a dimezzare la corrente richiesta rispetto al Bluetooth v. 1.2. La versione attuale del BT è **5.3**.

Uno step importante fu l’aggiunta della versione **Bluetooth Low Energy** (**BLE**), introdotta nel 2011. Essa infatti permette le stesse funzionalità del Bluetooth classico (ovviamente ridotte in velocità) ma con un consumo energetico minore, dovuto ad una modulazione dei segnali semplificata. Questo fondamentale sviluppo dello standard fu quello che permise la compatibilità del Bluetooth con i **device mobili**, i quali non erano ancora potenti come gli odierni smartphone, e tutt’ora forniscono il supporto a questa versione del protocollo.

Una volta descritta la storia, passiamo ai dettagli **tecnici** e **implementativi**.

Il protocollo Bluetooth opera alla frequenza **2,4 GHz** e utilizza un segnale di clock per mandare i segnali ogni 1,3 o 5 frame. Per ridurre le interferenze la banda viene divisa in 79 canali e provvede a commutare tra i vari canali 1.600 volte al secondo tramite la tecnica di trasmissione **FHSS**, ovvero **Frequence Hopping Spread Spectrum**, la quale può adattarsi per comunicare con il Wi-Fi. La modulazione del segnale Bluetooth avviene usando la tecnica detta **Gaussian Frequency Shift Keying**. Il Bluetooth sfrutta onde radio a corto raggio e richiede un modesto consumo di energia elettrica. Ogni dispositivo dotato di un chip è in grado di creare una rete di dimensioni limitate: il raggio d'azione può variare da 10m a 30m (**Personal Area Network – PAN**, descritto in **IEEE 802.15**). Una WPAN si differenzia dalle reti LAN/WLAN per l'area minore di copertura, per la tipologia esclusivamente **ad-hoc** (quindi senza Access Point), per il **plug-and-play** alla base dell'architettura e per il supporto alla **comunicazione vocale**.

Una rete Bluetooth prende il nome di **PICONET**: può supportare 7 device slave e 1 device master e l'unione di due o più reti di questo tipo è detta **SCATTERNET**. Bluetooth dispone di una gestione approfondita dei consumi, permettendo di mandare in **standby** i nodi attualmente non in uso. Esistono 3 **power class** (e la versione Low Energy) con velocità, data rates, range e usi diversi. Il Bluetooth consente due tipi di collegamento tra i vari nodi di una rete: **SCO** e **ACL**. Un collegamento **SCO** (*Synchronous Connection-Oriented*) permette fino a 3 link a banda fissa **punto-punto** sincroni, mentre un collegamento **ACL** (*Asynchronous ConnectionLess*) è unico nella PICONET ed è un link **multicast** asincrono. I link SCO sono usati per la voce in real time, mentre il link ACL per il trasporto dati.

Lo stack Bluetooth è composto da 2 **strati** (radio/fisico e Data-Link/Logical Link Control) e vari moduli **componibili**. I moduli obbligatori sono **LMP**, che gestisce il link radio, **L2CAP**, che si occupa della suddivisione in frame e **SDP**, il quale ricerca servizi e informazioni in altri device. Esistono poi dei protocolli che svolgono funzioni specifiche ai vari profili d'uso decisi dal produttore: **RFCOMM**, che simula le porte seriali, **OBEX**, che regola lo scambio di oggetti simile a HTTP, **TCS BIN**, che regola le chiamate e protocolli derivati da **TCP** e **UDP** per il trasporto dati. La sicurezza dei dispositivi è affidata all'algoritmo **SAFER+.**

