MANUALE DI PROGETTO

Forgia Paolo

SSSE SIG

INDICE

1 ABSTRACT

La Scuola d'arti e mestieri di Bellinzona sta sviluppando un piccolo rover pilotato da un radiocomando che comunica mediante una connessione Bluetooth. L'idea è quella di pilotare il rover mediante uno smartphone e non più utilizzare un radiocomando dedicato.

Il progetto consiste nello sviluppare un'applicazione mobile per pilotare un rover attraverso dei comandi inviati tramite Bluetooth. Il rover a sua volta invierà dei comandi al telefono, i quali daranno delle informazioni sullo stato della batteria e sulla distanza da un ostacolo.

Si sono dovuti escludere i dispositivi iOS siccome per comunicare tramite vecchie versioni di Bluetooth con dispositivi Apple è una certificazione e il modulo Bluetooth in uso non la possiede.

Per quanto riguarda i telefoni Android, l'obiettivo è stato raggiunto e si può pilotare il rover in tutte le sue funzionalità. Si è anche avuto il tempo di iniziare lo sviluppo di una nuova funzionalità "Programs", la quale dà la possibilità di predefinire un elenco di comandi da eseguire. Questa funzionalità è però ancora incompleta.

Alcuni possibili sviluppi futuri:

- Completare la funzionalità "Programs"
- Aggiungere il supporto ai tablet
- Aggiungere il supporto alla modalità portrait (orizzontale)
- Aggiornare il modulo Bluetooth con uno con il supporto ai dispositivi Apple e completare l'applicazione in modo da funzionare anche sugli iPhone

In conclusione, il progetto è stato un successo, come descritto nelle conclusioni, si è completa la feature principale nel tempo e funziona correttamente.

2 INTRODUZIONE

2.1 SCOPO

La Scuola d'arti e mestieri di Bellinzona sta sviluppando un piccolo rover pilotato da un radiocomando che comunica mediante una connessione Bluetooth. Nell'anno scolastico 2020/21 uno studente in elettronica del quarto corso ha sviluppato un primo prototipo funzionante del rover e del radiocomando.

Visto l'interesse suscitato da questo progetto, si è deciso di dar seguito allo sviluppo di un nuovo prototipo dove anche l'intera parte meccanica sarà sviluppata e realizzata in sede. In questa seconda fase l'idea è quella di pilotare il rover mediante uno smartphone e non più utilizzare un radiocomando dedicato.

2.2 PER CHI

Il progetto è stato commissionato da Daniele Kamm, della CPT di Bellinzona, scuola d'arti e mestieri (SAM).

2.2.1 Organigramma

Direttore	M. Zanella
Vicedirettore	M. Mozzini
Capilaboratorio	A. Keller: elettrotecnica
Insegnante/Laboratorio	D. Kamm

Mi sono rapportato con Daniele Kamm per il progetto e lui poi riferiva il tutto al capolaboratorio di elettronica A. Keller.

È disponibile anche l'organigramma completo della CPT di Bellinzona all'interno degli Allegati.

2.3 DESCRIZIONE PROGETTO

Il progetto consiste nello sviluppare un'applicazione mobile per pilotare un rover attraverso dei comandi inviati tramite Bluetooth.

L'applicazione dovrà avere una schermata nella quale sono presenti tutti gli elementi necessari per interagire con il rover in modo rapido e intuitivo.

Allo stesso tempo il rover invierà all'app dei comandi, i quali identificano dei valori, come ad esempio la percentuale della batteria, e anch'essi dovranno essere mostrati nella schermata insieme ai comandi.

2.4 OBIETTIVI

Sviluppare un applicativo per Android e iOS che vada a rimpiazzare il telecomando utilizzato al momento.

Questo applicativo dovrà connettersi al dispositivo e inviare i comandi al rover per pilotarlo. Allo stesso momento riceverà costantemente dati dal rover riguardo la percentuale della batteria e distanza da un ostacolo.

Le informazioni ricevute verranno poi mostrate nell'applicazione come informazione per l'utente.

Attraverso l'interfaccia dell'applicazione si dovrà essere in grado di gestire il rover in modo esaustivo.

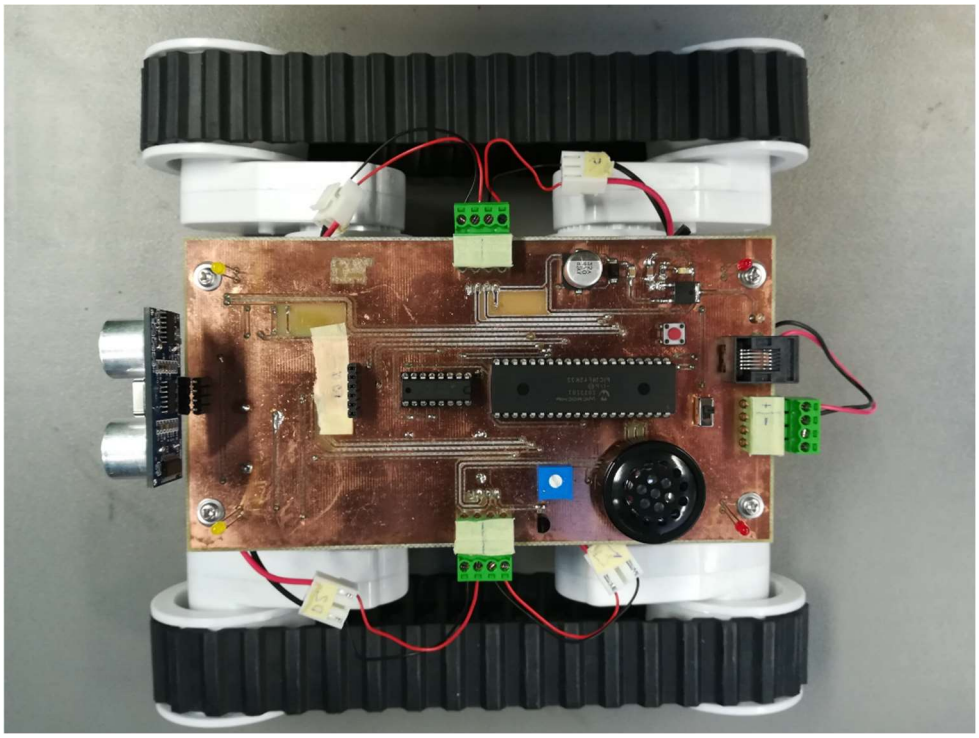


Figure 1 - Rover da controllare

2.4.1 Funzionalità aggiunta

Durante il progetto si è notato che si sarebbe concluso il tutto in anticipo rispetto ai tempi e si è aggiunta una nuova funzionalità: la possibilità di predefinire un elenco di comandi da eseguire (Programs), avendo così la possibilità di controllare il rover in modo autonomo.

Ci sarebbe una schermata, la quale darebbe all'utente la possibilità di definire il comando da eseguire e per quanto tempo. Una volta terminato avrebbe interrotto l'invio del comando e sarebbe passato a quello successivo.

2.5 Benefici attesi

Tra i benefici attesi c'è un miglioramento all'usabilità da parte dell'utente finale, il quale tramite un'interfaccia chiara e funzionale, potrà interagire con il rover più comodamente.

Tutte le funzioni saranno disponibili su un'unica schermata e verranno visualizzate anche le informazioni provenienti dal rover, quali stato della batteria e distanza da un ostacolo.

Gli utenti potranno poi scaricare l'applicativo e installarlo sul proprio telefono così da poter controllare il rover. Con questo sistema di andrà ad eliminare il problema di un unico telecomando, il quale se dovesse scaricarsi o rompersi renderebbe impossibile interfacciarsi con il rover.

Inoltre, si avrebbe la possibilità di aggiungere funzionalità o migliorare quelle presenti, molto più facile tramite un semplice aggiornamento dell'applicazione.

2.6 RISULTATO OTTENUTO

L'applicazione funziona correttamente e può controllare il rover in tutte le sue funzioni. Inoltre, riceve ed interpreta correttamente i valori inviati dal rover.

C'è però da tenere in conto che l'applicazione è stata sviluppata solo per i telefoni Android e non iOS. Questo è dovuto ad una limitazione legata alle certificazioni rilasciate da Apple e al modulo Bluetooth utilizzato.

L'applicazione è stata conclusa in anticipo, per questo motivo si è aggiunta una funzionalità, la quale si è decisa con il cliente verso la metà del progetto.

2.6.1 Programs

La parte relativa la creazione di programmi è ancora in fase di sviluppo, non è ancora utilizzabile.

2.1 LIMITAZIONI

Una limitazione che è sopraggiunta subito dopo un'analisi delle tecnologie è legata supporto per l'applicazione ad iOS. Siccome per comunicare tramite vecchie versioni di Bluetooth con dispositivi Apple è necessaria la certificazione MFi[1], si è deciso di scartare il supporto ai dispositivi Apple.

Nonostante il supporto ad iPhone è stato rimosso si è deciso comunque di rimanere su tecnologie multipiattaforma, ovvero che hanno la possibilità di creare applicazioni sia per Android che per iOS. Questo soprattutto perché i vantaggi dello sviluppo nativi sono soprattutto legati alle performance, e l'applicazione in questione è molto leggera.

Inoltre, se in futuro la situazione dovesse evolvere e l'implementazione su iOS dovesse diventare una possibilità concreta, avendo sviluppato con tecnologie come Xamarin o React Native, il passaggio ad iOS è molto rapido.

[1] Vedi approfondimento nel Manuale Tecnico, sezione relativa a Bluetooth

3 VISION

PER	Forgia Paolo	cliente
IL QUALE	Controllare via Bluetooth il rover	necessità
IL	CPTRoverApp	nome prodotto
È UN	Applicazione mobile	categoria
CHE	Da la possibilità di controllare in modo comodo e rapido	benefici
	tutte le funzionalità del rover.	
CONTRARIAMENTE A	Il radiocomando	competitors
IL NOSTRO PRODOTTO	Visualizza le informazioni inviate del rover in modo comodo e rapido	valore aggiunto
	Permette di controllare tutte le funzionalità comodamente	
	Ha una migliore usabilità	
	La disponibilità non è più legata alla quantità di radiocomandi	

4 SITUAZIONE ATTUALE

4.1 COME VIENE FATTO IL LAVORO ORA

Al momento entrambe le schede elettroniche imbarcano un microcontrollore e un modulo Bluetooth per la comunicazione. Sulla scheda del rover sono presenti inoltre: un buzzer, un sensore di distanza ad ultrasuoni, quattro LED indipendenti e l'elettronica necessaria all'azionamento dei motori. Sul radiocomando, invece, sono stati previsti: due joystick a due assi (X/Y), quattro pulsanti, quattro LED e un display LCD.

La comunicazione tra i due dispositivi avviene mediante lo scambio di stringhe di testo debitamente formattate. I due moduli Bluetooth, accoppiati tra loro in precedenza, gestiscono la comunicazione in modo trasparente per i microcontrollori che, dalla loro prospettiva, scambiano semplici caratteri ASCII attraverso la propria interfaccia seriale.

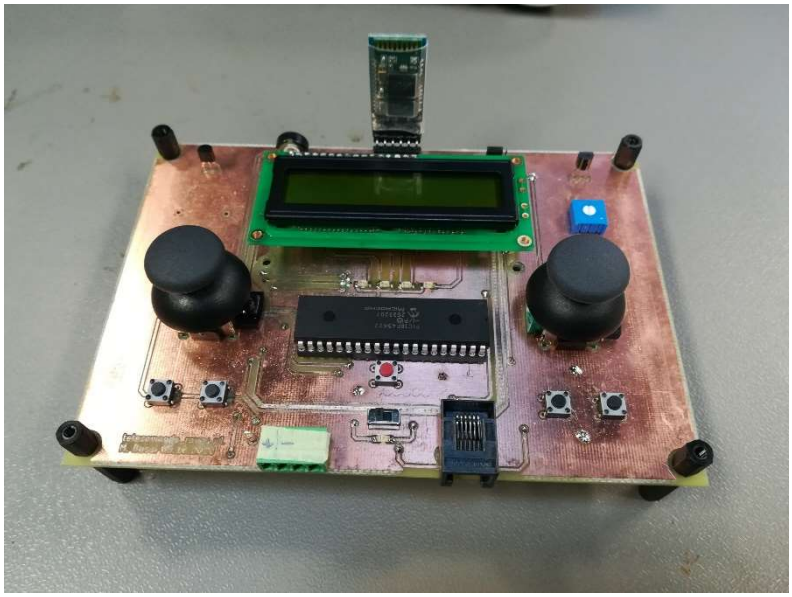


Figure 2 - Radiocomando

4.2 MAPPA PROCESSI AZIENDALI

Non è stata utilizzata nessuna mappa dei processi aziendali siccome non c'era nessuna relazione relativa a questo tipo di progetto.

