MANUALE TECNICO

Paolo Forgia

SSSE SIG

INDICE

1 DESCRIZIONE TECNOLOGIE E SOFTWARE

Il software è stato creato utilizzando Xamarin, un framework per creare applicazioni Android e iOS in C#. Come ambiente di sviluppo è stato utilizzato Visual Studio, e per testare l'applicativo un emulatore e uno smartphone Android.

2 REQUISITI IN DETTAGLIO

2.1 COMANDI

I comandi sono configurati nel modo seguente:

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

Il bitrate utilizzato per la comunicazione via Bluetooth è di 9'600 bit/secondo.

2.1.1 Inviati

I comandi scambiati tra il radiocomando e il rover sono identificati da una lettera e un numero. La fine del comando viene identificato da '\n'.

La struttura dei comandi inviati è la seguente:

Descrizione	Lettera	Range valori
Velocità motore sinistro	L	• 0 – velocità massima indietro • 128 – fermo • 255 – velocità massima avanti
Velocità motore destro	R	• 0 – velocità massima indietro • 128 – fermo • 255 – velocità massima avanti
Buzzer / Clacson	B	• 0 – disattivato • 1 – attivato
Luci anteriori	F	• 0 – disattivate • 1 – attivate
Luci posteriori	P	• 0 – disattivate • 1 – attivate
Stop d'emergenza	S	• 0 – disattivato • 1 - attivato

Una volta inviato il comando per attivare lo 'Stop d'emergenza' verrà bloccata la trasmissione di elettricità ai motori.

2.1.2 Ricevuti

Il rover invierà all'applicativo i seguenti comandi:

Descrizione	Lettera	Range valori
Percentuale della batteria	T	• 0 – Batteria scarica • 100 – Batteria carica
Distanza da un ostacolo	D	• 0 – 0 cm di distanza • 100 – 100 cm di distanza

2.1.3 Completamento
Tutti i requisiti legati ai comandi inviati e ricevuti sono completati al 100%.

2.2 Dispositivi
I dispositivi da supportare inizialmente erano tutti gli smartphone moderni, i quali si possono categorizzare in due gruppi identificati dal sistema operativo: Android e iOS.

2.2.1 Revisione dei dispositivi
Dopo una fase di analisi iniziale si è notato che il modulo Bluetooth in uso non aveva la certificazione MFi, la quale è necessaria per la comunicazione con i dispositivi iOS.

Vista l'impossibilità di cambiare il modulo Bluetooth si è deciso di scartare i dispositivi con iOS dai dispositivi da supportare.

2.2.2 Tablet
Inizialmente si era anche pensato come requisito con importanza bassa, il supporto a tablet.

Questo requisito è stato poi rimandato per dare spazio ad altre funzionalità ritenute più importanti e per assicurarsi che esse venissero portate a termine.

2.2.3 Completamento
Il supporto ad Android per smartphone è stato completato al 100%, mentre quello per iOS e per tablet non è mai stato iniziato.

Si può però stimare che il supporto per iOS sia comunque attorno ad un 70%, siccome il codice scritto è per il 98% in comune, ma per abilitare il supporto sarebbe necessario cambiare modulo Bluetooth e quindi potenzialmente riscrivere una parte della configurazione di esso.

Per quanto riguarda il supporto al tablet non è stato testato ma l'interfaccia è responsive, ovvero si adatta automaticamente alle varie dimensioni dei dispositivi. Per quanto riguarda il funzionamento non dovrebbe presentare problemi, potrebbero esserci bisogno di adattamenti grafici per migliorare l'esperienza d'uso su schermi così grandi.

2.3 Configurazione

2.3.1 Bitrate
Uno dei requisiti era di configurare il modulo Bluetooth e, se necessario adattare il codice dell'app, per far funzionare la comunicazione con un bitrate di 115'200 bit al secondo, ovvero il bitrate massimo supportato dal modulo HC-05. Il bitrate di default era di 9'600.