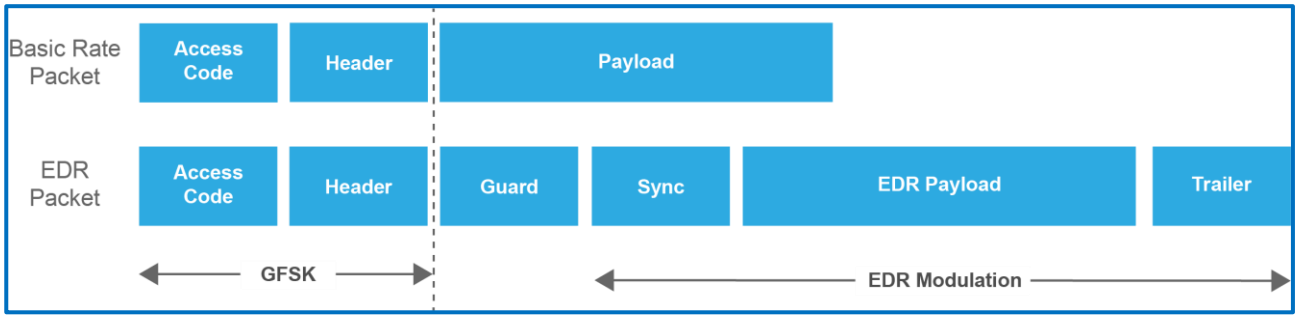
Il Bluetooth impiega vari **codec** audio e video come **SBC** (audio base), **MPEG-1** (A2DP), H.263 e MPEG-4 (video) e sistemi di correzzione dell'errore **FEC**.

Header Bluetooth

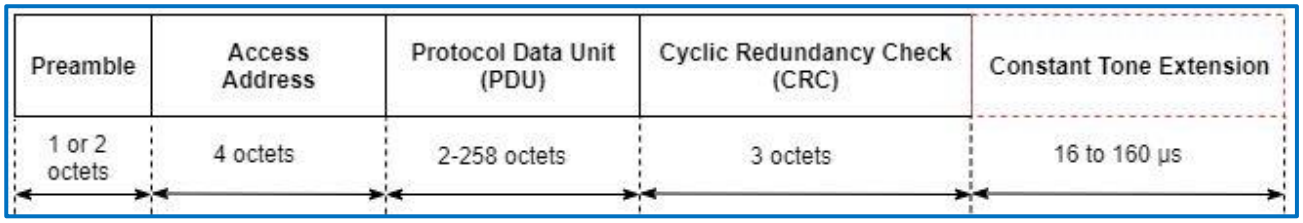
Il pacchetto trasmesso da due dispostivi che comunicano tramite il Bluetooth è molto **semplice**, permettendo una comunicazione agile e a basso consumo.

L'intestazione del pacchetto Bluetooth (versione 5.0) sono le seguenti:

Bluetooth classico



Bluetooth Low Energy



Il **Bluetooth classico** (<u>in due diversi data rate per profili diversi</u>) è composto da:

- Codice d'accesso (indirizzo)
- Intestazione con le informazioni
- Payload
- Banda di guardia
- Segnale di sincronizzazione
- Trailer che segnala la fine del pacchetto

Data rate base

Enhanced data rate

Il **Bluetooth LE** invece è molto più semplice ed è composto da:

- Preambolo, che esegue le operazioni di sincronizzazione
- Indirizzo d'accesso
- PDU, ovvero il payload
- CRC, ovvero un sistema di rilevazione errori basato su polinomi

Opzionalmente, possiamo trovare il **CTE** (*Constant Tone Extension*), il quale fornisce informazioni sul pacchetto.

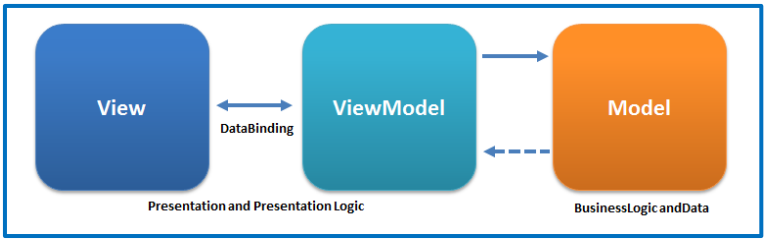
Architetture software: P2P e MVVM

Nel corso dello sviluppo del progetto è evidente come alcune delle funzionalità dell'app siano riconducibili a template ricorrenti. Oltre la già citata architettura **master-slave**, alla base del funzionamento del Bluetooth, sono due i modelli che risultano utili alla risoluzione della specifica: il primo riguarda la modalità di comunicazione tra due dispositivi (**Peer-to-Peer**) mentre il secondo è specifico della struttura software dell'applicazione (**Model-View-ViewModel**).

