Documentazione progetto LAP2.

Specifiche

E' stato richiesto lo sviluppo di un'app che consenta di connettere remotamente due device affinché uno possa trasmettere all'altro dati raccolti dalla fotocamera, giroscopio, microfono e gps. In particolare la trasmissione dei sopracitati dati deve essere effettuata soltanto da un device verso l'altro e non viceversa, pertanto possono essere identificati due soggetti differenti: un device controller ed un device target. Il device controller attraverso un'interfaccia grafica minimale dà all'utente un set di comandi relativi a quale tipo di dato si voglia ricevere dal device controller; invece il device target eseguirà in maniera silente i comandi ricevuti dal device controller permettendone altresì l'esecuzione in background mode ovvero a display bloccato e display spento.
Le richieste includono anche lo sviluppo dell'applicazione per i dispositivi Android attraverso il framework React Native.

Strumenti utilizzati e metodologia di sviluppo

Framework e linguaggio

React Native è un framework per lo sviluppo di applicazioni su piattaforme native che permette di unificare l'esperienza di sviluppo su diversi sistemi operativi.
React Native è un progetto basato su ReactJS, che adotta un nuovo approccio allo sviluppo Web. Esso permette di sviluppare con il linguaggio di programmazione Java Script arricchito con JSX ovvero un linguaggio per la strutturazione del layout della GUI attraverso tag XML.

Device target & SDK

Dal momento che non c'erano specifiche particolari sui device da utilizzare, su quali versioni di Android andare a sviluppare, come SDK è stato scelto il range [5.0 - 9.0] e quindi le versioni SDK [21.0 - 28.0].
I test sono stati effettuati su:

- Emulatore Genymotion Google Nexus 6 con versione 8.0, API 26
- Asus Zenfone 2 ZE551ML con versione 5.0, API 21
- Samsung Galaxy S2 plus con versione 5.1, API 22
- Samsung Galaxy S9 con versione 9, API 28.0

Tecnologie e librerie di terze parti utilizzate

- react-native : libreria standard di React Native per le principali funzioni
- websocket: per la connessione al signaling server
- react-native-webrtc: per stabilire una connessione p2p tra device target e device controller
- react-native-qrcode: per la generazione di un codice QR a partire da una stringa in input
- react-native-qrcode-scanner: per l'acquisizione di informazioni a partire da un codice QR scansionato con la fotocamera
- react-native-sensors: per accedere all'hardware fornito dai dispositivi android
- react-native-maps: api di google-maps per visualizzare la mappa
- react-native-foreground-service: per mantenere l'applicazione in esecuzione anche quando essa viene killata.

Signaling server

Per la fase di pairing i due device devono consentire la rilevazione del proprio indirizzo di rete dal momento che essi molto probabilmente si troveranno in una sottorete nattata con un ip pubblico. Per fare ciò viene impiegato un servizio minimale che ha come unico scopo quello di rendere i due device rispettivamente visibili sulla rete. Questo è ottenuto dapprima dal device controller, che non appena genera il QR code si collega al signaling server fornisce tutte le caratteristiche della propria rete e comunica inoltre l'username con cui esso potrà essere raggiunto dal device target; il device target appena acquisisce il QRcode farà altrettanta procedura. Il Signaling server è ospitato nei server di open shift.

Panoramica generale

Work flow

Dopo che l'app viene avviata si potrà scegliere quale soggetto assegnare al device: controller o target. Il target è il device che vorrà essere controllato mentre il controller è quello che vorrà prendere il controllo dell'altro.
In generale questa operazione viene fatta in contemporanea cosicché possa i due device possano essere associati subito. Quindi si seleziona controller, l'app tenterà di collegarsi con il signaling server e, dopo che avrà stabilito la connessione, verrà generato un QR code che servirà al device target per autenticarsi con il signaling server e poi con il controller.
Nel device target si aprirà come prima cosa uno scanner di QRCODE e quindi si punterà verso il codice generato dal controller. Dopo l'acquisizione avverrà la fase di pairing e se ha successo allora nel target apparirà la schermata che consiglia l'utente di chiudere l'app (in realtà avrà un foreground service che fin quando non verrà bloccato l'app rimarrà in esecuzione; contemporaneamente nel device controller si aprirà la console che permetterà al di inviare i comandi per ricevere i relativi dati (streaming audio/video, posizione gps, giroscopio)

Descrizione delle funzionalità

Connettività

Per stabilire la connessione tra i due device è stata utilizzata la tecnologia webrtc la quale permette di creare uno straming realtime.