2.3.2 Nome dispositivo
Siccome ci saranno molteplici rover che potranno essere controllati, è necessario poter configurare ogni modulo con il proprio nome e configurare l'applicativo per ricercare il dispositivo con il nome definito.

2.3.3 Intervallo invio comandi

È stato impostato un intervallo di tempo fisso (non configurabile) come delay per l'invio a ripetizione dei comandi. Alcuni comandi, come per il motore (sinistro e destro) e per il buzzer (quando attivo) devo essere inviati ripetutamente.

Questo è stato fatto per far si che se il rover dovesse perdere il segnale o scollegarsi si fermerebbe e non continuasse ad eseguire gli ultimi comandi ricevuti.

Quindi il rover per capire che deve andare in avanti, deve ricevere il comando per andare avanti per il motore sinistro e destro ripetutamente. L'intervallo non è stato aggiunto per limitazioni ma semplicemente per non sovraccaricare il modulo di dati in entrata e uscita.

È stato impostato un delay di 100 millisecondi che garantisce di non rischiare di sovraccaricare il modulo ma allo stesso tempo è abbastanza basso da non notarlo quando il rover viene pilotato.

Questo delay è presente anche sul software montato dal rover, il quale invierà al telefono lo stato della batteria e la distanza da un ostacolo con lo stesso intervallo di tempo.

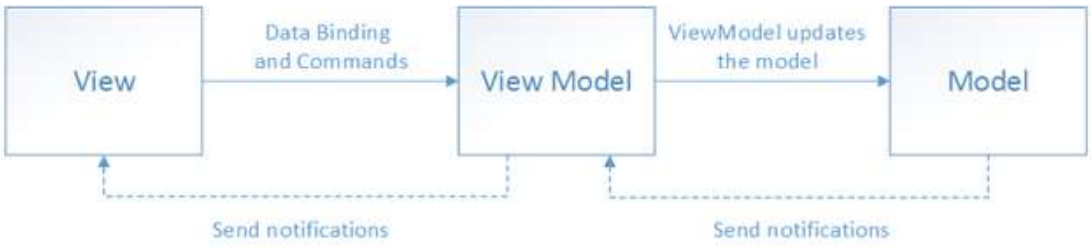
2.3.4 Completamento

Tutti i requisiti legati alla configurazione di bitrate, nome del dispositivo e intervallo nei comandi ricevuti e inviati sono completati al 100%.

3 STANDARD DI SVILUPPO

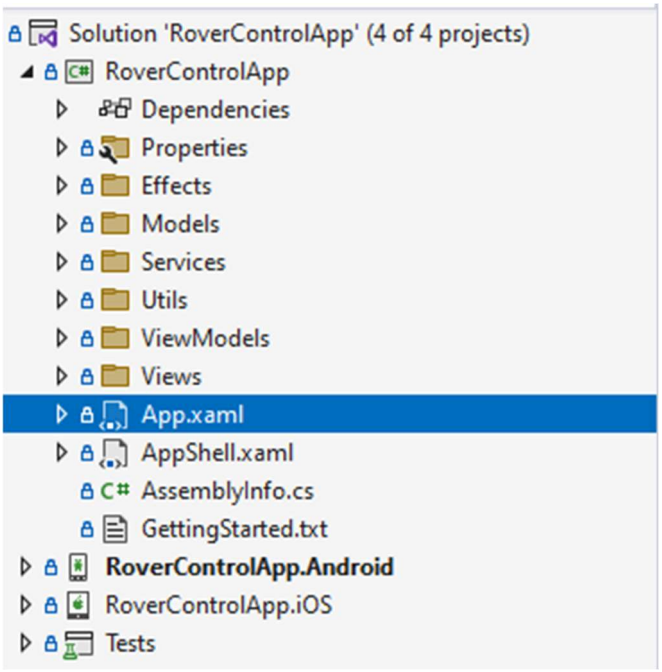
3.1 SUDDIVISIONE CODICE

Si è utilizzato il pattern Model-View-ViewModel (MVVM) il quale aiuta a tenere separata la logica dall'interfaccia.