4.3 ANALISI DOCUMENTI INTERNI

All'inizio del progetto mi sono stati consegnati i seguenti documenti (trovabili sotto "Allegati"):

- Annuncio di progetto CPT Cingolato.docx
- Quaderno dei compiti.pdf
- Progetto Rover.zip

"Progetto Rover.zip" contiene tutti i files e i documenti creati dallo studente che ha realizzato il rover e il radiocomando. Leggere questi documenti è stato importante soprattutto nella fase iniziale per comprendere il funzionamento di varie parti della comunicazione tra radiocomando e rover.

4.4 CONFRONTO VECCHIO-NUOVO

Per quanto riguarda il rover non ci sarà differenza tra l'essere controllato dal telecomando meccanico o dall'applicazione.

L'unica differenza che c'è è che il bitrate è cambiato da 9'600 a 115'200 bit/secondo. Ovvero la comunicazione che avviene tra il microcontrollore (MCU) e il modulo Bluetooth (MC-05).

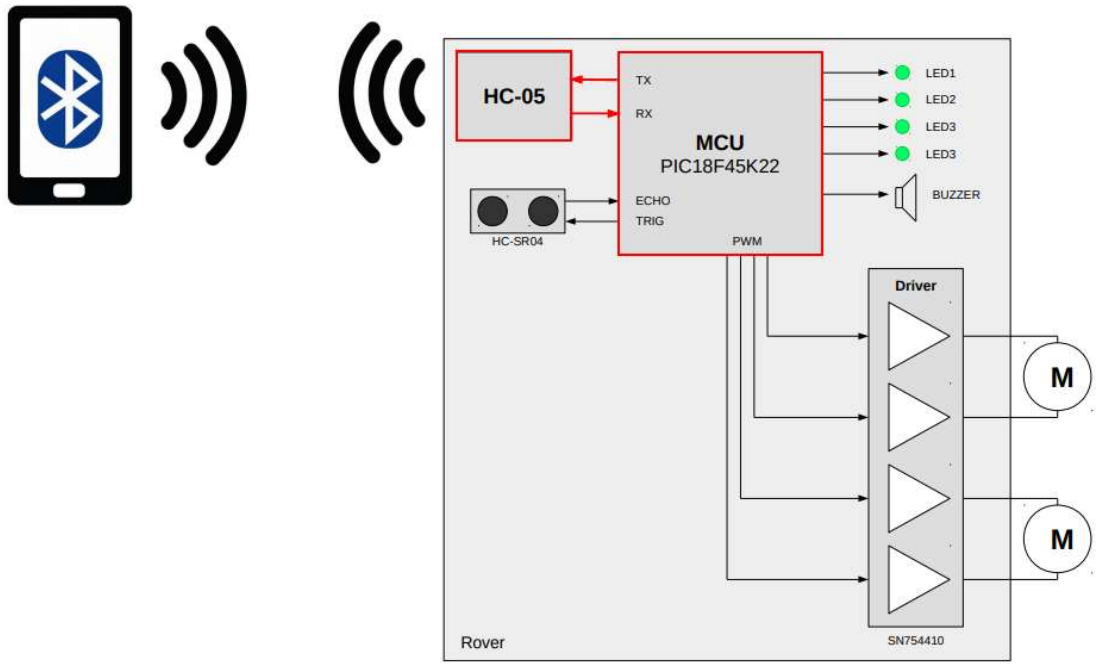


Figure 3 - Evidenziato in rosso la comunicazione tra MCU e HC-05

5 ANALISI

5.1 MACRO REQUISITI

5.1.1 Dispositivi da supportare

Il target è il più ampio numero di smartphone recenti. Si parla di telefoni che sono ancora ampiamente in uso, nel nostro caso si parla di smartphone Android e iOS.

Vista l'attuale limitazione del modulo Bluetooth nei confronti dei dispositivi iOS rimangono solamente i dispositivi Android. L'applicazione deve essere compatibile anche con le precedenti versioni di Android, idealmente dalla versione Android 4.4 in avanti così da andare a supportare il 99.6% dei dispositivi in commercio[2].

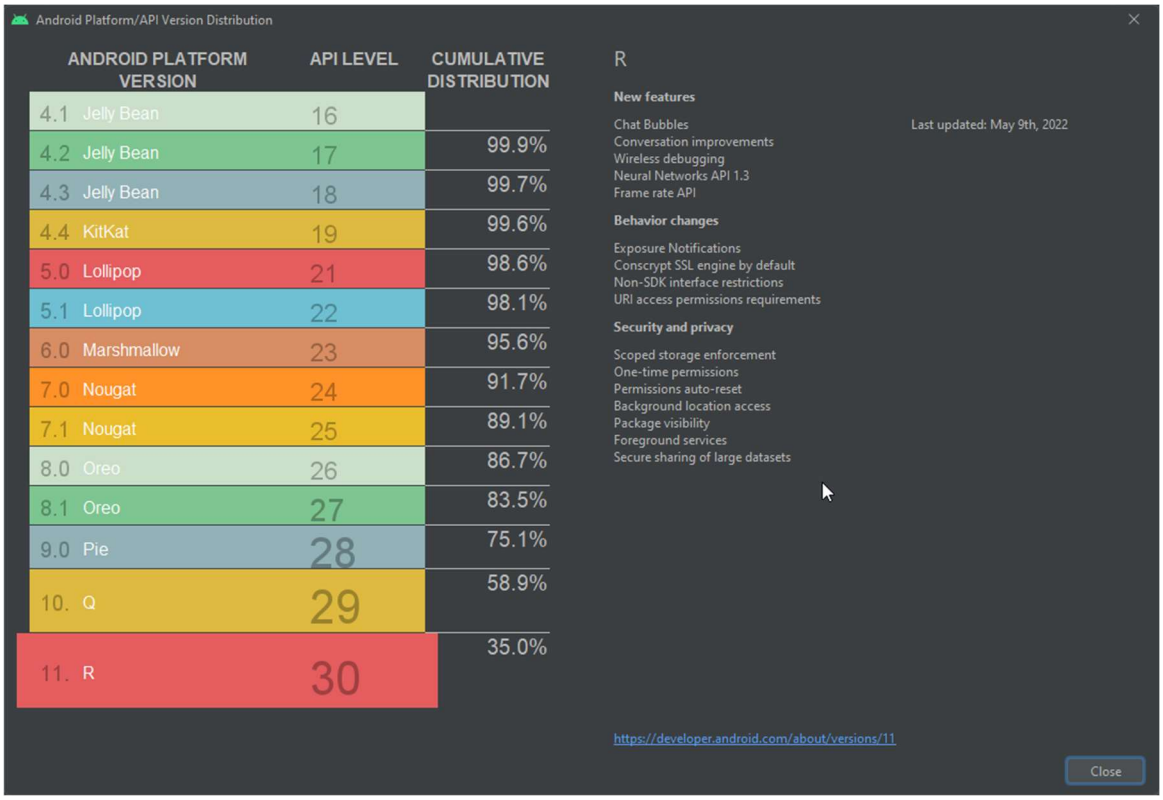


Figure 4 - Distribuzione delle versioni di Android

5.1.2 Comunicazione con il rover

L'applicazione deve inviare e ricevere comandi dal rover, questi comandi avranno un formato predefinito:

- 8 bit di dati
- Nessuna parità
- 1 bit di stop

I comandi avranno la seguente struttura:

- Una lettera maiuscola che identifica il comando (rosso)
- Un numero compreso da 0 a 255 che identifica il valore (blu)

[2] https://9to5google.com/2022/05/20/android-2022-distribution-numbers-chart/

- Il bit di stop (verde)

5.2 USE CASE

5.2.1 Use Case di contesto

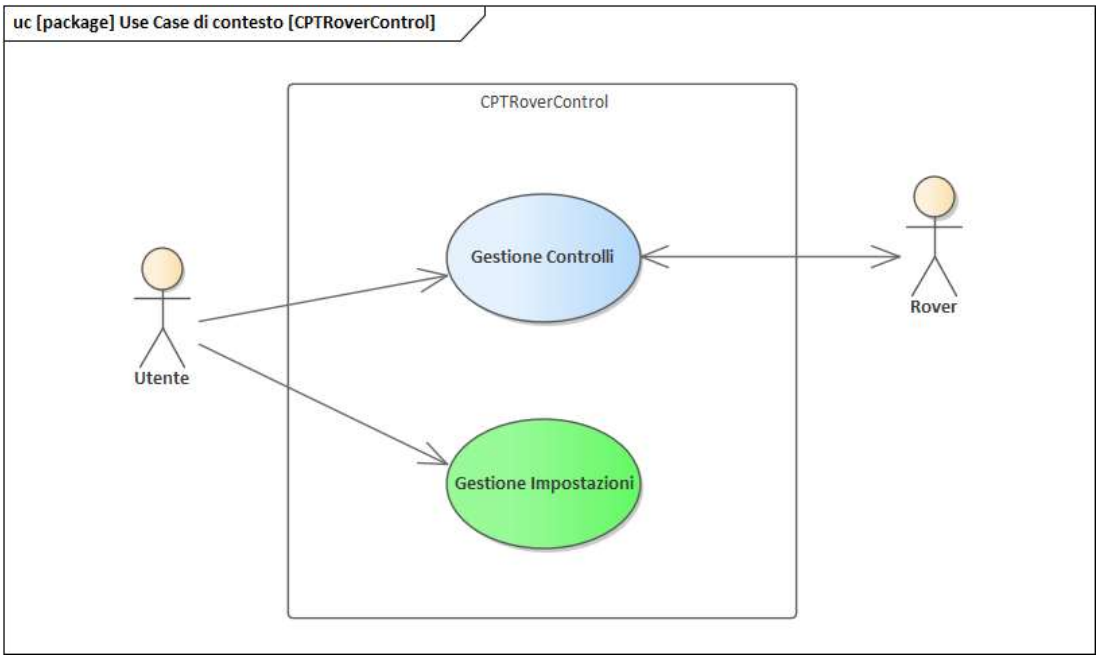


Diagramma di contesto delle sezioni principali dell'applicazione.

5.2.2 Controlli

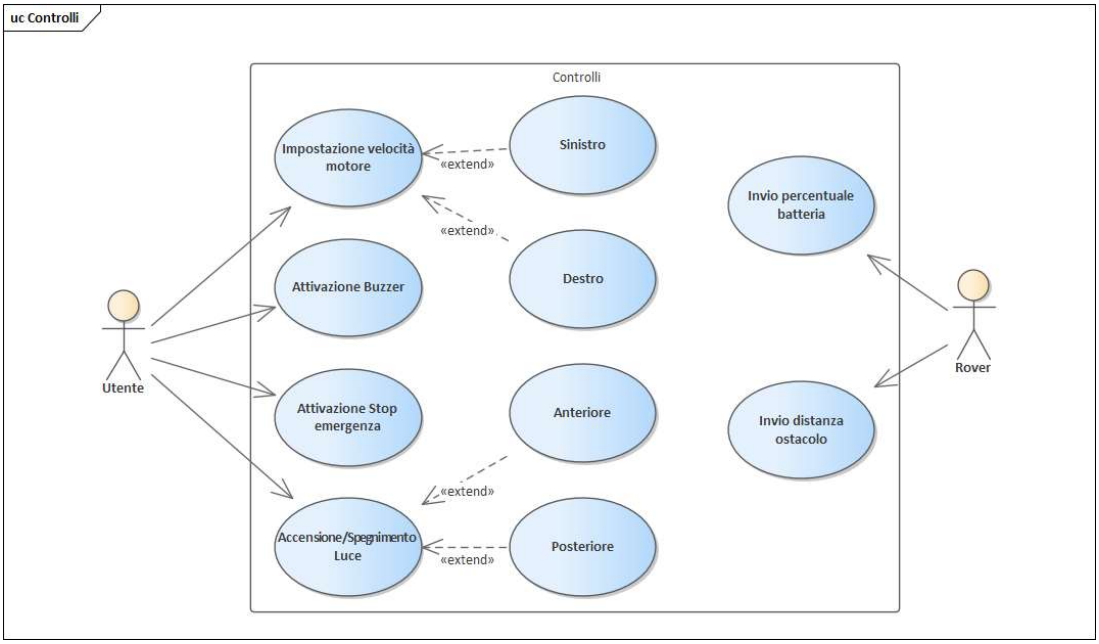


Diagramma dei controlli che vengono scambiati tra applicazione e rover.