webRTC permette di stabilire una connessione p2p e quindi senza la necessità di un server intermediario. Il problema principale in una connessione p2p è la visibilità che i dispositivi hanno in rete. Dal momento che la maggior parte dei dispositivi sono dietro una connessione nattata è necessario conoscere l'ip pubblico con cui si affacciano alla rete. Questo problema webRTC lo risolve lasciando libero arbitrio agli sviluppatori di costruire un qualche canale di comunicazione che permette ai dispositivi che vogliono partecipare di scambiarsi i relativi ip pubblici con cui essi poi dovranno comunicare. La connessione viene stabilita attraverso gli ICE candidate che sono dei pacchetti di informazioni circa lo stato degli apparati in cui si trova il dispositivo; al momento che si vuole stabilire la connessione il protocollo di webRTC inizierà una mediazione attraverso questi pacchetti per testare la velocità di connessione e altre caratteristiche riguardo l'hardware del dispositivo; una volta negoziate le caratteristiche della rete e l'hardware dei device la connessione i dispositivi inizieranno a comunicare. Inoltre lo scambio di questi pacchetti avviene quando si apre lo streaming audio/video o di dati.

In questo progetto il signaling server è stato realizzato attraverso una websocket che rimane in ascolto su un web server creato adhoc sui server di openshift. Data la natura del progetto non è stata implementata una vera e propria autenticazione infatti i device che vogliono partecipare alla connessione si registrano a tale server soltanto attraverso username generato randomicamente dal controller.

La connessione dei due dispositivi avviene con il seguente iter:

1. Il device controller genera due stringhe casuali che rappresenteranno l'username di se stesso e l'username del device target.
2. Il device controller si connetterà al signaling server attraverso una web socket e si registrerà ad esso con l'username generato da se stesso. (Questo implica che ogni volta che si inizializza l'app verrà generato un nuovo user name e quindi avverrà una nuova registrazione. Questo comportamento dovrebbe essere sistemato se si decide di voler mantenere attiva la connessione anche quando killa l'app del tutto)
3. Il device controller genererà un QRcode a partire da un array di due elementi contente come primo elemento l'username del controller e come secondo elemento l'username del target.

4. Dopo che il target acquisisce username-Target e username-Controller con la scansione del QRcode effettuerà la registrazione al web server con l'username-Target; dopo che avviene questo il device target inizializza il protocollo di webRTC.
5. Il device controller e il device target adesso sono connessi ed inizializzano lo streaming dati per scambiarsi messaggi.

In particolare il protocollo di webRTC include tre canali di comunicazione che sono dati, audio e video. Il canale dati è usato per lo scambio di messaggi; i messaggi che il controller invia al target rappresentano i comandi che il target dovrà eseguire; i messaggi che il target invia al controller rappresentano i dati acquisiti dai sensori. Per quanto riguarda lo streaming audio/video vengono utilizzati i rispettivi canali che vengono aperti quando il controller invia il relativo comando.