Nonostante l'utilizzo del pattern MVVM è stato necessario collocare alcune parti di logica anche nella view, questo è dovuto al fatto che il comportamento di alcune azioni è fortemente dipendente dallo stato dell'interfaccia. In particolar modo per le pressioni prolungate, il quale non sono gestibili dal View Model e il cambio di valore degli Slider (per il motore sinistro e destro).

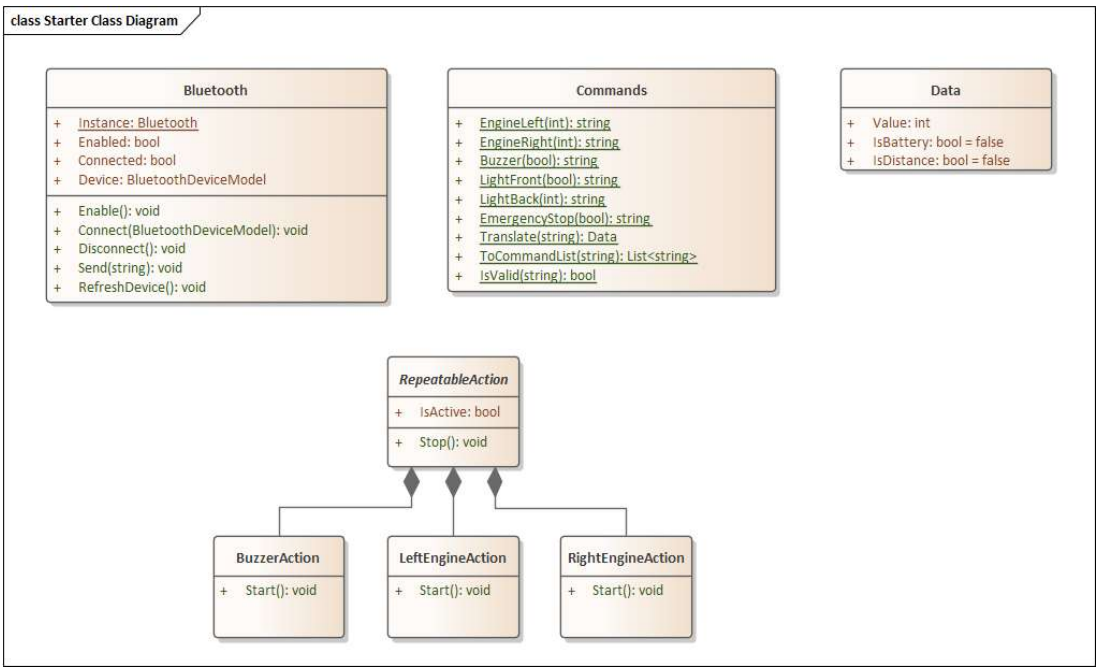
Si è però cercato di tenere la struttura il più simile possibile a quella prevista dal MVVM.



4 DOCUMENTAZIONE CODICE

4.1 DIAGRAMMA DELLE CLASSI

Una forma di documentazione del codice più ad alto livello è il diagramma delle classi, il quale mostra alcune classi, e le loro proprietà e funzioni.



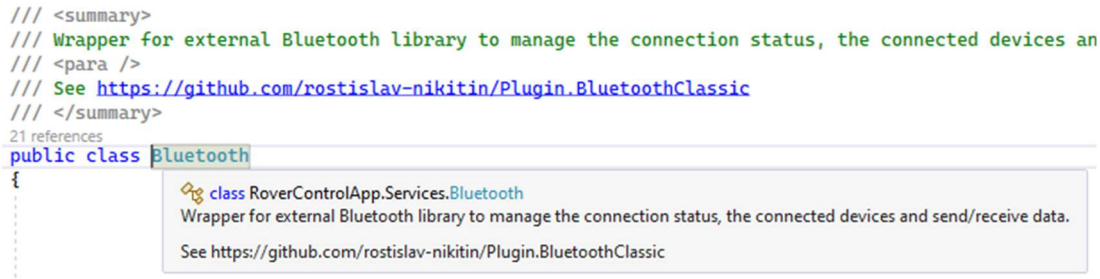
4.2 COMMENTI

Per commentare il codice sono stati utilizzati i commenti in formato XML (XML Doc) i quali vengono scritti nel modo seguente:

```
/// <summary>
///  Questo è un commento in formato XML.
/// </summary>
```

È stato scritto un commento all'inizio di ogni classe per spiegarne la funzione.