5.2.3 Impostazioni

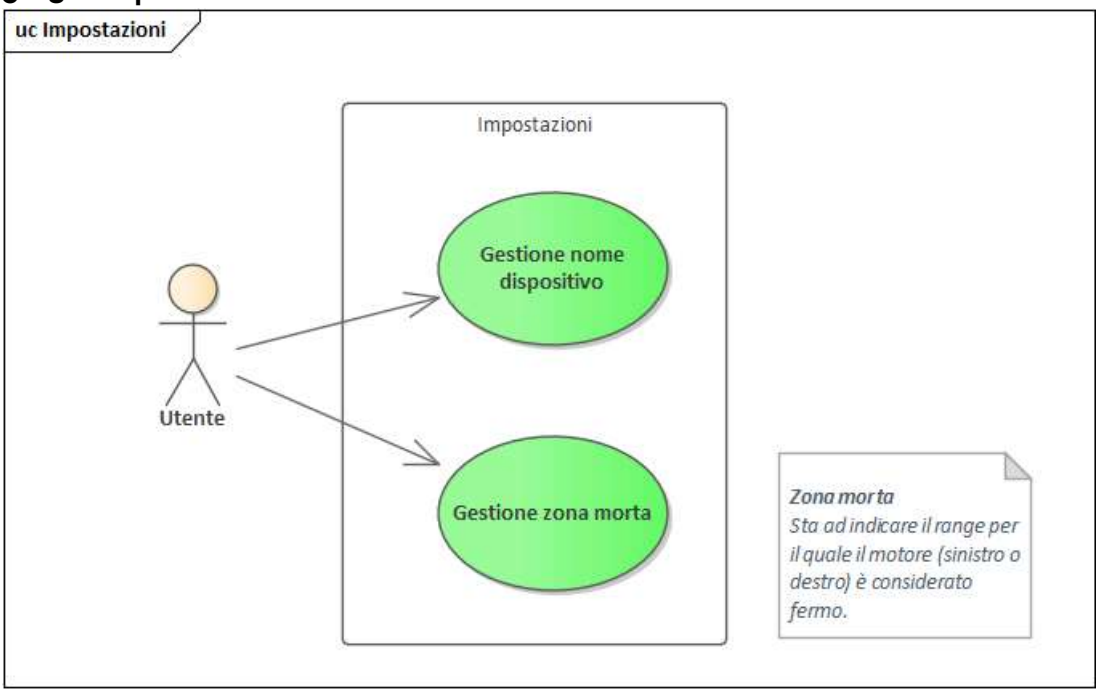
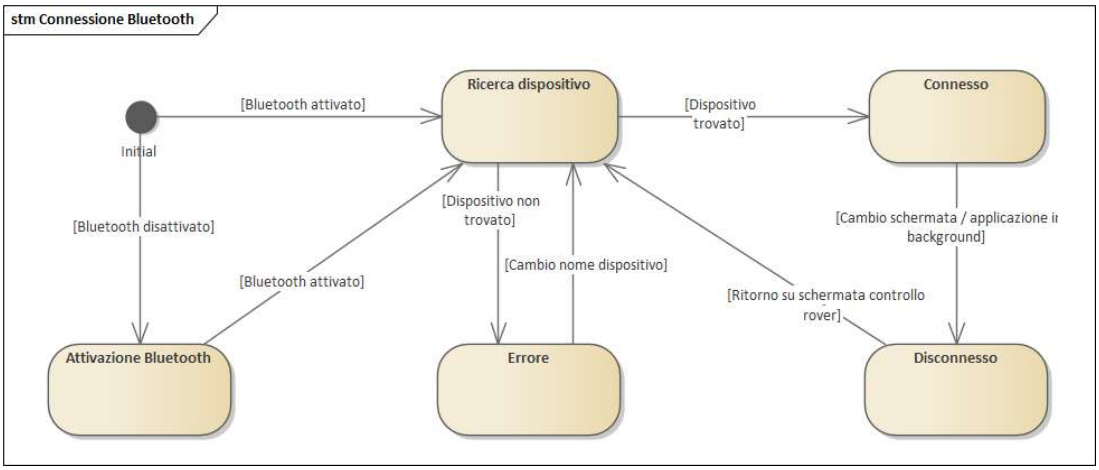


Diagramma delle impostazioni che l'utente può impostare.

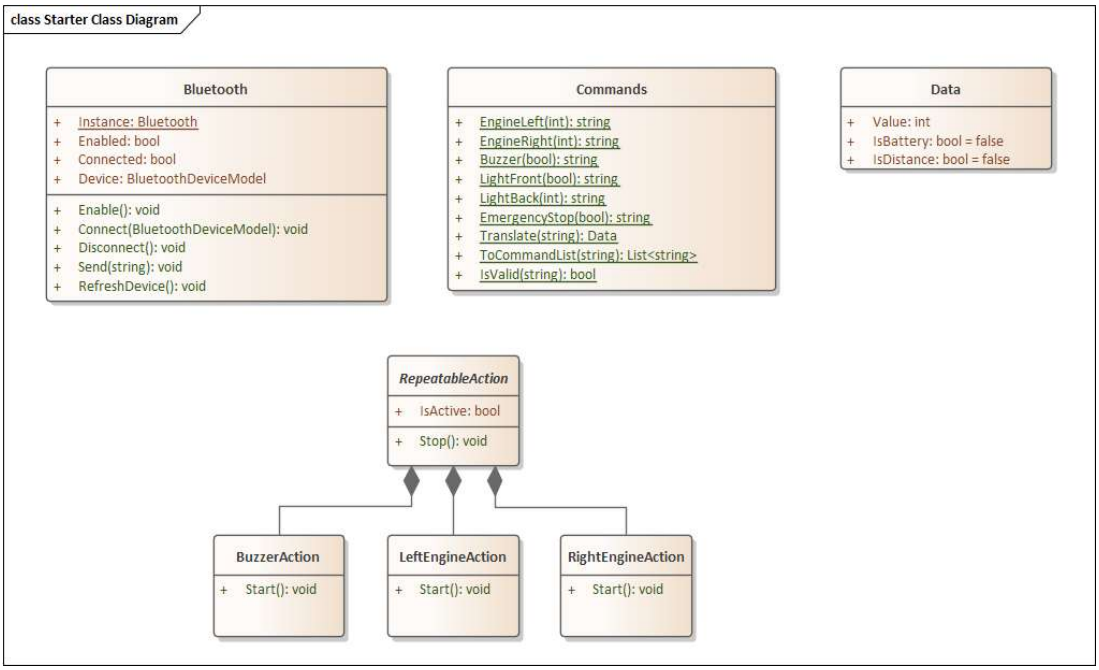
5.3 STATE DIAGRAM

5.3.1 Connessione Bluetooth



State diagram della gestione della connettività Bluetooth all'interno dell'applicazione.

5.4 CLASS DIAGRAM



Classe	Descrizione
Bluetooth	Gestione di Bluetooth sul dispositivo. • Accendere Bluetooth • Ricercare dispositivi • Collegarsi ad un dispositivo • Inviare dati al dispositivo • Ecc.
Commands	Genera le stringhe di testo conformi alle regole prestabilite. Si occupa anche di convertire stringhe di testo contenente comandi, validare se sono conformi alle regole e convertirle in classi 'Data'.
Data	Una classe che rappresenta uno dei possibili dati in entrata.
RepeatableAction	Classe astratta che definisce la logica per delle azioni ripetibili fino a che non viene cancellata l'operazione.
BuzzerAction	Estende 'RepeatableAction' e definisce l'azione da eseguire in ripetizione per attivare e disattivare il buzzer.
LeftEngineAction	Estende 'RepeatableAction' e definisce l'azione da eseguire in ripetizione per definire la velocità del motore sinistro.
RightEngineAction	Estende 'RepeatableAction' e definisce l'azione da eseguire in ripetizione per definire la velocità del motore destro.

5.5 APP IBRIDA VS NATIVA

5.5.1 Nativa

Un app nativa è compilata in uno specifico linguaggio per una specifica piattaforma. Queste applicazioni non sono veramente native, perché questo vorrebbe dire sviluppare un'applicazione diversa per ogni sistema, ma ci si avvicinano molto.

Le applicazioni native possono avere accesso a funzionalità legate all'hardware del dispositivo, come fotocamera, GPS, contatti, ecc.

5.5.2 Ibrida

Un'applicazione ibrida può girare

Un app ibrida usa HTML, CSS e JavaScript per creare una versione web dell'applicativo e poi viene visualizzato all'interno del dispositivo.

Al contrario delle applicazioni native, quelle ibride non hanno accesso ad alcune funzionalità del dispositivo. Questo non vuol dire necessariamente che non sia sempre possibile accedergli: Ionic, ad esempio, usa dei plugin di Cordova per integrare le funzionalità native richieste per il funzionamento dei componenti hardware.

5.6 ANALISI TECNOLOGICA

Sono stati scelti esclusivamente frameworks che supportano lo sviluppo cross-platform, siccome l'applicativo deve funzionare sia su dispositive Android che iOS, e la possibilità di sviluppare due sistemi distinti è stata scartata.

5.6.1 Xamarin

Sviluppatore	Microsoft
Rilascio	2011
Linguaggio	C#, F#
Tipologia	Nativo
Open source	Si
IDE	Visual Studio

5.6.2 Flutter

Sviluppatore	Google
Rilascio	2017
Linguaggio	Dart
Tipologia	Nativo
Open Source	Si
IDE	Indipendente

5.6.3 Ionic

Sviluppatore	Drifty
Rilascio	2013
Linguaggio	JavaScript con Angular, React o Vue
Tipologia	Ibrido
Open Source	Si
IDE	Indipendente

5.6.4 Cordova

Sviluppatore	Adobe
Rilascio	2019
Linguaggio	HTML5, CSS3, and JavaScript
Tipologia	Ibrido
Open Source	Si

IDE	Indipendente

5.6.5 React Native

Sviluppatore	Meta (Facebook)
Rilascio	2015
Linguaggio	JavaScript con React
Tipologia	Nativo
Open Source	Si
IDE	Indipendente

5.6.6 Confronto

	Xamarin	Flutter	Ionic	Cordova	React Native
Sviluppatore	Microsoft	Google	Drifty	Adobe	Facebook
Rilascio	2011	2017	2013	2009	2015
Linguaggio	C#, F#	Dart	JavaScript con Angular, React o Vue	HTML5, CSS3, and JavaScript	JavaScript con React
Nativo/Ibrido	Nativo	Nativo	Ibrido	Ibrido	Nativo
Open source	Si	Si	Si	Si	Si
IDE	Visual Studio	Indipendente	Indipendente	Indipendente	Indipendente

5.7 ANALISI TECNOLOGICA APPROFONDITA

Tutte le tecnologie sono delle valide possibilità per questo progetto ma vorrei evitare le tecnologie ibride siccome andiamo a lavorare con Bluetooth, quindi ad interagire in modo diretto con l'hardware del telefono.