L'architettura *Peer-to-Peer* (**P2P**) è una architettura logica tra le più comuni negli ambiti dell'informatica e delle telecomunicazioni. Essa si differenzia dall'altro modello principale, Client-Server, poiché in questa struttura i nodi non seguono una **gerarchia**, quindi tutti i nodi sono equivalenti e **all'occorrenza** possono diventare il processo client o server, anche contemporaneamente. In questo modo, sono risolti problemi di **interoperabilità** tra nodi che potrebbero avere caratteristiche diverse ed inoltre non esiste un **SPOF**, cioè **Single Point of Failure**: se un nodo smette di funzionare per qualsiasi motivo, il messaggio può comunque trovare un percorso alternativo. Un altro vantaggio è di non dover mantenere un server dalle prestazioni elevate, promuovendo una **rete scalabile agevolmente** e rapida. Un grande svantaggio è la questione della sicurezza in presenza di agenti malevoli. Questo tipo di architettura adempie alla perfezione ai requisiti della specifica del progetto sviluppato. Degli esempi di architettura P2P applicata a servizi sono alcuni sistemi di file sharing come **EMule** o **Napster**.

L'architettura *Model-View-ViewModel* (**MVVM**) riguarda invece lo sviluppo del codice. Esso è un pattern software, consigliato da Google, che astrae lo stato di **vista** ("*view*") dal comportamento stesso. In questo modo abbiamo tre agenti:

- **Model**: contiene e gestisce la logica dell'applicativo e i dati (linguaggi di definizione del software come *Kotlin*/*Java*);
- **View**: responsabile di ciò che l'utente finale vede sullo schermo, tra cui layout, struttura e aspetti (linguaggi di markup come *XAML*/*XML*);
- **ViewModel**: intermediario tra il modello e la vista, il quale invoca le classi del modello e consegna i dati alla vista in maniera agevole.



API Bluetooth

Il concetto di **API** è fondamentale per la diffusione dello sviluppo software in grandi comunità di developer. L'acronimo API sta per **Application Programming Interface**, cioè "*interfaccia di programmazione di una applicazione*" e fungono da astrazione di un set di procedure tra il programmatore e l'hardware. Esistono inoltre le **ABI**, ovvero **Application Binary Interface**, le quali svologono lo stesso compito tra sistema operativo e applicazioni.

Essenzialmente le API sono il **modello** tramite il quale gli sviluppatori in un dato ambito distribuiscono le **librerie** che forniscono dei servizi senza dover riscrivere ogni volta le funzioni base. Le API contengono funzioni, variabili, procedure e strutture dati a secondo degli scopi per cui è stata ideata, comportandosi come una specie di "**scatola nera**" che nasconde l'implementazione al developer con livelli di astrazione elevati.

Esistono molti modelli di distribuzione delle API: possono essere libere, open source, **aperte**, disponibili al pubblico o al privato o addirittura disponibili in modo esclusivo a un certo numero di **sviluppatori certificati** (per esempio, per poter sviluppare su console Xbox, Microsoft fornisce le proprie API solamente a degli sviluppatori registrati nei loro sistemi). Il **contenuto** di un'API può essere dei più disparati: interfacce BIOS, funzioni dei sistemi operativi, web service, linguaggi di programmazione e servizi come meteo, finanza o news.