Esempio di commento con il codice XML e il risultato interpretato da Visual Studio.



5 Bluetooth

Ci sono varie nomenclature per varie tecnologie Bluetooth e sono variate nel corso degli anni. Bluetooth Classic e Bluetooth Low Energy possono sembrare molto simili inizialmente ma sono protocolli molto diversi e incompatibili uno con l'altro.

5.1 Bluetooth Classic

Bluetooth Classic è sostanzialmente Bluetooth prima del 2010, ovvero prima della versione 4.0 che ha introdotto la nuova variante Low Energy (o LE).

5.2 Bluetooth Low Energy

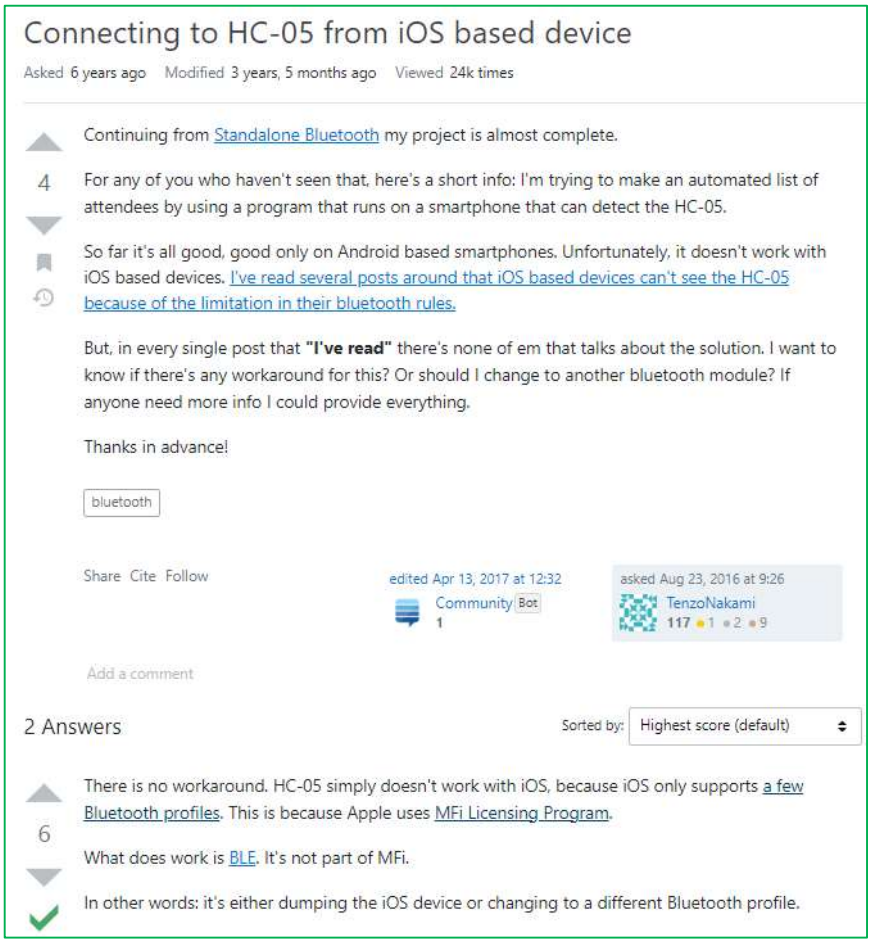
Dal 2010 è stata introdotta una nuova versione a basso consumo, Bluetooth Low Energy (o BLE). La nuova versione però non rimuove il supporto per la versione precedente, questo permette a dispositivi con le nuove versioni Bluetooth di collegarsi con dispositivi che usano una versione inferiore alla 4.0.[1]

5.3 Bluetooth Classic ed Apple MFi

I dispositivi che utilizzano Bluetooth Classic necessitano di una certificazione MFi[2] per poter comunicare con un dispositivo Apple. MFi certifica che il dispositivo è conforme agli standard imposti da Apple.