Inoltre, ho deciso di scartare Flutter per via del nuovo linguaggio da dover imparare, il quale rallenterebbe notevolmente lo sviluppo iniziale.

5.7.1 Xamarin

5.7.1.1 Sviluppo

Gli strumenti all'interno di Visual Studio velocizzano il processo di sviluppo con controlli e layout predefiniti. La funzione di "Hot Reloading", ovvero la possibilità di vedere l'applicazione aggiornata immediatamente senza bisogno di ricompilarla, accelera molto la parte di sviluppo.

5.7.1.2 Performance

A livello di performance è quasi alla pari con le applicazioni sviluppate nativamente per iOS o Android.

5.7.1.3 Orientamento dispositivo

La possibilità di avere schermate diverse a dipendenza se il dispositivo è in orizzontale o in verticale è direttamente integrata nel framework.

5.7.1.4 Dimensioni dispositivo

Supporta vari tipi di dispositivo e si possono configurare facilmente delle viste per determinati dispositivi.

5.7.1.5 Bluetooth
Per connettersi ad un dispositivo che usa Bluetooth Classic, si può aggiungere un plugin chiamato "Plugin.BluetoothClassic[3]".

5.7.1.6 Emulazione
Visual Studio racchiude un emulatore Android tramite il quale si può vedere l'applicativo e viene aggiornato in tempo reale. L'emulatore gira senza problemi su una macchina dotata di almeno 16GB di RAM[4] e con installato Hyper-V o altri software che assistono alla virtualizzazione, come Intel HAXM.

5.7.2 React Native

5.7.2.1 Sviluppo
Come per Xamarin, anche React Native supporta la funzione di 'Hot Reloading'.
Performance

A livello di performance è quasi alla pari con le applicazioni sviluppate nativamente per iOS o Android.

5.7.2.2 Orientamento dispositivo
Non è predisposto ad avere la gestione comoda dell'interfaccia a dipendenza dell'orientamento del dispositivo. Per fare ciò è necessario avvalersi di una libreria esterna (react-native-orientation), la quale richiede una configurazione manuale scritta nel linguaggio nativo per il dispositivo (Objective C per iOS o Java per Android).

Tutto questo rende il tutto impossibile da testare senza compilare il tutto e installarlo su un dispositivo.

5.7.2.3 Dimensioni dispositivo
React Native usa un sistema senza unità per definire le dimensioni dei vari elementi, il quale si basa sulla densità di pixel del dispositivo. Questa è una soluzione che può creare dei problemi quando si passa su dispositivi con uno schermo e una densità di pixel molto alti. Ci sono però delle librerie che danno più flessibilità e aiutano a dare un risultato migliore (ad esempio "react-native-size-matters").

È anche possibile implementare due schermate diverse a dipendenza della dimensione del dispositivo.

5.7.2.4 Bluetooth
Al contrario di Xamarin non c'è una soluzione per la connessione Bluetooth direttamente integrate ma esistono delle soluzioni sviluppate da terzi.

Un esempio è la libreria "react-native-bluetooth-classic[5]" offre un'implementazione altrettanto semplice e veloce quanto quella di Xamarin.

5.7.2.5 Emulazione
React Native non necessita però di un emulatore ma l'applicativo può essere visualizzato su un qualsiasi browser moderno.

---

[3] https://github.com/rostislav-nikitin/Plugin.BluetoothClassic
[4] Riferimento personale in base alle risorse utilizzate avendo l'ambiente di sviluppo e la macchina virtuale avviata.
[5] https://www.npmjs.com/package/react-native-bluetooth-classic

5.7.3 Scelta finale: Xamarin

Xamarin è sembrata la scelta più solida fin dall'inizio, l'unica cosa anche mi frenava inizialmente era l'emulazione in fase di sviluppo che richiede molte risorse e il computer che mi era stato assegnato per lavorare era limitato sotto quel punto di vista.

Dopo aver creato un prototipo ho però notato la possibilità di eseguire il deploy direttamente su un reale telefono Android. Questo fattore non solo ha rimosso la necessità di avere un computer con hardware più performante, ma ha anche dato la possibilità di testare più comodamente le funzioni.

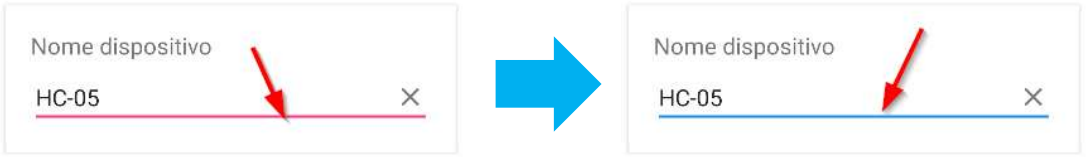
5.7.4 Considerazioni a posteriori

Xamarin si è rivelato uno strumento molto completo e facile ma poco flessibile quando ho provato ad uscire dalle classiche customizzazioni od operazioni da eseguire.

I componenti che fornisce Xamarin sono molti e hanno incluse molte funzionalità, questo rende lavorare con Xamarin molto intuitivo e rapido.

Per quanto riguarda la personalizzazione alcune volte vengono fornite delle funzioni direttamente sul componente, mentre altre volte bisogna andare a toccare molti componenti per una semplice customizzazione.

Una di queste customizzazioni è stato cambiare il colore del bordo del input di testo da rosa a blu.



Per eseguire ciò ho dovuto applicare le modifiche e creare le classi seguenti:

```
<Entry
    x:Name="DeadZoneInput"
    ClearButtonVisibility="WhileEditing"
    IsSpellCheckEnabled="False"
    IsTextPredictionEnabled="False"
    Keyboard="Numeric"
    ReturnType="Done"
    TextChanged="OnDeadZoneChange">
    <Entry.Effects>
        <effects:InputEntry />
    </Entry.Effects>
</Entry>
```

Figure 5 - Applicato un Effects, il quale permette di customizzare per ogni piattaforma i controlli

```
namespace RoverControlApp.Effects
{
    1 reference
    public class InputEntry : RoutingEffect
    {
        0 references
        public InputEntry() : base("InputEntryGroup.InputEntryEffect")
        {
        }
    }
}
```

Figure 6 - Creato l'Effects base, ovvero indipendente dalla piattaforma

```
[assembly: ResolutionGroupName("InputEntryGroup")]
[assembly: ExportEffect(typeof(AndroidInputEntryEffect), "InputEntryEffect")]
namespace RoverControlApp.Droid.Renderer
{
    2 references
    public class AndroidInputEntryEffect : PlatformEffect
    {
        0 references
        public AndroidInputEntryEffect()
        {
        }

        0 references
        protected override void OnAttached()
        {
            if (Control != null)
            {
                Android.Graphics.Color entryLineColor = Android.Graphics.Color.Rgb(33, 150, 243); // Primary color
                Control.BackgroundTintList = ColorStateList.ValueOf(entryLineColor);
            }
        }
```

Figure 7 - Creato l'Effect legato alla piattaforma, in questo caso Android

Nel complesso devo dire che sono soddisfatto della scelta di Xamarin nonostante quanto appena menzionato siccome il tempo speso per customizzare questi dettagli, è stato ampiamente compensato da molte altre funzionalità che hanno accelerato lo sviluppo.

Nel caso si volesse creare un applicativo con un design molto specifico sconsiglierei l'utilizzo di Xamarin.

5.7.5 Fonti

Qui si possono trovare le fonti utilizzate per l'analisi delle tecnologie.

Ravichandran, Adhithi. 2019. "React Native vs. Ionic: Which One Is Right for You?" LogRocket Blog. July 26, 2019. https://blog.logrocket.com/react-native-vs-ionic/.

"Xamarin vs Ionic - Which Is Better?" 2021. August 22, 2021. https://blog.back4app.com/xamarin-vs-ionic/.

Gray, Lerma. 2022. "Xamarin vs Flutter: 3 Critical Lessons Learnt after Using Xamarin." DevCount.com. May 1, 2022. https://devcount.com/xamarin-vs-flutter/.

Solanki, Jignesh. 2021. "Xamarin vs Ionic for Mobile Application Development in 2022." Simform - Product Engineering Company. January 22, 2021. https://www.simform.com/blog/xamarin-vs-ionic/.

Ravichandran, Adhithi. 2019. "React Native vs. Ionic: Which One Is Right for You?" LogRocket Blog. July 26, 2019. https://blog.logrocket.com/react-native-vs-ionic/.

"Cordova vs. React Native: A Detailed Analysis | Waldo Blog." n.d. Www.waldo.com. Accessed November 4, 2022. https://www.waldo.com/blog/cordova-vs-react-native.

"Xamarin vs React Native - Make a Choice in 5 Steps [2022 Guide]." n.d. Brainhub.eu. Accessed November 4, 2022. https://brainhub.eu/library/react-native-vs-xamarin.

Alferd, Sten. 2019. "Flutter vs Xamarin vs React Native — Let the Battle Begin!" Medium. December 3, 2019. https://medium.com/@stenalferd/flutter-vs-xamarin-vs-react-native-let-the-battle-begin-d3e783bb4bf1.

Vallecillo, Yamill. 2022. "Yamill/React-Native-Orientation." GitHub. October 26, 2022. https://github.com/yamill/react-native-orientation.

Diaz, Jorge Perales. 2021. "How to Integrate Bluetooth LE in Xamarin Forms." EnigmaMx. April 19, 2021. https://medium.com/enigmamx/how-to-integrate-bluetooth-le-in-xamarin-forms-2dcdf974703a.

"Different Mobile and Desktop Layouts with React." 2016. Gosha Arinich. December 20, 2016. https://goshacmd.com/different-mobile-desktop-tablet-layouts-react/.

Kadam, Akshay. 2018. "Scaling React Native Apps for Tablets." React Native Training. May 1, 2018. https://medium.com/react-native-training/scaling-react-native-apps-for-tablets-211de8399cf1.

6 MOCKUP

6.1 INIZIALE

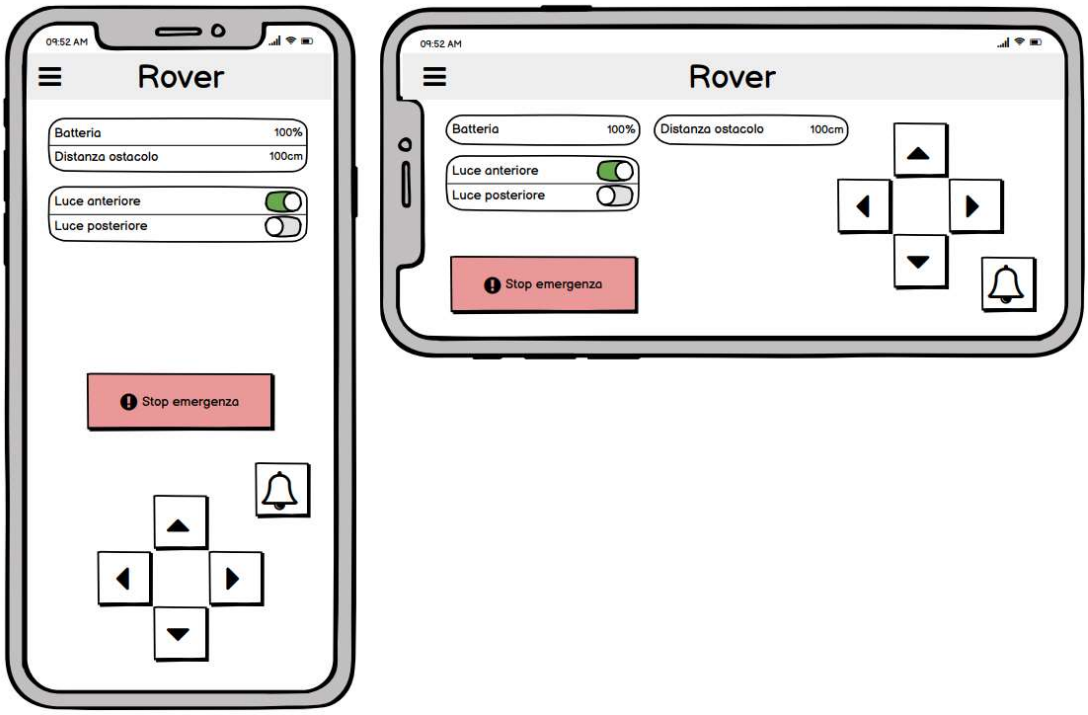


Figure 8 - Mockup iniziale (telefono)