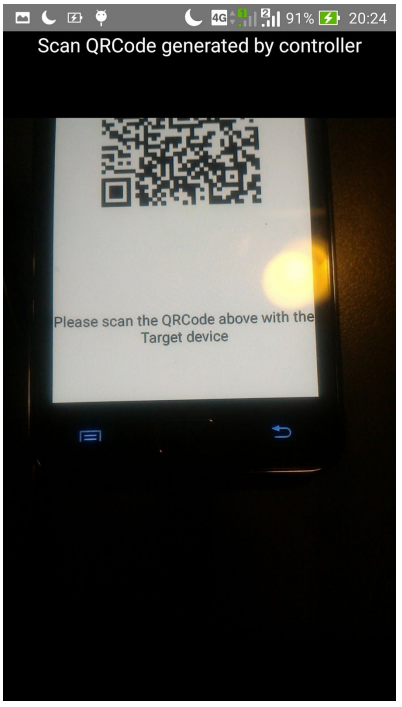
Background mode

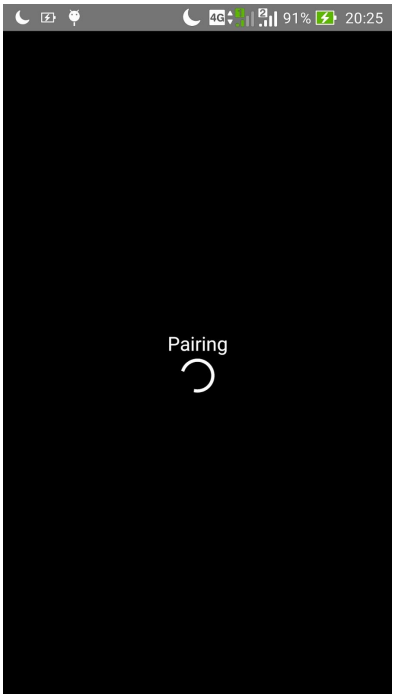
Date le numerose limitazioni imposte da parte di Android, soprattutto ultime versioni, è stata necessaria l'adozione di stratagemmi affinché l'app una volta premuto il tasto home essa possa continuare a funzionare. Praticamente il problema è che ho voluto risolvere è stato quello che è una volta che la venga posizionati in background, ovvero dopo la pressione del tasto home, di non permettere ad Android di rimuovere l'applicazione dallo scheduler. Per fare ciò , ho adottato una libreria chiamata nome libreria che gli consente di rimanere in esecuzione mantenendo però una notifica fissa indica all'utente che l'applicazione è in esecuzione. quando il device target riceve il comando per disconnettersi allora l'applicazione il foreground service e quindi sparirà la notifica. Un idea sarebbe quella di entrare nelle impostazioni delle notifiche e togliere la spunta affinché all'utente non venga mostrata più la fastidiosa notifica. In questo modo si risolve il problema.