Il rover monta il modulo Bluetooth HC-05, il quale non è certificato da Apple, questo rende impossibile comunicare da un dispositivo iOS al rover[3]. Per questo motivo iOS è stato scartato dal target per lo sviluppo dell'app.

[1] https://blog.nordicsemi.com/getconnected/the-difference-between-classic-bluetooth-and-bluetooth-low-energy
[2] Made for iPhone/iPod/iPad
[3] https://stackoverflow.com/a/12201785/1685157



Connecting to HC-05 from iOS based device

Asked 6 years ago Modified 3 years, 5 months ago Viewed 24k times

4

Continuing from Standalone Bluetooth my project is almost complete.

For any of you who haven't seen that, here's a short info: I'm trying to make an automated list of attendees by using a program that runs on a smartphone that can detect the HC-05.

So far it's all good, good only on Android based smartphones. Unfortunately, it doesn't work with iOS based devices. I've read several posts around that iOS based devices can't see the HC-05 because of the limitation in their bluetooth rules.

But, in every single post that **"I've read"** there's none of em that talks about the solution. I want to know if there's any workaround for this? Or should I change to another bluetooth module? If anyone need more info I could provide everything.

Thanks in advance!

bluetooth

Share Cite Follow

edited Apr 13, 2017 at 12:32
Community Bot
1

asked Aug 23, 2016 at 9:26
TenzoNakami
117 ● 1 ● 2 ● 9

Add a comment

2 Answers Sorted by: Highest score (default)

6

There is no workaround. HC-05 simply doesn't work with iOS, because iOS only supports a few Bluetooth profiles. This is because Apple uses MFi Licensing Program.

What does work is BLE. It's not part of MFi.

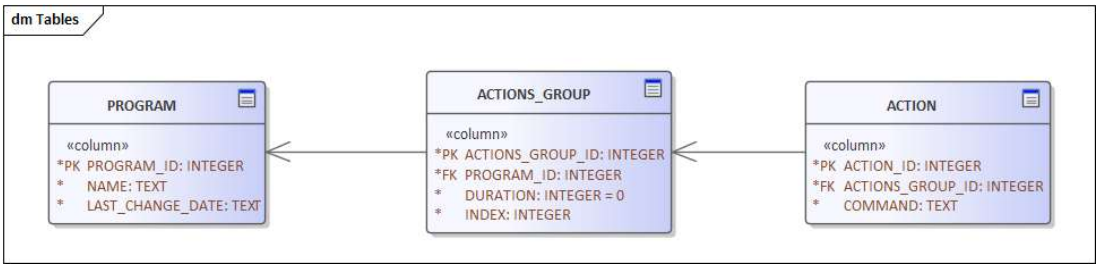
In other words: it's either dumping the iOS device or changing to a different Bluetooth profile.

Immagine 1 - Modulo HC-05 non certificato MFi - https://electronics.stackexchange.com/a/253712

6 DATABASE

Per il database si è utilizzato SQLite in quanto suggerito da Xamarin per lo sviluppo mobile.

Sono stati creati anche dei diagrammi ER per facilitare la creazione del database e le loro relazioni.



7 TESTING

7.1 UNITARI

Sono stati creati dei test unitari per la creazione dei comandi da inviare, e l'analisi dei comandi in entrata.

7.1.1 Creazione comandi

Lo scopo è di assicurarsi che la classe generi il comando corretto passando i vari parametri. E che non ritorni un comando nel caso venisse passato un parametro non valido.