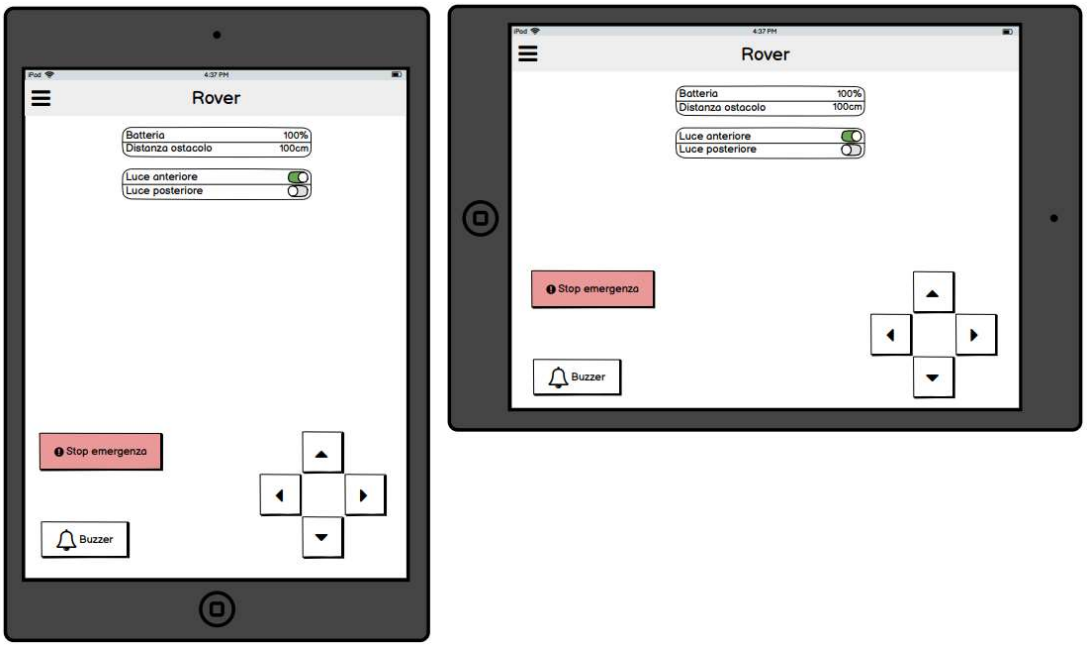


Figure 9 - Mockup iniziale (tablet)

6.2 FINALE

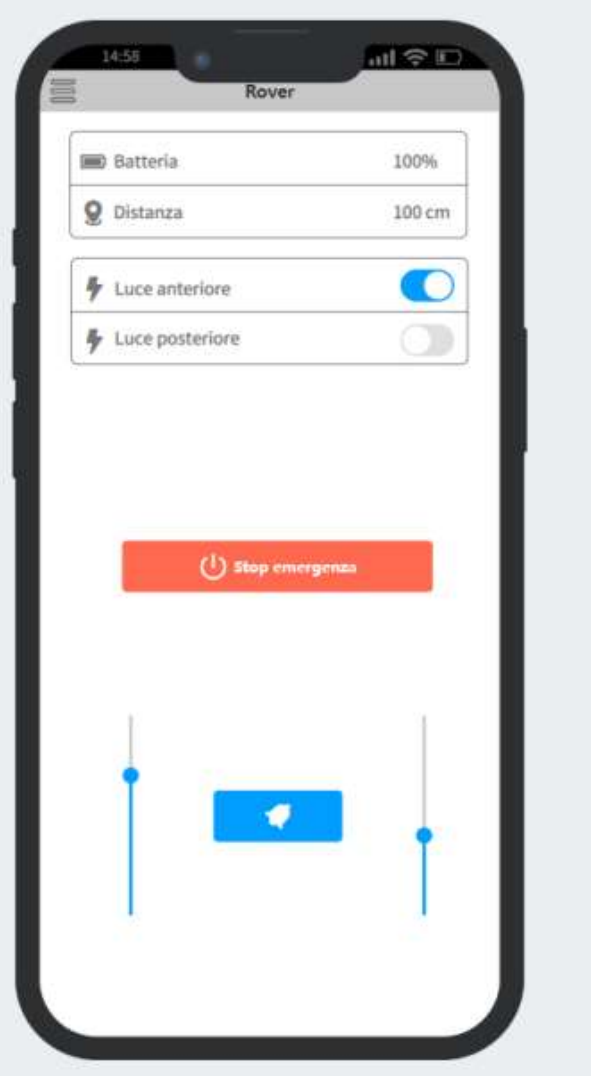


Figure 10 - Mockup finale (telefono)

Il cambiamento maggiore è stata l'eliminazione della versione tablet, la quale è stata rimandata.

Per quanto riguarda i mockup del telefono sono rimasti quasi invariati, l'unico cambiamento è stato relativo al controllo direzionale del rover.

Al seguito di un confronto con il cliente, il quale ha evidenziato che i motori sinistro e destro sono distinti, e che possono essere controllati solamente in avanti e indietro con un range di velocità, si è allora deciso di cambiare la croce direzionale in due slider.

6.3 PRODOTTO FINALE

Figure 11 - Prodotto finale (telefono)

Il prodotto realizzato rispecchia molto fedelmente il mockup finale, sono stati effettuati alcuni accorgimenti, quali: una migliore gestione degli spazi e delle dimensioni di bottoni e testi; ed è stata aggiunta una variazione di colore agli slider a dipendenza del tipo di input che mandano: blu – fermo, rosa – indietro e verde – avanti.

Questa aggiunta del colore degli slider migliora l'usabilità. Un altro miglioramento all'usabilità è stato aumentare la lunghezza degli slider in modo da consentire un controllo più preciso.

6.3.1 Mockup Programs

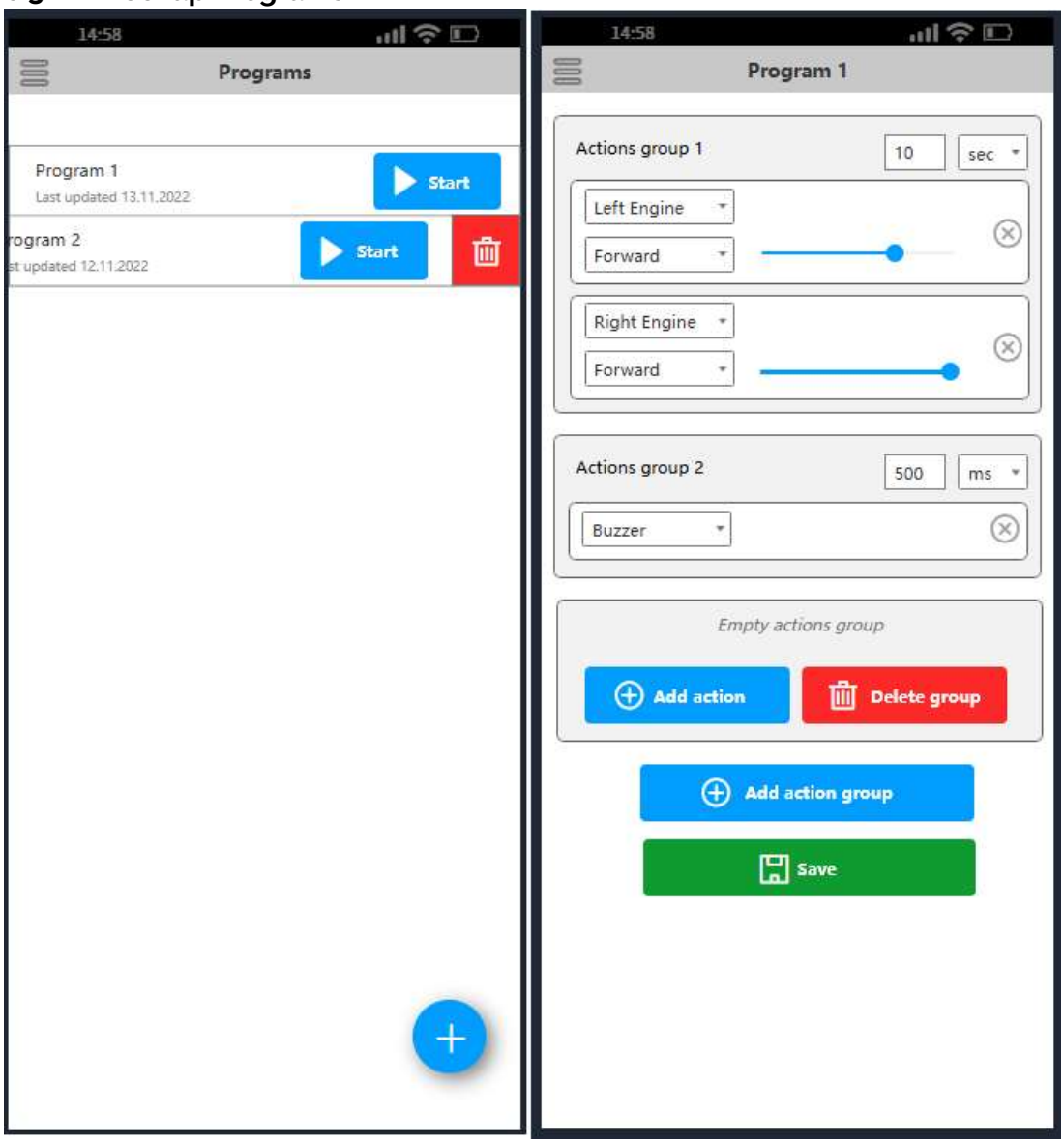


Figure 12 - Selezione dei programmi (sinistra), aggiunta o rimozione dei comandi da eseguire (destra)

A sinistra c'è la pagina iniziale, la quale darà la possibilità di eseguire un programma esistente, oppure, selezionarne uno per modificarlo o eliminarlo. Oppure anche crearne uno nuovo.

Nel caso si selezionasse un programma o se ne vorrà creare uno nuovo, si aprirà la schermata a destra, la quale permetterà di creare delle "Actions Group" e di inserire in esse dei comandi.

Le "Actions Group" sono dei gruppi di comandi, i quali verranno eseguiti insieme e a ripetizione per una durata di tempo. Una volta terminata l'esecuzione di un gruppo, si passerà a quello seguente.

7 Motivazioni didattiche

L'idea è nata per motivare gli studenti del primo anno di elettrotecnica a costruire il rover il quale potrà poi essere controllato con un'applicazione installata sul proprio smartphone.

8 Ricerche di mercato

È stata fatta provare l'applicazione a degli studenti del terzo anno e gli è stato chiesto di dare dei feedback.

Di seguito ho elencato i feedback ricevuti e sotto di essi ho aggiunto dei commenti.

8.1 Pro

Apk facile da installare
Non direttamente collegato allo sviluppo siccome è una funzionalità standard di Android, ma comunque un'ottima cosa che non ci siano stati problemi durante l'installazione.

Modulo Bluetooth molto buono
Parzialmente legato alla parte hardware e non all'applicativo, ma è stato percepito positivamente anche la gestione Bluetooth da parte dell'app (si è collegato senza problemi e ha funzionato correttamente).

Interfaccia user friendly
È stato compreso il funzionamento di tutto quasi subito.

8.2 Cons

È lento
Questo era riferito alla velocità del rover e l'app non può far andare il rover più veloce di quanto è stato predefinito.

Noioso che bisogna tenere premuto il clacson (toggle sul pulsante)
Questo è stato un requisito dal committente e anche dopo il feedback è rimasto comunque dell'idea che è corretto il comportamento attuale.