L'API Bluetooth utilizzata per lo sviluppo del progetto, essendo su piattaforma Android, è mantenuta e definita da **Google**. Essa è distribuita in due forme: il **Bluetooth Classico-RFCOMM** e il **Bluetooth Low Energy-GATT**. Una volta che è stata effettuata la scelta, possiamo importare il package nel nostro progetto e usare le **classi** che servono ai nostri scopi. Come esempio dell'utilizzo dell'API, è presentato una breve porzione di codice significativo.

```
// Il risultato è TRUE se il Bluetooth è disponibile
// Il risultato è FALSE se non lo è o se il telefono non lo supporta
@SuppressLint("HardwareIds")
fun isBluetoothAvailable() =
    !(bluetoothAdapter == null || TextUtils.isEmpty(bluetoothAdapter.address))

// Ottieni il MAC Address
fun getBluetoothDevice(macAddress: String) = bluetoothAdapter?.getRemoteDevice(macAddress)
```

Il linguaggio di programmazione Kotlin e l'ambiente di sviluppo Android Studio

Android è un sistema operativo per dispositivi mobile con interfaccie utente specializzate ed embedded open-source basato sul kernel **Linux** e sviluppato da **Google**. Poiché Android è l'OS mobile più diffuso al mondo (71%) ed è facile ottenere dispositivi che possono eseguirlo, la scelta di sviluppare un'app partendo da Android è scontata. Per fare ciò esistono strumenti adatti.

Nel 2012 un'azienda di sviluppo ceca, **JetBrains**, rilascia al pubblico il linguaggio di programmazione **Kotlin**, che prende il nome da un'isola vicino le coste di San Pietroburgo. Esso è un linguaggio basato su *Java* e sulla *JVM*, il quale tenta di mettere a fattor comune le caratteristiche principali dei moderni linguaggi di sviluppo, principalmente del paradigma ad **oggetti**. Kotlin è un linguaggio orientato verso la programmazione a oggetti (ma permette anche lo sviluppo **funzionale**) con tipicizzazione **forte** e **statica**. Come nel caso di Bluetooth, anche per Kotlin nacque un'ente per gestire lo sviluppo: la **Kotlin Foundation**. Sebbene lo sviluppo Android è possibile in molte forme, tra cui lo stesso Java, dal 2019 Kotlin è diventato il linguaggio preferito per lo sviluppo di app di Google. Questo è il motivo che mi ha portato ha scegliere Kotlin come linguaggio di sviluppo per l'applicazione del progetto. Nello stesso periodo Kotlin divenne la prima scelta disponibile in **Android Studio**, l'ambiente di lavoro mantenuto da Google per lo sviluppo di applicazioni. In particolare, ciò avvenne nella versione 3.0, mentre la versione stabile attuale dell'IDE è denominata "*Chipmunk*" e corrisponde più o meno alla versione 5.2. Android Studio è l'IDE multipiattaforma ufficiale per Android ed è basato sull'IDE **IntellJ IDEA** sviluppato da **JetBrains**. Anche qui la scelta risulta molto semplice, poiché essendo l'IDE ufficiale di Google permette un supporto adeguato, una **grande comunità di sviluppo**, funzioni ad-hoc per lo sviluppo di app e una **facilità** per il passaggio da codice, database, interfaccia utente (**UI**) e test su dispositivo (fisico o **virtualizzato**). L'estensione dei file è *.kt*.

Per quanto l'interfaccia grafica (la quale è ovviamente molto semplice e basica), è stato utilizzato **Material Design**, design sviluppato da Google e rilasciato nel 2014. I device su cui l'app è stata testata sono un **Samsung A40** e un **Samsung S9** di mia proprietà, i quali appartengono a due fasce di prezzo differenti.

L'**API** Bluetooth utilizzata è gestita da Google.

Illustrazione dell'app e delle sue funzioni

Le specifiche dell'applicazione prefissate come risultato di questo progetto sono semplici ma significative e permettono di lavorare a 360° con l'API Bluetooth e di applicare in modo preciso tutto il percorso teorico compiuto fino a questo punto.