7.1.2 Analisi comandi ricevuti

Lo scopo è di assicurarsi che il codice eseguisse correttamente i seguenti controlli.

Test	Criteri di accettazione
Comando è valido?	• Contiene solo caratteri validi • Termina con "\n" • La lettera che identifica il comando è maiuscola • Il comando è tra quelli riconosciuti
Il comando ha un valore accettato?	• A dipendenza del tipo di comando, il valore può essere compreso da 0 a 1, o da 0 a 255
Viene identificato il comando?	• Il tipo di comando viene identificato correttamente • Il valore viene identificato correttamente

7.2 HTERM

HTerm è un programma che permette di visualizzare i dati scambiati su una porta USB selezionata.

Nel nostro caso leggerà i comandi ricevuti sul modulo Bluetooth, il quale sarà collegato via USB al computer.

Sarà possibile, inoltre, inviare dei comandi i quali verranno trasmessi dal modulo Bluetooth al telefono.

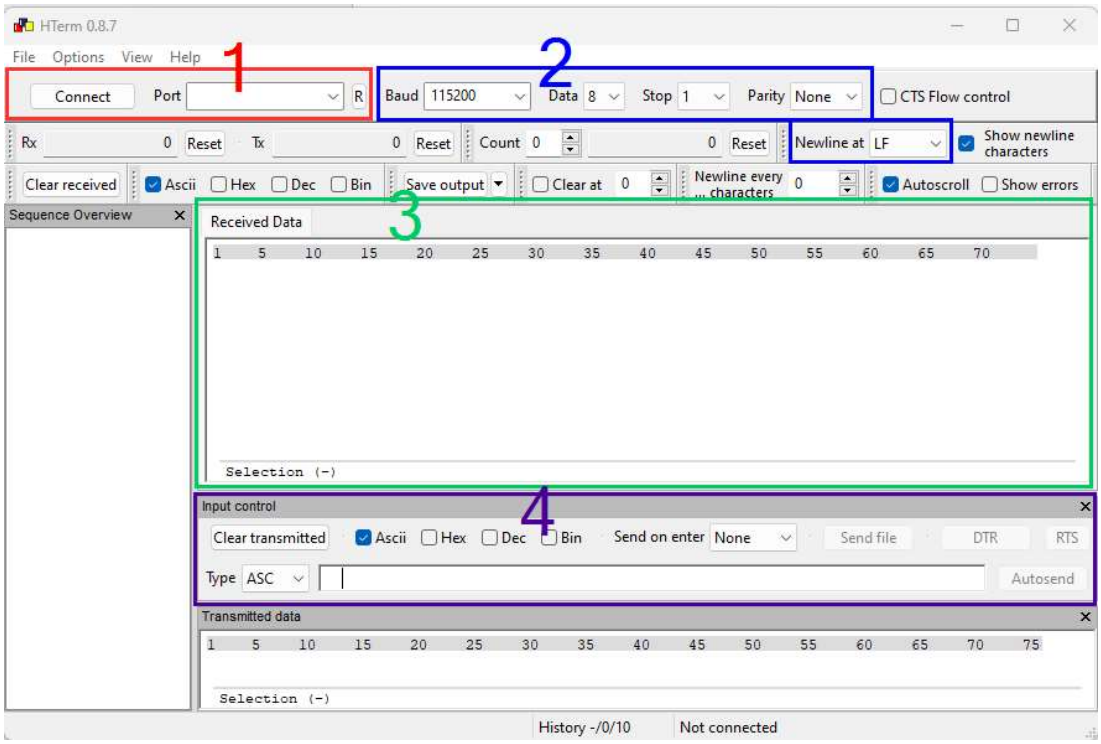


Figure 1 - Interfaccia di HTerm

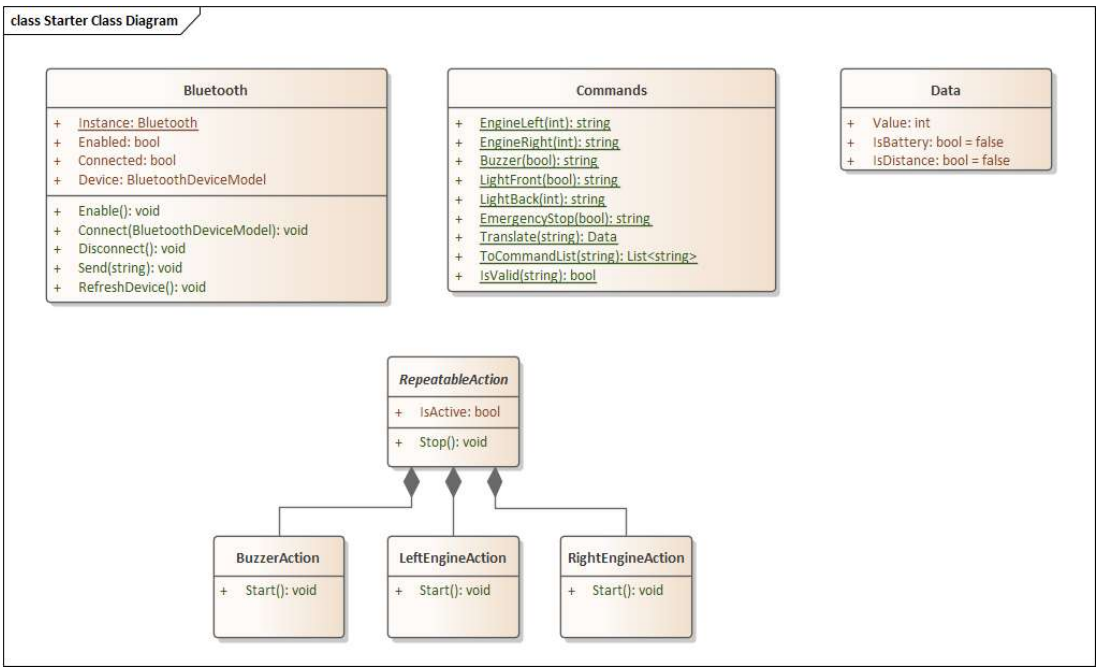
1. Selezione della porta alla quale collegarsi (refresh "R")
2. Impostazioni riguardanti la comunicazione
    a. Baud: 115'200 (o come configurato sul modulo Bluetooth)
    b. Data: 8 (bit)
    c. Stop: 1 (bit)
    d. Parity: None (nessuna parità)
    e. Newline at: LF (per visualizzare un comando per linea)
3. Lista dei comandi ricevuti
4. Input per inserire i comandi da inviare
    a. Selezionare "LF" su "Send on enter" così da aggiungere "\n" alla fine del comando

Il video dimostrazione è situato nella cartella Allegati/Video/HTerm.mp4.

8 CODICE SIGNIFICATIVO