Bug della modalità dark da risolvere
Nel caso non si riuscisse a risolvere è stato suggerito di mostrare un popup all'avvio che avvisa che l'app non è interamente compatibile con la dark mode.

Il sensore di distanza non serve a molto. (Si potrebbe impostare una distanza alla quale la macchinina si ferma)
La possibilità di fermare il rover quando il sensore rileva un oggetto molto vicina si era discussa con il committente ma lasciata come un "nice to have" per sviluppi futuri.

Se si lascia andare il motore al massimo tenderà ad andare verso destra
Il problema è legato ai motori del rover e non all'applicazione.

8.3 Migliorie

Scrivere motore destra e motore sinistra sopra alla barra dei due motori
Con il committente si è deciso che questa miglioria non è necessaria. Una constatazione personale è che da questo feedback si denota che le funzionalità relative ai motori non sono subito intuitive.

8.4 Funzioni supplementari

Modalità volante
Sicuramente interessante come funzione, richiederebbe però un notevole tempo di sviluppo.

Possibilità di modificare la canzone del clacson
Funzione non presente nell'app siccome la melodia riprodotta è selezionata nel software installato sul rover e non sull'applicazione mobile.

Modalità randomica
Si intende una modalità in cui il rover si muove autonomamente e quando si avvicina troppo ad un ostacolo cambierebbe direzione. Questa funzionalità è interessante ma molto simile alla funzione "Programs", la quale non era ancora presente quando gli studenti hanno provato l'app.

9 QUALITÀ

9.1 NORME AZIENDALI

Non sono presenti degli standard aziendali per quanto riguarda la parte di sviluppo.

9.2 REGOLAMENTO DOCUMENTAZIONE

Per quanto riguarda la documentazione ho cercato di creare una struttura di capitoli ordinata ed intuitiva. Ho mantenuto la stessa struttura e formattazione all'interno dei vari documenti creati, cambiando il colore principale, il quale è visibile in copertina, nei titoli di livello più alto, e come colore utilizzato per le tabelle.

Ho utilizzato tabelle o immagini di supporto dove aiutavano a capire dei concetti, o aiutavano a semplificare delle parti.

9.3 RILASCIO SOFTWARE

Si era inizialmente pensato di rilasciare l'applicativo sullo store di Android (Google Play Store) ma si è infine deciso di non pubblicarlo ma di esportarlo sul computer e poi distribuirlo manualmente.

9.4 STANDARD

9.4.1 Microsoft

Standard che raccomanda Microsoft per lo sviluppo con C#.

9.4.1.1 Naming conventions

L'utilizzo del "Pascal case"[6] è raccomandato per i nomi di classi, metodi e field pubblici.

Variabili locali, field privati o argomenti ricevuti nei metodi utilizzano il "Camel case"[7].

Evitare lo "Screaming case"[8] o delle sue varianti, siccome attirano troppo l'attenzione.

9.4.1.2 Tipi variabili

Utilizzare i tipi predefiniti e non specifici per il sistema, ovvero utilizzare "int" invece di "Int16", "Int32" o "Int64".

Utilizzare il tipo "var" per la dichiarazione di variabili dove il tipo è intuibile. Esempio invece di "string deviceName = …" utilizzare "var deviceName = …".

9.4.1.3 Nomi file

I file devono avere lo stesso nome della loro classe principale. Quindi se un file ha all'interno la classe "public class Bluetooth {}" il file si chiamerà "Bluetooth.cs".

---

[6] Anche conosciuto come "Upper camel case", usa il formato seguente: "StartEventProcessing".
[7] Come "Pascal case" ma con l'iniziale minuscola.
[8] Utilizzare tutte le lettere in maiuscole, come ad esempio "SCREAMING_SNAKE_CASE".

9.4.2 Personali

Standard che mi sono autoimposto per rendere il codice più leggibile e organizzato.

9.4.2.1 Nomenclature

Si è evitata l'abbreviazione di qualsiasi nome o nomenclatura, le quali possono creare confusione nel caso il progetto venga preso in mano in futuro. Si sono dati nomi chiari che facciano capire lo scopo, anche a discapito di diventare lunghi.

Esempi:

- EmergencyStopOffLabel
- EmergencyStopOnLabel
- EmergencyStopOffColor
- EmergencyStopOnColor

Siccome ci sono 4 variabili le quali sono molto simili come nome e funzionalità è stato importante dare dei nomi chiari.

9.4.2.2 File di utility

Ho preferito esportare alcune funzioni in file di utility così da ridurre le dimensioni di alcune classi e renderle più leggibili.

I file di utility sono suddivisi per categoria o tema, e non devono essere istanziati, contengono solamente variabili e metodi statici.

Un esempio è la funzione per mostrare degli alert popup, la quale è stata esportata in un file di utility dedicato agli alert.

```
1 reference
public static void DisplayAlert(string title, string description, string buttonText = "OK")
{
    Device.InvokeOnMainThreadAsync(() => Application.Current.MainPage.DisplayAlert(
            title,
            description,
            buttonText));
}
```

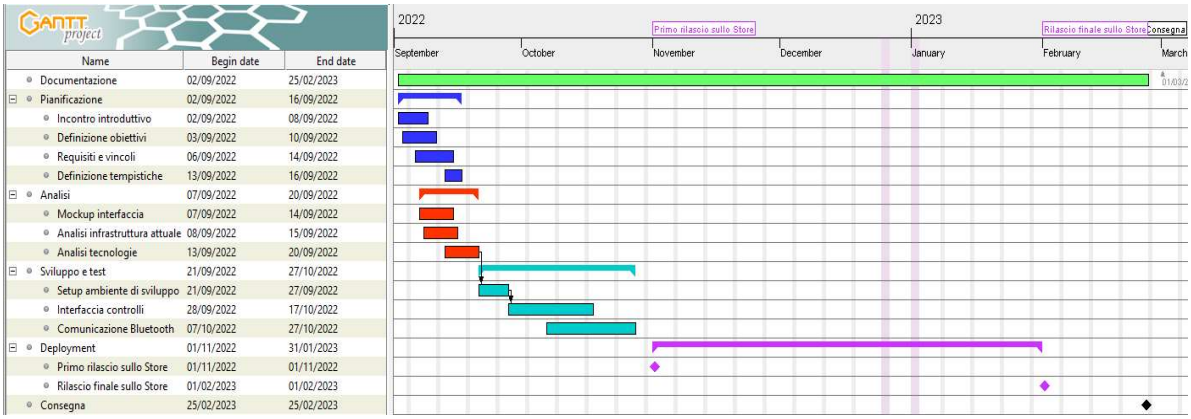
Figure 13 - Metodo per mostrare il popup

Figure 14 - Esempio di alert popup

10 PIANIFICAZIONE

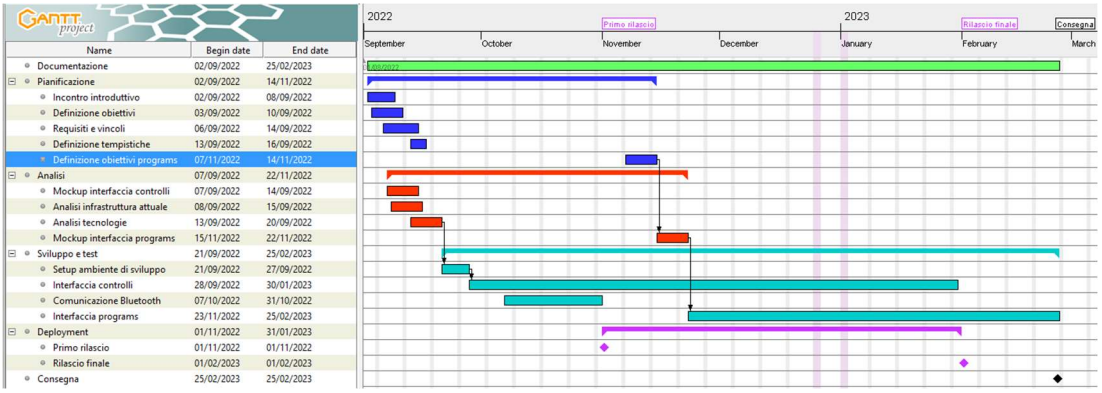
10.1 GANTT

10.1.1 Iniziale



Name	Begin date	End date
Documentazione	02/09/2022	25/02/2023
Pianificazione	02/09/2022	16/09/2022
Incontro introduttivo	02/09/2022	08/09/2022
Definizione obiettivi	03/09/2022	10/09/2022
Requisiti e vincoli	06/09/2022	14/09/2022
Definizione tempistiche	13/09/2022	16/09/2022
Analisi	07/09/2022	20/09/2022
Mockup interfaccia	07/09/2022	14/09/2022
Analisi infrastruttura attuale	08/09/2022	15/09/2022
Analisi tecnologie	13/09/2022	20/09/2022
Sviluppo e test	21/09/2022	27/10/2022
Setup ambiente di sviluppo	21/09/2022	27/09/2022
Interfaccia controlli	28/09/2022	17/10/2022
Comunicazione Bluetooth	07/10/2022	27/10/2022
Deployment	01/11/2022	31/01/2023
Primo rilascio sullo Store	01/11/2022	01/11/2022
Rilascio finale sullo Store	01/02/2023	01/02/2023
Consegna	25/02/2023	25/02/2023

10.1.2 Finale



Name	Begin date	End date
Documentazione	02/09/2022	25/02/2023
Pianificazione	02/09/2022	14/11/2022
Incontro introduttivo	02/09/2022	08/09/2022
Definizione obiettivi	03/09/2022	10/09/2022
Requisiti e vincoli	06/09/2022	14/09/2022
Definizione tempistiche	13/09/2022	16/09/2022
Definizione obiettivi programs	07/11/2022	14/11/2022
Analisi	07/09/2022	22/11/2022
Mockup interfaccia controlli	07/09/2022	14/09/2022
Analisi infrastruttura attuale	08/09/2022	15/09/2022
Analisi tecnologie	13/09/2022	20/09/2022
Mockup interfaccia programs	15/11/2022	22/11/2022
Sviluppo e test	21/09/2022	25/02/2023
Setup ambiente di sviluppo	21/09/2022	27/09/2022
Interfaccia controlli	28/09/2022	30/01/2023
Comunicazione Bluetooth	07/10/2022	31/10/2022
Interfaccia programs	23/11/2022	25/02/2023
Deployment	01/11/2022	31/01/2023
Primo rilascio	01/11/2022	01/11/2022
Rilascio finale	01/02/2023	01/02/2023
Consegna	25/02/2023	25/02/2023

Nel Gantt finale è stata aggiunta la parte relativa l'analisi e sviluppo della sezione Programs.