Il nome scelto per l'applicazione è **Bluetooth P2P Chat** (*BP2PC*), il quale descrive in modo efficace quelle che sono le proprie funzioni. L'app permette in modo semplice e rapido la comunicazione tra due dispositivi mobile (esclusivamente, escludendo quindi PC, laptop, tablet e altri dispositivi che potrebbero disporre di un chip Bluetooth) sfruttando il protocollo Bluetooth in due modalità: **offline** e **online**. Mentre la modalità online si comporta come una classica connessione stabilita tra due device, la modalità offline permette di scrivere dei messaggi anche con il Bluetooth non attivo e eventualmente spedirli quando esso viene attivato.

L'applicazione è composta da tre schermate principali: la pagina **iniziale**, la pagina che illustra sia **i dispositivi precedentemente accopiati** sia i dispositivi **nelle vicinanze** che sono disponibili a comunicare tramite Bluetooth e **le singole chat aperte** con ogni utente.

All'apertura dell'app, compare sullo schermo un messaggio che, nel caso in cui il **Bluetooth sia spento**, **informa** l'utente che questa applicazione opera tramite questo protocollo e chiede se si vuole utilizzare l'app in modalità offline o se si vuole attivare il Bluetooth (chiedendo il permesso al sistema Android).

Una volta fatto ciò, se ci troviamo al primo avvio, l'unica azione disponibile è quella di **aprire la pagina** dei dispositivi disponibili ed effettuare una scelta: nel caso in cui il dispositivo non sia stato accoppiato in precedenza con nessun device, possiamo sia effettuare una **ricerca** per trovare i dispositivi attivi nelle vicinanze, sia rendere il nostro dispositivo **disponibile** e **visualizzabile** per un certo **periodo di tempo** (*300s*).

Successivamente, possiamo scegliere tra tutti i dispositivi disponibili per avviare una comunicazione. Nel caso in cui la scelta ricadesse su un dispositivo **non-mobile**, come ad esempio delle *cuffie auricolari wireless*, sullo schermo apparirà un **messaggio** che informa l'utente che non può comunicare, così come lo stesso

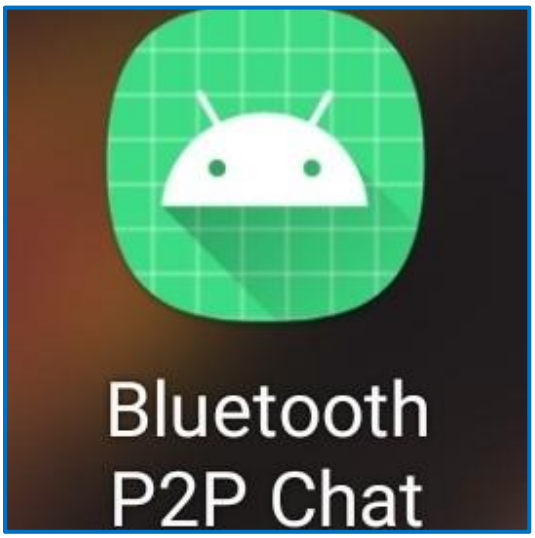
accade se tenta di stabilire un link con un device **obsoleto** che **non supporta** o **non possiede** il Bluetooth. Per avviare una comunicazione BT, ovviamente, entrambi i dispositivi devono possedere l'app e essere nelle condizioni di avviare una comunicazione stabile (nel caso l'app sia usata in modalità online-sincrona).

Una volta stabilita la comunicazione Bluetooth, i due device possono finalmente comunicare **liberamente** in entrambi i versi, come in una qualsiasi applicazione di chat disponibile sul Google Play Store o sull'App Store. Ovviamente, questa comunicazione online può perdurare fino a che due requisiti rimangono validi:

- I due dispositivi mantengono il **Bluetooth attivo**;
- I due dispositivi rimangono nel **range di comunicazione** supportato da quella versione specifica del Bluetooth usata per comunicare.