8.1 DIAGRAMMA DELLE CLASSI

Per dare una visione più chiara di alcune classi è stato creato una diagramma delle classi, nel quale vengono mostrate le classi più importanti, le loro funzioni e le loro dipendenze da altre classi. Alcune sono state approfondite con del codice significativo nei prossimi capitoli.



8.2 GESTIONE BLUETOOTH

È stata creata una classe che fa da wrapper alla libreria utilizzata, in modo da facilitarne l'uso e nel caso in futuro si volesse cambiare libreria non si debba andare a toccare altre parti del codice.

```
public class Bluetooth
{
    private static readonly Bluetooth instance = new Bluetooth();

    1 reference
    public Recived OnReceiveEvent { get; set; }
    1 reference
    public StateChanged OnStateChangedEvent { get; set; }

    private readonly IBluetoothAdapter bluetoothAdapter;
    private BluetoothDeviceModel device;
    private IBluetoothManagedConnection connection;

    1 reference
    public void Connect(BluetoothDeviceModel bluetoothDeviceModel)
    {
        connection = bluetoothAdapter.CreateManagedConnection(bluetoothDeviceModel);
        connection.Connect();
        connection.OnRecived += OnReceiveEvent;
        connection.OnStateChanged += OnStateChangedEvent;
    }

    2 references
    public void Disconnect()
    {
        connection?.Dispose();

        connection = null;
    }

    13 references
    public void Send(string command)
    {
        if (connection == null) return;

        connection.Transmit(new Memory<byte>(Encoding.ASCII.GetBytes(command)));
    }
```