10.2 TRELLO

Ho seguito un approccio Agile, definendo degli sprint di 4 settimane e definendo delle storie (o task) i quali ho raggruppato sono le seguenti macro-categorie:

- Analisi
- Interfaccia
- Funzionalità
- Documentazione

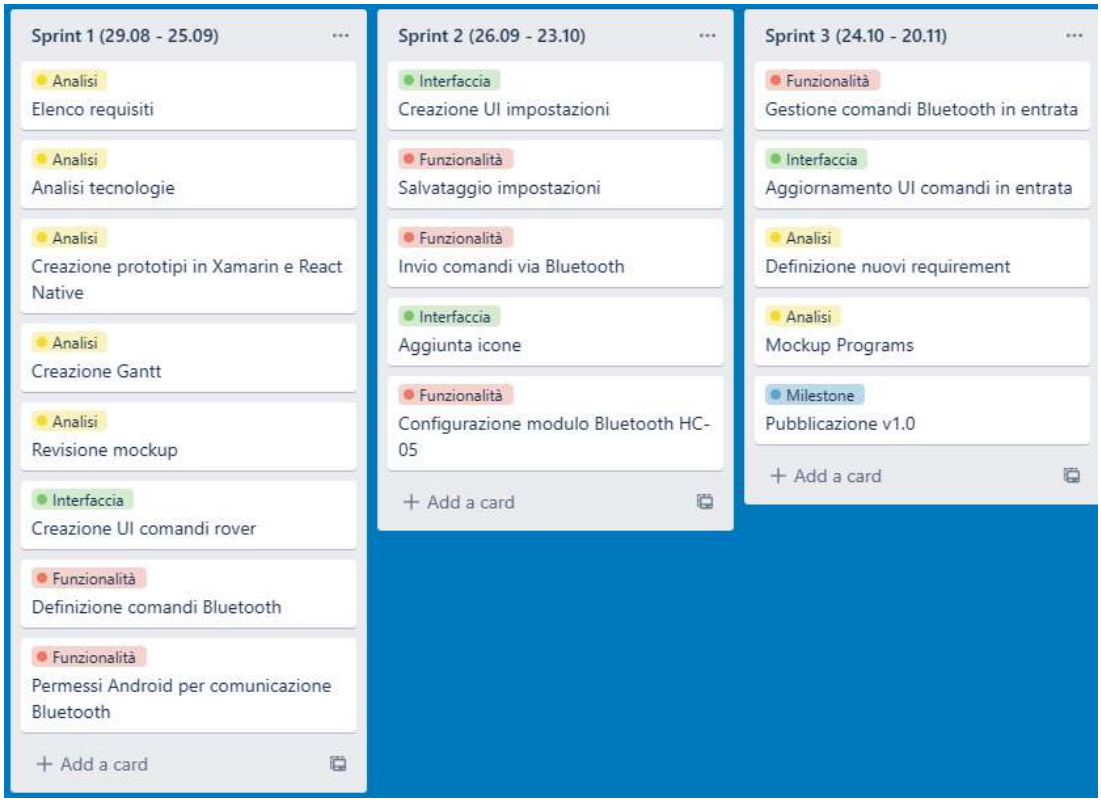


Figure 15 – Trello (Sprint 1 – 3)

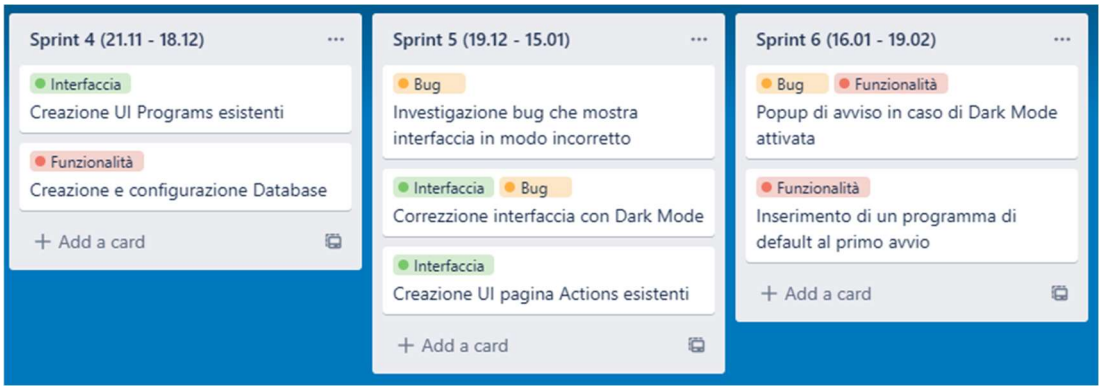


Figure 16 - Trello (Sprint 4-6)

11 CICLO DI VITA DEL SOFTWARE

Controlli rover	
Raccolta dei requisiti e analisi	
Creazione mockup e use case	
Sviluppo	
Testing	
Deployment	
Controlli rover	**Programs**
Maintenance	Raccolta dei requisiti e analisi
	Creazione mockup e use case
	Sviluppo

Una volta completata la parte relativa ai controlli del rover (Controls) questa è passata nella fase di maintenance e la parte relativa ai Programs è iniziata. Questa però è ancora in fase di sviluppo.

12 METODOLOGIA

12.1 FONTE DI ISPIRAZIONE

Per la metodologia ho utilizzato Agile e mi sono ispirato a Scrum, siccome l'ho già utilizzata diverse volte in ambito lavorativo e siccome la suddivisione del lavoro in tanti piccoli task e pianificare il lavoro per un periodo relativamente corto è una cosa che mi aiuta.

12.2 ADATTAMENTO DELLA METODOLOGIA TEORICA

Lavorando da solo invece che in un team ho dovuto tagliare alcuni elementi classici di Scrum, come ad esempio la daily standup, la sprint review, ecc.

Ho invece definito lo sprint da 4 settimane, e inserivo le storie che pianificavo di fare in quel periodo, che fosse il periodo corrente o uno futuro.

Alla fine dello sprint, nel caso una storia fosse stata completata solamente a metà, la dividevo in 2 storie più piccole, così da segnare cosa era stato realmente fatto in quello sprint.

Non avevo una gestione dello stato della storia (ToDo, In Progress, Done) ma consideravo l'insieme delle storie come completato una volta che passavo allo sprint seguente, se non fosse così verrebbe spostata in quello successivo o suddivisa.

Il backlog è stato rimosso e al suo posto inserivo le storie in uno sprint futuro. Siccome io avevo sia il ruolo di developer che di Scrum master, quando creavo delle storie, veniva subito pianificato quando era necessario finirla.

12.3 ADATTAMENTO NEL CORSO DEL PROGETTO

Avevo inizialmente definito gli sprint da 2 settimane, in modo da avere una sorta di Sprint review ogni volta che vedevo il cliente e lo aggiornavo sui progressi fatti.

Questo è però risultato troppo limitante, alcune storie, nonostante piccole, prendevano più di 2 settimane. Invece di ridurre ulteriormente le storie per adattarle agli sprint, ho deciso di passare a sprint più lunghi.

Questo ha ridotto notevolmente il tempo speso di lavoro impiegato all'aggiustamento delle storie e pianificazione degli sprint futuri.

12.4 MOTIVAZIONE DELLA SCELTA

Ho scelto la metodologia Agile perché la suddivisione in piccoli task funzionanti, i quali sono testabili e sui quali potevo ricevere dei feedback dal cliente, era un grande bonus. Essendo anche un progetto molto particolare, del quale non esistono dei modelli o delle interfacce già collaudate, era quindi molto facile sbagliare o fraintendere i requisiti del cliente.

Nel caso avessi usato il processo a cascata (waterfall) probabilmente il progetto non sarebbe andato a buon fine a causa che le prime verifiche del funzionamento si riescono ad effettuare solo verso la fine del processo. Questo avrebbe fatto fuoriuscire delle incongruenze nate in fase di sviluppo solamente verso le prime fasi di test o nella messa in produzione.

13 Analisi rischi

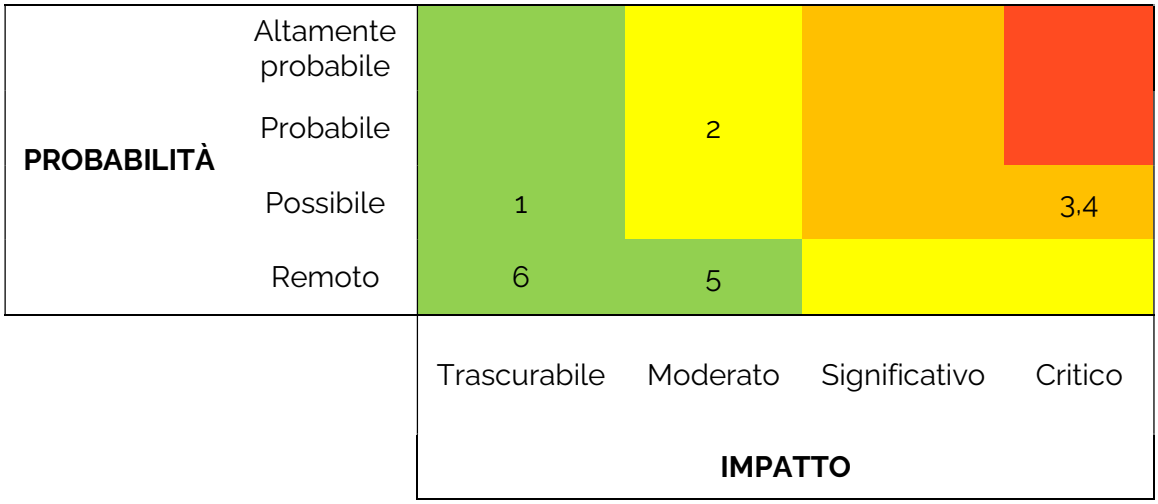
L'applicativo non è un processo indispensabile per il lavoro o il funzionamento dell'attività di costruzione del rover da parte degli studenti, quindi anche se il progetto non dovesse andare a buon fine, non porterebbe a grandi impedimenti.

13.1 Rischi

Lista dei rischi:

1. Risorse del pc troppo ridotte
2. Smarrimento o rottura di componenti del rover
3. Comunicazione tra telefono e rover non funziona
4. Modulo Bluetooth incompatibile con alcuni modelli di telefoni/sistemi operativi
5. Traffico di dati scambiati troppo elevati
6. Impossibilità di terminare l'applicazione nei tempi prestabiliti

13.2 Strumenti



13.3 What-if

#	What if	Manifestazione	Conseguenze	Contromisure
1	Risorse del pc troppo ridotte	L'emulazione richiede troppe risorse.	Impossibilità testare ciò è stato sviluppato.	Ricercare altri metodi per testare l'applicativo. Testare mentre si lavora da casa.
2	Smarrimento o rottura di componenti del rover	Durante il trasporto a casa di uno o più componenti, questi vengono danneggiati o persi.	Impossibilità di lavorare per un periodo di tempo. Nuovi costi per comprare il componente.	Acquistare in anticipo dei pezzi di ricambio. Portare via solo il minimo indispensabile.

3	Comunicazione tra telefono e rover non funziona	La libreria non comunica correttamente con il modulo Bluetooth del rover.	Necessità di cambiare la libreria utilizzata e dover riscrivere parte del codice.	Appena aggiunta la libreria provare ad inviare dei comandi al rover per accertarsi il corretto funzionamento.
4	Modulo Bluetooth incompatibile con alcuni modelli di telefoni/sistemi operativi	Il modulo Bluetooth HC-05 utilizzato è considerato legacy e questo potrebbe creare dei problemi con telefoni più recenti.	Impossibilità di scambiare comandi tra telefono e rover.	Avere un modulo Bluetooth più recente come alternativa. Ridurre il target di dispositivi supportati.

13.4 Rischi accaduti

Rischio (4) è accaduto in modo parziale. Il dispositivo necessita della certificazione MFi per comunicare con i dispositivi Apple e il modulo HC-05 non dispone di quest'ultima, per questo motivo iOS è stato scartato dai sistemi operativi da supportare.

14 Volumi / Quantità / Flussi

14.1 Scalabilità

Si è provato a controllare il rover quando nella stanza ne erano presenti più di uno per verificare che venissero mandati comandi solo a quello corretto, ed ha funzionato correttamente.

Questo funziona anche se ci sono più rover con il modulo Bluetooth con lo stesso nome, i comandi verranno inviati solo a quello selezionato. Il problema risiede nel collegarsi siccome non si ha nessun parametro per determinare a quale rover ci si sia collegando.

Il problema giunge solamente se il segnale ad un rover viene interrotto, in quel caso si proverà nuovamente a collegare ad uno dei due e potrebbe non più collegarsi al rover precedente ma all'altro.

14.2 Volumi futuri

I volumi non andranno ad aumentare se non per il database per i programmi che sono stati creati, i quali sono comunque molto poco impattanti a livello di volume.

14.3 Stress test

Si è provato a controllare un rover con uno smartphone, il quale era collegato con altri dispositivi Bluetooth.

I dispositivi con cui era collegato erano i seguenti:

- Rover
- Cuffiette (mentre si ascoltava musica)
- Orologio (SmartWatch)
- Luci di casa[9]

Sia il rover che gli altri dispositivi collegati funzionavano correttamente e non sono state notate differenze nell'utilizzo rispetto a quando erano collegati singolarmente.

