UNIVERSITÀ DEGLI STUDI
DI GENOVA

DIPARTIMENTO DI INGEGNERIA NAVALE, ELETTRICA, ELETTRONICA
E DELLE TELECOMUNICAZIONI

CORSO DI STUDIO IN INGEGNERIA ELETTRONICA E TECNOLOGIE
DELL'INFORMAZIONE

Tesi di Laurea Triennale

Settembre 2024

Progetto e implementazione di un sistema embedded per il controllo remoto di auto radiocomandate

Design and implementation of an embedded system for remote control of radio-controlled cars

Candidato: Lorenzo Sgrò

Relatore: Prof. Riccardo Berta

Correlatore: Dr. Matteo Fresta

Sommario

La presente tesi propone di analizzare e sviluppare un sistema di controllo per una macchina radiocomandata, utilizzando la tecnologia Bluetooth Low Energy (BLE) tramite un Arduino Nano 33 BLE Sense e un'applicazione Flutter.

Il sistema consente di gestire le funzioni di sterzata e accelerazione, inviando comandi dall'app Flutter all'Arduino tramite Bluetooth. L'Arduino, installato a bordo della macchina, controlla un servomotore per lo sterzo e un ESC (Electronic Speed Control) per regolare la velocità del motore.

Un meccanismo di sicurezza integrato garantisce l'inattività della macchina in caso di perdita di connessione o assenza di comandi per un certo periodo, prevenendo comportamenti anomali.

Prima dello sviluppo del software, sono state effettuate sessioni di laboratorio per analizzare i segnali generati dalla ricevente originale della macchina e il loro effetto su servomotore ed ESC. Una volta compresi i segnali, questi sono stati replicati tramite l'Arduino, modificandoli in base ai comandi ricevuti dall'applicazione.

Il lavoro include una descrizione dettagliata della configurazione hardware, della logica di programmazione e delle tecniche di comunicazione BLE, con l'obiettivo di fornire una base solida per la progettazione di veicoli radiocomandati sicuri e avanzati.

Indice

1 Introduzione

Nel panorama della robotica e delle tecnologie di controllo, i veicoli radiocomandati (RC) rappresentano una piattaforma ideale per sperimentare e implementare soluzioni innovative. Questi veicoli offrono un terreno fertile per l'applicazione di nuove tecnologie di controllo e comunicazione, permettendo l'esplorazione di tecniche avanzate a costi relativamente contenuti. L'uso di veicoli radiocomandati come banchi di prova per le tecnologie emergenti è ben consolidato, poiché consente di testare e perfezionare sistemi senza le complessità e i costi associati a veicoli più complessi e costosi. Questo lavoro si inserisce in un contesto di ricerca più ampio, volto allo sviluppo di un sistema di guida autonoma, e si focalizza sulla progettazione e realizzazione di un sistema di controllo remoto per una macchina radiocomandata, costituendo il punto di partenza di un progetto ambizioso.

L'obiettivo principale di questa tesi è stato quello di modernizzare una macchina radiocomandata esistente, sostituendo il suo meccanismo di comunicazione originale con una soluzione basata su Bluetooth Low Energy (BLE) e un microcontrollore Arduino Nano 33 BLE Sense. Il veicolo originale in figura 1, comprensivo di telaio, servomeccanismo di sterzo e motore, è stato conservato nella sua forma originale, mentre è stato completamente riprogettato il sistema di ricezione e controllo. Questo approccio ha permesso di mantenere l'integrità strutturale e funzionale del veicolo, focalizzandosi invece sull'aggiornamento della sua elettronica di controllo. La scelta del Bluetooth Low Energy (BLE) come tecnologia di comunicazione è stata dettata dalla sua efficienza energetica e dalla sua capacità di gestire comunicazioni a bassa latenza, cruciali per il controllo preciso del veicolo.