Se uno di questi due requisiti cessa di essere **vero**, la comunicazione passa in modo **automatico** alla modalità offline, la quale memorizza tutti gli eventuali messaggi spediti all'altro device nel database (Room) interno al processo del client, sfruttando il principio dell'architettura **P2P** seconda il quale ogni utente è il proprietario **responsabile** dei propri dati prima che vengono trasferiti agli altri utenti, essenzialmente agendo da proprio server integrato.

Tutta l'implementazione di questa applicazione si basa interamente sull'**API** Bluetooth gestita da **Google** per permettere agli sviluppatori di utilizzare e sfruttare il protocollo il quale è libero, semplice da usare e flessibile. Inoltre, è fondamentale anche l'inclusione del file `AndroidManifest.xml`, il quale ha funzione di primo strato dell'applicazione.



Innanzitutto, prima di iniziare il progetto bisogna scegliere quale versione del Bluetooth utilizzare tra Bluetooth "**classico**" e Low Energy. Poiché ci troviamo nel campo delle telecomunicazioni, la scelta risulta obbligata per il Bluetooth **classico**. Nel caso ci fossimo trovati per esempio nell'ambito **IoT**, il BLE sarebbe stata una scelta possibile e consigliata, poiché ci troviamo nel campo dei **sensori**, i quali hanno come prerequisiti quelli di consumare poca corrente e trasmettere dati semplici.

Ora sono presentate le principali funzioni e le primitive dell'API che sono state sfruttate per la realizzazione del progetto, direttamente dalla **documentazione ufficiale** (*in inglese*):

Using the Bluetooth APIs, an app can perform the following:

- Scan for other Bluetooth devices.
- Query the local Bluetooth adapter for paired Bluetooth devices.
- Establish RFCOMM channels.
- Connect to other devices through service discovery.
- Transfer data to and from other devices.
- Manage multiple connections.

Key classes and interfaces

All of the Bluetooth APIs are available in the `android.bluetooth` package. The following are the classes and interfaces you need in order to create Bluetooth connections:

`BluetoothAdapter`

Represents the local Bluetooth adapter (Bluetooth radio). The `BluetoothAdapter` is the entry-point for all Bluetooth interaction. Using this, you can discover other Bluetooth devices, query a list of bonded (paired) devices, instantiate a `BluetoothDevice` using a known MAC address, and create a `BluetoothServerSocket` to listen for communications from other devices.