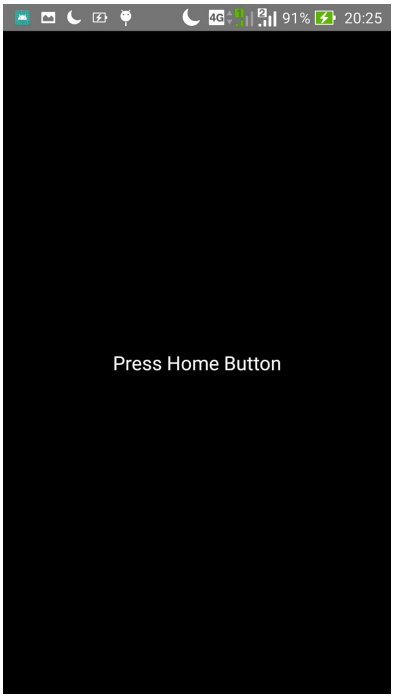
Screenshot and navigation

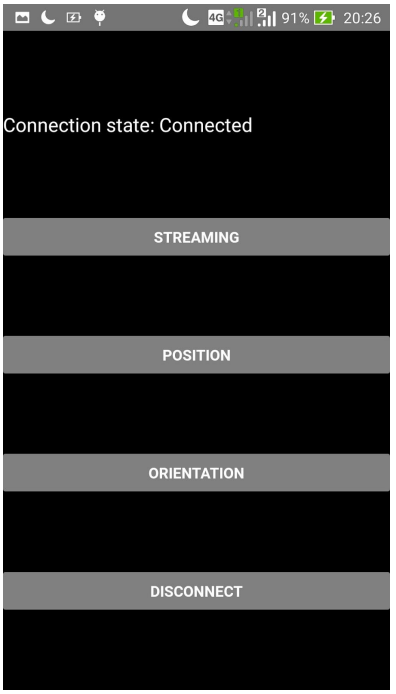

92% 20:23
SpyApp
I'M CONTROLLER
I'M TARGET
SIGNALING SERVER ADDRESS
ws://2.226.180.184:9090

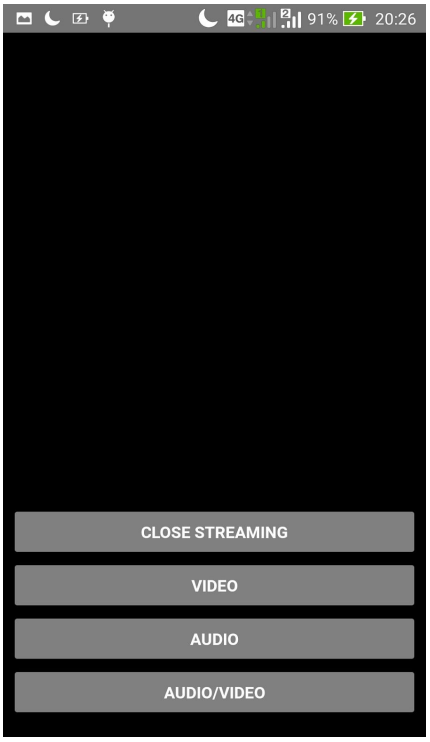

92% 20:23
Please scan the QRCode above with the Target device

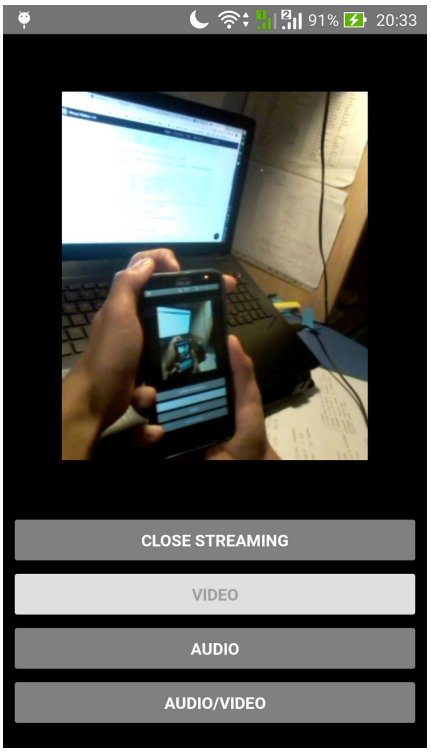

91% 20:24
Scan QRCode generated by controller
Please scan the QRCode above with the Target device

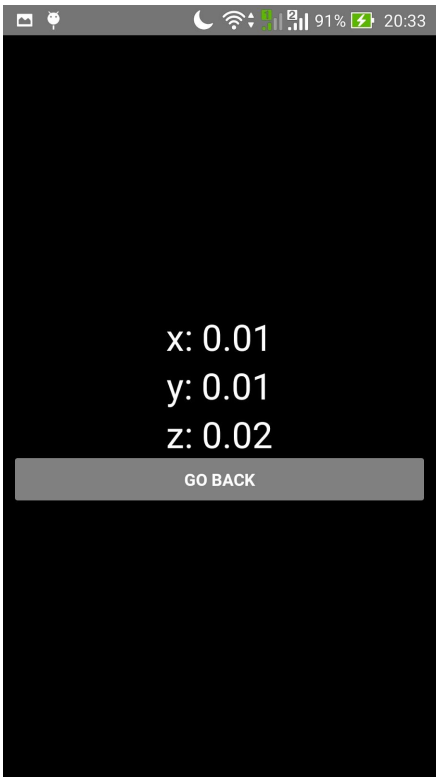

91% 20:25
Pairing

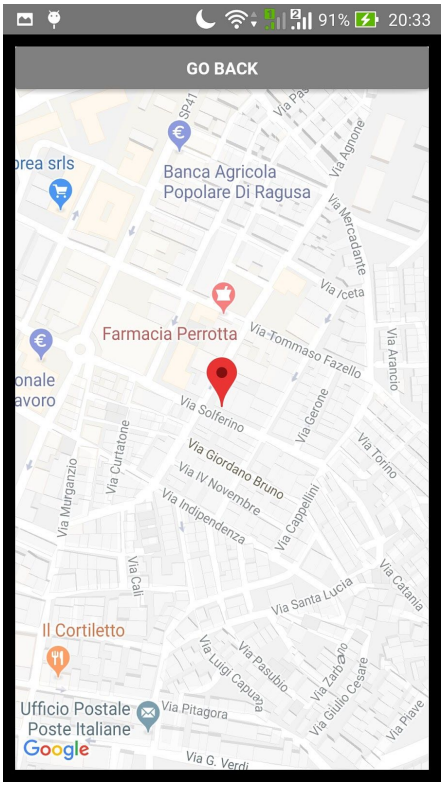

91% 20:25
Press Home Button


91% 20:26
Connection state: Connected
STREAMING
POSITION
ORIENTATION
DISCONNECT


91% 20:26
CLOSE STREAMING
VIDEO
AUDIO
AUDIO/VIDEO


91% 20:33
CLOSE STREAMING
VIDEO
AUDIO
AUDIO/VIDEO


91% 20:33
x: 0.01
y: 0.01
z: 0.02
GO BACK


91% 20:33
GO BACK
Banca Agricola Popolare Di Ragusa
Farmacia Perrotta
Il Cortiletto
Ufficio Postale Poste Italiane
Google
Via Solferino
Via Giordano Bruno
Via IV Novembre
Via Indipendenza
Via Cappellini
Via Santa Lucia
Via Pitagora
Via G. Verdi