8.3 INVIO DEI COMANDI A RIPETIZIONE

Per inviare i comandi in modo asincrono e ripetibile fino a che non venisse richiesto di essere fermato, è stata creata una classe astratta la quale crea un Task.

Il task esegue un'operazione asincrona alla quale è possibile registrare un cancelation token, ovvero un token il quale notifica il task che tale operazione è stata cancellata.

```
public abstract class RepeatableAction
{
    protected static readonly int DELAY = 100;
    protected CancellationTokenSource tokenSource;
    protected Bluetooth bluetooth;

    4 references
    public bool IsActive => tokenSource != null;

    3 references
    public RepeatableAction()
    {
        bluetooth = Bluetooth.Instance;
    }

    3 references
    protected void Start(Action<CancellationToken> action, Action onStop)
    {
        tokenSource = new CancellationTokenSource();
        tokenSource.Token.Register(() => onStop());
        Task.Run(() => action(tokenSource.Token), tokenSource.Token);
    }

    5 references
    public void Stop()
    {
        tokenSource.Cancel();
    }
}
```

Il metodo “Start” si occupa della creazione del Task e riceve come parametro due metodi, il primo è l'operazione asincrona che viene eseguita, mentre la seconda è un'operazione che verrà eseguita quando la cancellazione del task sarà chiamata.

```
public class BuzzerAction: RepeatableAction
{
    1 reference
    public BuzzerAction() : base()
    {
    }

    1 reference
    public void Start()
    {
        Start(RunBuzzer, OnBuzzerStop);
    }

    1 reference
    void RunBuzzer(CancellationToken cancellationToken)
    {
        while (!cancellationToken.IsCancellationRequested)
        {
            bluetooth.Send(Commands.Buzzer(true));
            Thread.Sleep(DELAY);
        }
    }
    1 reference
    void OnBuzzerStop()
    {
        bluetooth.Send(Commands.Buzzer(false));
    }
}
```

La classe che implementa quella astratta dovrà creare i due metodi da passare a “Start”, ovvero il comando da inviare via Bluetooth in loop, e il comando da inviare quando il task deve essere terminato.

```
0 references
void OnBuzzerPressed(object sender, EventArgs args)
{
    buzzerAction.Start();
}

0 references
void OnBuzzerReleased(object sender, EventArgs args)
{
    buzzerAction.Stop();
}
```

Quando il bottone del buzzer verrà premuto “OnBuzzerPressed” verrà eseguito il quale darà il via al task. E quando il bottone verrà rilasciato verrà eseguito “OnBuzzerReleased” che terminerà il task.

9 GESTIONE SICUREZZA

Sul modulo Bluetooth è impostato un PIN, il quale può essere configurato. Questo previene che una persona non autorizzata si possa collegare al rover.

10 MIGRAZIONE DATI

Non sono presenti dati da migrare.

11 DEPLOYMENT

Visual Studio andrà a creare una nuova versione dell'applicazione nel formato apk[4]. Il deployment non avverrà sullo store, bensì verrà generato il file sul computer e lo si potrà inviare sul telefono tramite una piattaforma di file hosting (Dropbox, Google Drive, ecc.).

12 FORMAZIONE UTENTE

Non è necessaria una formazione dell'utente. Agli studenti a cui è destinata l'applicazione verranno spiegate le funzionalità che avrà il rover mentre lo dovranno costruire. Una volta capito cosa può fare il rover l'app diventa auto-esplicativa.

[4] Formato dei file eseguibili in Android