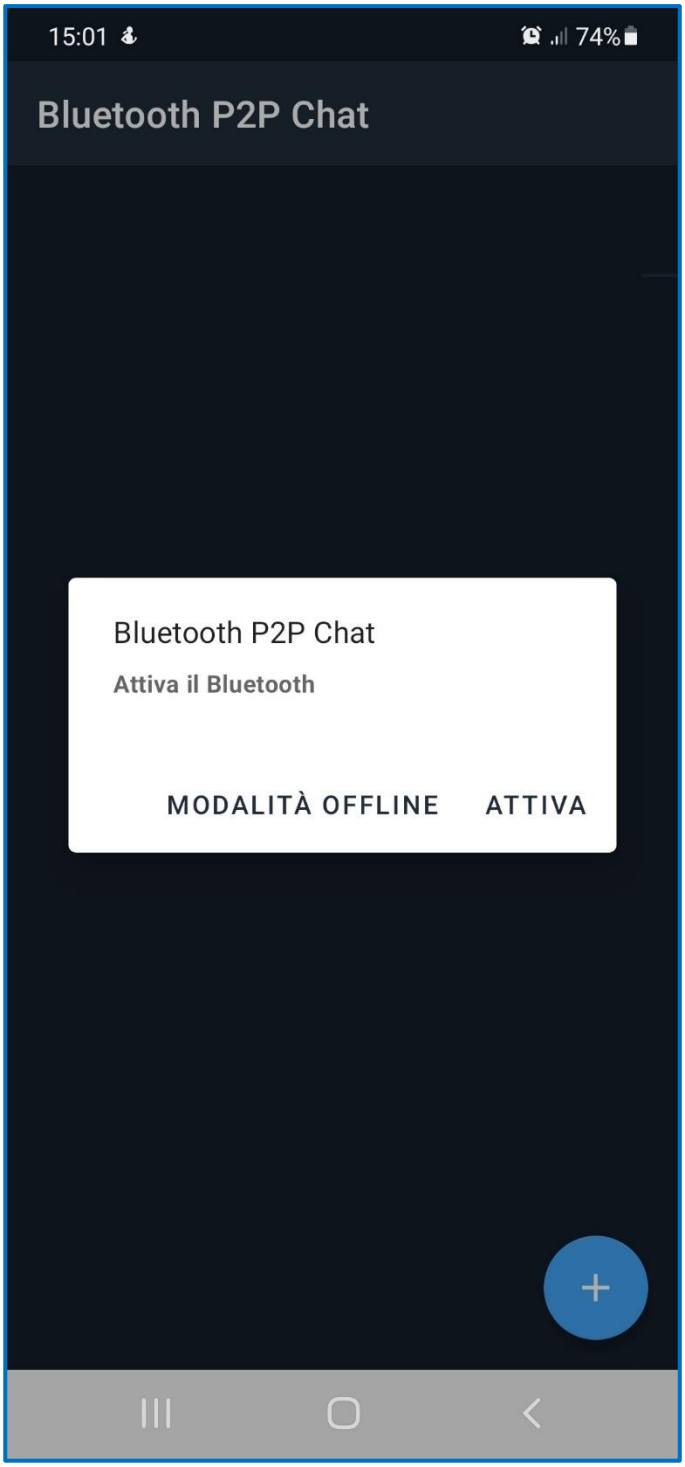
`BluetoothDevice`

Represents a remote Bluetooth device. Use this to request a connection with a remote device through a `BluetoothSocket` or query information about the device such as its name, address, class, and bonding state.

`BluetoothSocket`

Represents the interface for a Bluetooth socket (similar to a TCP `Socket`). This is the connection point that allows an app to exchange data with another Bluetooth device using `InputStream` and `OutputStream`.

Per concludere la trattazione critica dell'argomento che è stato selezionato per il progetto e della sua implementazione, verranno ora presentate le schermate che l'utente finale visualizzerà quando utilizzerà l'applicazione:




15:01
74%
Bluetooth P2P Chat
Galaxy S9 di Rachele
Prova prova prova


15:01
74%
Galaxy S9 di Rachele
Prova prova prova 19:03
Messaggio

Use cases e esempi d'uso

Ogni applicazione deve avere bene in mente quali possono essere i possibili casi d'uso specifici per lo sviluppo o la messa in commercio, i quali devono essere generalmente pensati e ben definiti **prima di iniziare** a sviluppare il progetto, almeno nelle loro **specifiche minime**.

I casi d'uso che ho pensato per questa applicazione e che mi hanno scelto a sviluppare questo progetto, oltre alla personale curiosità di lavorare e sfruttare il Bluetooth, riguardano ovviamente ambiti **particolari** della **comunicazione**.

Il problema della comunicazione mobile, sia **telefonica** che tramite comuni **app di messaggistica** che sfrutta la rete Internet/**Wi-Fi**, è che, pur essendo il sistema dominante, **stabile**, economico e sicuramente destinato a durare, può risultare in particolari situazioni che sia **indisponibile** o addirittura negato. Tutto ciò è dovuto al fatto che le torri telefoniche sono essenzialmente un Single Point of Failure per le **zone geografiche** a cui forniscono servizi (ovviamente sono sistemi molto sicuri e stabili, non si sta cercando di mettere in dubbio l'utilità di queste infrastrutture ma di provare a trovare possibili alternative come esperimento mentale).