14.4 PERFORMANCE TEST

14.4.1 Distanza

Secondo le specifiche di Bluetooth del modulo installato, la distanza massima equivale a 10 metri, la quale è fortemente dipendente dalla presenza di ostacoli (persone, muri, ecc.) che impediscono una comunicazione ideale.

Nei nostri test siamo riusciti anche a superare i 18 metri di comunicazione, in un ambiente chiuso e senza la presenza di ostacoli tra il telefono e il rover.

14.4.2 Latenza nell'invio

La latenza dell'invio dei comandi può arrivare fino a 100 millisecondi (0.1 secondi) per come è gestito l'invio di essi. Anche quando il rover si trovava a 18 metri di distanza dal telefono non si sono notati aumenti nella latenza.

14.4.3 Modulo Bluetooth

Il modulo Bluetooth monta Bluetooth 2.0 il quale supporta una comunicazione fino a 3 Mbps.

I comandi inviati sono di 9 bit (8 bit di dati e 1 di stop) e vengono inviati dall'applicazione 10 al secondo (1 ogni 100ms), ovvero il traffico generato equivale 90 bps. I comandi che devono essere inviati a ripetizione sono 3: motore sinistro, motore destro, e se attivo il buzzer. Questo vuol dire che il traffico massimo sarà di 270 bps, ben lontano limite del modulo Bluetooth.

14.4.4 Bitrate o Baud

La comunicazione tra il microcontrollore sul rover e il modulo Bluetooth avviene con un bitrate di 115'200 bps, quindi anch'esso copre senza problemi il volume di dati trasmesso.

[9] È possibile accenderle, spegnerle e regolare la luminosità

15 REPORTISTICA

Non è presente alcuna reportistica in quanto non richiesta.

16 ANALISI COSTI E BENEFICI

Tutti i costi sono espressi in CHF.

16.1 COSTI TELECOMANDO MECCANICO

I costi sono per un telecomando per uno studente. Ogni studente del primo anno costruirà un telecomando, questo significa che i costi sono proporzionati al numero di studenti nel corso degli anni.

Descrizione	Ore	Costo orario	Totale
Manodopera	4	25.00	100.00
		Totale	**100.00**

Descrizione	Quantità	Costo unitario	Totale
Materiali[10]	1	46.09	46.09
		Totale	**46.09**

TOTALE	**146.09**

Si stima che ogni anno ci sono 54 studenti al primo anno, il quale significa un costo di CHF 7,888.96 all'anno.

16.2 COSTI APP

16.2.1 Premessa

I costi per l'applicazione sono stati impostati come se fossi diplomato.

16.2.2 Costi

L'applicazione non ha costi negli anni e il suo costo non varia in base al numero di studenti.

Descrizione	Ore	Costo orario	Totale
Studio di fattibilità	20	-	100.00
Sviluppo	200	80.00	16'000.00
Test di qualità	50	80.00	4'000.00
		Totale	**20'100.00**

Descrizione	Quantità	Costo unitario	Totale
Costi acquisto telefono Android	1	200.00	200.00
Costi licenze	1	0	0

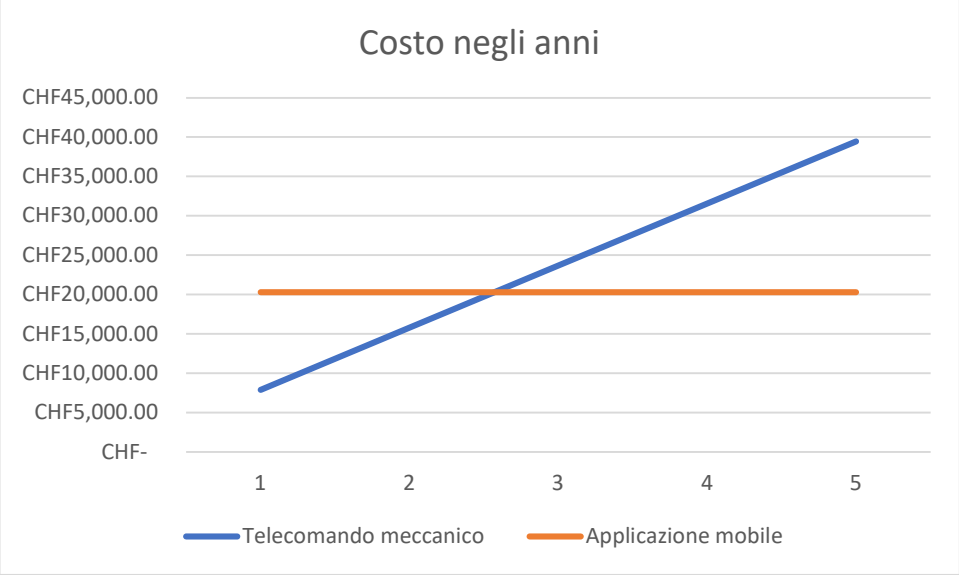
[10] Dettaglio negli allegati nel documento "CostiBenefici.xlsx"

Totale	**200.00**

TOTALE	**20'300.00**

16.3 COSTI NEGLI ANNI

L'applicazione ha un notevole costo iniziale, il quale però viene compensato, se utilizzato invece di costruire un radiocomando, per almeno 3 anni.



17 GESTIONE SICUREZZA

Al primo avvio dell'applicativo, all'utente apparirà un popup tramite il quale potrà consentire o rifiutare l'accesso all'applicazione a Bluetooth, questo processo è interamente gestito dal sistema operativo.

Il modulo Bluetooth sul rover sarà configurato con un nome e un PIN.

Nell'applicativo ci sarà una pagina dedicata tramite la quale l'utente potrà inserire il nome con il quale il dispositivo Bluetooth è configurato, così da non collegarsi accidentalmente ad un altro dispositivo.

Figure 17 - Schermata per configurare il nome del dispositivo

Nel caso il dispositivo con il nome definito non venisse trovato verrà visualizzato un popup che avviserà l'utente.

18 Migrazione dati

Per questo progetto non è stata necessaria nessuna migrazione di dati.

19 Gestione della comunicazione

19.1 Eventi ricorrenti

Ci sono stati incontri regolari con il cliente, i quali avvenivano il venerdì pomeriggio ogni due settimane.

Questi incontri servivano ad aggiornare il cliente sullo stato di avanzamento, ad avere uno scambio di pareri ed a chiarire alcuni dubbi sorti nei giorni precedenti.

19.2 Verbali previsti e tempistiche

Non sono stati creati dei verbali siccome gli incontri erano di persona. Una volta definiti i prossimi passi o cosa era necessario correggere, veniva inserito in Trello. Questo fungeva anche da ulteriore conferma per il mandante, il quale poteva controllare cosa fosse presente e a che punto fosse.

19.3 Incontri formali

Una volta al mese sono stati pianificati degli incontri formali nei quali sono stati scambiati dei feedback con il cliente.

Questi incontri comprendevano delle formalità da completare, quali firmare il foglio delle ore dedicate al progetto, ma anche una pianificazione dei prossimi passi per il mese successivo.

Ci si accordava a voce e poi venivano inseriti i task da svolgere nello sprint in Trello. Il mandante aveva accesso e poteva, se necessario, aggiungere degli elementi, modificarne alcuni esistenti o lasciare dei commenti.

19.4 Distanza/presenza

Il progetto è stato portato avanti per la maggior parte a distanza, la parte effettuata in presenza equivale circa ad un 10% del tempo complessivo.

19.5 Gestione materiali

I materiali erano divisi in due categorie: quelli che rimanevano dal cliente, mentre quelli che erano indispensabili per la parte di sviluppo, i quali erano sempre con me.

Il rover e il telecomando meccanico rimanevano sempre dal cliente, se ci fossero stati delle prove da eseguire sul rover sarebbero state eseguite durante il giorno in presenza.

Mentre, il modulo Bluetooth del rover, collegato ad un modulo il quale permetteva di essere collegato via USB al computer, e uno smartphone venivano portati avanti e indietro durante le sessioni a distanza e in presenza.

20 SVILUPPI FUTURI

20.1 FUNZIONALITÀ MANCANTI

La funzionalità Programs non è completata, manca l'opzione di creare nuovi programmi, modificare quelli esistenti e di eseguirli.

20.2 LISTA BUG ESISTENTI

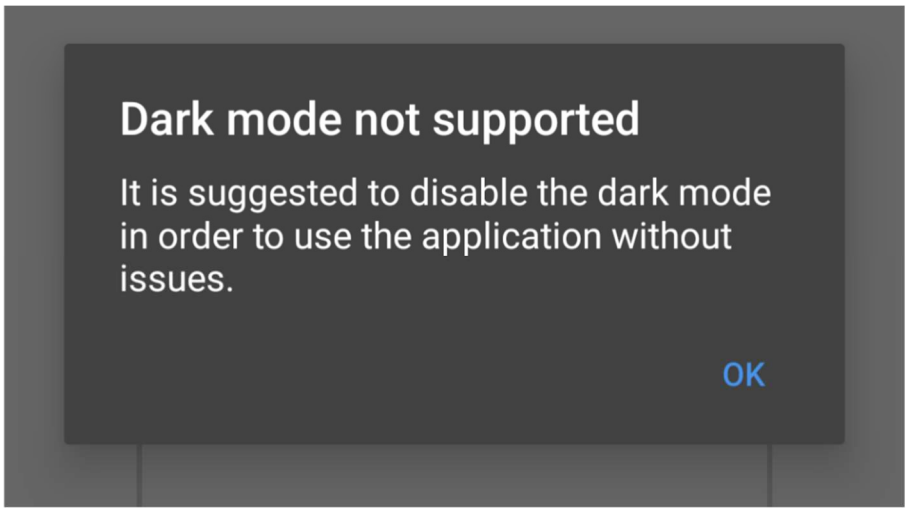
20.2.1 Switch in Dark Mode

Il componente Switch non viene visualizzato correttamente se il telefono ha la Dark Mode attivata.

Figure 18 - Switch con Dark Mode attiva

Figure 19 - Switch con Dark Mode disattivata

Seguendo i feedback ricevuti, è stato messo un avviso se si dovesse aprire l'applicazione con la dark mode attiva.



20.3 NICE TO HAVE

20.3.1 Tablet

Il supporto per i tablet è stato inizialmente menzionata come possibilità, ma infine scartata per dare spazio a funzionalità più importanti.

Il dispositivo principale in cui sarebbe stato utilizzato è comunque lo smartphone.

20.3.2 Portrait Mode

La possibilità di utilizzare il dispositivo in modalità portrait (in orientamento orizzontale) era anche stata menzionata ma anch'essa scartata perché meno importante.

20.3.3 Modulo Bluetooth con certificazione MFi

Vista la limitazione legata ai dispositivi Apple per il modulo Bluetooth corrente, sarebbe interessante avere un nuovo modulo certificato MFi in modo da poter sviluppare l'applicativo anche per iOS.

20.3.4 Device name

Invece dell'inserimento manuale del nome del rover, sarebbe più comodo avere un dropdown nel quale sono presenti i dispositivi trovati. Questo renderebbe la selezione più rapida ed eviterebbe eventuali errori di scrittura.

20.3.5 Fermare il rover prima di schiantarsi

Come suggerito da alcuni studenti del terzo anno, sarebbe comodo che il rover si fermasse autonomamente quando rileva che sia troppo vicino ad un ostacolo, così da evitare che ci vada a sbattere.

21 Conclusioni

21.1 Progettuali

Il progetto è stato un successo, si è conclusa l'implementazione della feature principale, ovvero controllare il rover via Bluetooth, entro i tempi prestabiliti. Ed è pure rimasto del tempo a disposizione per iniziare l'implementazione di una nuova feature. Il cliente è rimasto soddisfatto del risultato.

21.2 Personali

Mi è piaciuto molto sviluppare un'applicazione di questo tipo siccome ho potuto imparare nuove tecnologie e creare un app abbastanza unica nel suo genere.

21.3 Scolastici

La scelta delle tecnologie e la decisione di andare su Xamarin, è stata fortemente influenzata dalla conoscenza di C# acquisita negli anni scolastici e anche dalla possibilità di aver avuto esperienze con lo sviluppo di applicazioni ibride con Ionic.