Alcuni di questi esempi possono essere **disastri naturali**. Un caso su tutti può essere il recente **uragano** che si è scatenato su **Puerto Rico**, rendendo necessari lavori durati mesi per ricostruire le infrastrutture. Un altro possibile uso può essere la comunicazione in **aree rurali**, popolate in modo sparso e senza un supporto costante per scambiare messaggi. Al contrario, in eventi con una **grande densità** di **persone** concentrate nello stesso punto come *concerti*, *conventions*, *proteste*, *eventi sportivi* e altri le **infrastrutture** potrebbero **cedere**. Un ultimo importante utilizzo, da non trascurare, potrebbe essere in nazioni come la **Cina** o la **Corea del Nord**, dove la **libertà di parola** e espressione non è garantità e le infrastrutture sono **monitorate** e sotto **censura**. Quindi, tramite la comunicazione Bluetooth potrebbe essere possibile **aggirare** questi sistemi opprimenti.

Ovviamente rimane fondamentale ricordare la pesante **limitazione** dovuta dal **corto raggio** del protocollo Bluetooth, il quale rende difficili comunicazioni agli angoli di una nazione.

Considerazioni finali e conclusione

Il compimento di questo progetto ha dato modo, partendo da conoscenze base degli argomenti, di apprendere in modo dettagliato il funzionamento **pratico** del **Bluetooth**, le varie tecnologie su cui si basa, l'infrastruttura e il **valore** di questo protocollo, il quale è semplice e leggero ma **fondamentale** nell'uso quotidiano di migliaia di dispositivi in ogni ambito, senza il quale la realizzazione di tantissimi **servizi** e **device** sarebbe **impossibile**. Anche se presenta alcune **limitazioni** di potenza, propagazione e latenza resta di infinita importanza.

Inoltre ha dato la possibilità di avere una **panoramica** più **ampia** e dettagliata riguardo la direzione verso la quale si sta spingendo questo protocollo, le nuove tecnologie e i nuovi sviluppi del Bluetooth e della branca dello sviluppo software che riguarda l'ambiente Android.

Inizialmente ho trovato **qualche difficoltà** data la **novità** del linguaggio adottato e l'integrazione con l'**API Bluetooth**, ma questi ostacoli sono stati superati nel corso dello sviluppo del progetto.

Per concludere è innegabile constatare la soddisfazione di aver svolto questo tipo di ricerca al fine di **arricchire** le **conoscenze**, sia **teoriche** che **pratiche**. In particolare, la possibilità di sviluppare un'applicazione partendo da conoscenze base, imparando un nuovo linguaggio di programmazione e un nuovo ambiente di sviluppo, **applicando la teoria del corso di Laurea in un contesto pratico**.

Bibliografia e sitografia

Teoria:
Materiale del corso del prof. Ing. Antonio Della Selva

Android Studio:
https://developer.android.com/studio

Kotlin:
https://developer.android.com/kotlin
https://it.wikipedia.org/wiki/Kotlin_(linguaggio_di_programmazione)

Bluetooth – Storia, SIG, loghi:
https://www.bluetooth.com/

Bluetooth – Aspetti pratici e implementativi:
https://developer.android.com/guide/topics/connectivity/bluetooth/setup
https://developer.android.com/reference/android/bluetooth/BluetoothManager

Architettura dell'app – P2P, MVVM:
https://it.wikipedia.org/wiki/Peer-to-peer
https://developer.android.com/topic/architecture?gclid=CjwKCAjw_ISWBhBkEiwAdqxb9sPlpwFAmmGmy0DNoAtsYDzP2w8NmxhTOp39k5LxnbXAbq9R7wvzsBoCiuEQAvD_BwE&gclsrc=aw.ds

Design dell'app:
https://material.io/design

Device usati per testare l'app:
https://www.samsung.com/it/