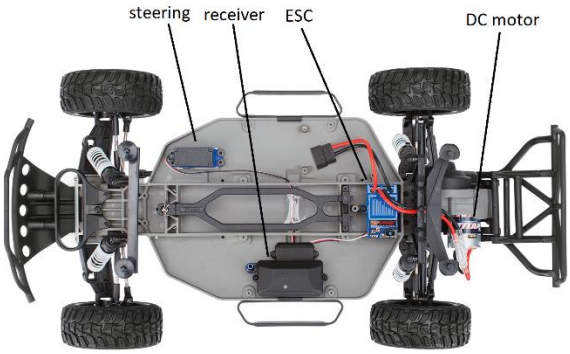

Figura 1: Componenti principali della macchina

Un elemento cruciale di questo progetto è stata l'analisi approfondita dei segnali prodotti dalla ricevente originale del veicolo. Utilizzando un oscilloscopio, è stato possibile studiare i segnali emessi dalla ricevente e comprendere come questi influenzassero il servomotore e l'ESC (Electronic Speed Control) per gestire le funzioni di sterzata e accelerazione. Questo processo di decodifica e comprensione dei segnali è stato fondamentale per replicare e adattare i comandi nel nuovo sistema di controllo (illustrato in figura 2) basato su Arduino e BLE.

L'accuratezza nella ricostruzione dei segnali è stata essenziale per garantire che il nuovo sistema di controllo potesse emulare con precisione il comportamento del sistema originale, assicurando una transizione fluida e senza problemi. Questa fase di analisi ha incluso anche la calibrazione dei segnali per assicurare che il controllo del servomotore e dell'ESC fosse perfettamente sincronizzato con le nuove istruzioni inviate tramite BLE.

Figura 2: Schema funzionale del progetto

Lo sviluppo futuro del progetto prevede l'espansione verso l'implementazione della guida autonoma, che rappresenta una delle frontiere più avanzate nella robotica e nella tecnologia dei veicoli. Questa evoluzione comporterà l'integrazione di sensori ambientali attraverso l'uso di Arduino, che saranno in grado di raccogliere dati in tempo reale, come la distanza dagli ostacoli, la velocità e l'orientamento del veicolo. Questi dati saranno inviati al dispositivo di controllo tramite un app Flutter, che gestirà sia la comunicazione in tempo reale con il veicolo sia la trasmissione dei dati raccolti su un sistema cloud. L'uso del cloud permetterà la consultazione e l'elaborazione dei dati, facilitando l'analisi e l'ottimizzazione delle performance del sistema di guida autonoma. Inoltre, l'analisi dei dati raccolti fornirà preziose informazioni per affinare ulteriormente gli algoritmi di guida autonoma e migliorare l'affidabilità e la sicurezza del sistema.

La tesi è organizzata in tre sezioni principali. La prima sezione, "Metodi e strumenti utilizzati", descrive in dettaglio il processo di realizzazione del progetto, comprese le tecnologie e gli strumenti necessari, come Arduino e BLE, e le tecniche di decodifica dei segnali. La seconda sezione, "Sperimentazioni e risultati", analizza i risultati ottenuti durante le fasi di test, valutando l'accuratezza e l'efficacia del sistema di controllo implementato, e discute le sfide affrontate e le soluzioni adottate. Infine, la sezione "Contributo personale e considerazioni conclusive" offre una riflessione sulle conclusioni finali, discutendo il contributo personale al progetto e le implicazioni future del lavoro svolto, con particolare attenzione agli sviluppi futuri e alle possibili evoluzioni del progetto.

2 Metodi e strumenti utilizzati

2.1 Decodifica dei segnali della ricevente originale con oscilloscopio

Il primo passo fondamentale in questo progetto è stato comprendere come il sistema di controllo originale del veicolo radiocomandato (RC) operava. Per farlo, è stato utilizzato un oscilloscopio, uno strumento essenziale per analizzare i segnali elettrici trasmessi dalla ricevente al servomotore e all'ESC (Electronic Speed Control). L'oscilloscopio è stato configurato per catturare e visualizzare i segnali in tempo reale, permettendo un'analisi dettagliata della frequenza, ampiezza e forma d'onda dei comandi inviati dalla ricevente.

Il segnale principale analizzato era un segnale PWM (Pulse Width Modulation) visibile in figura 3, utilizzato comunemente nei sistemi RC per trasmettere informazioni relative alla posizione del servomotore e alla velocità del motore.

Decodificare questo segnale è stato cruciale per capire come replicare questi comandi nel nuovo sistema di controllo basato su Arduino. L'analisi ha rivelato che il segnale con frequenza 100 Hz ed ampiezza 6 V variava in funzione della posizione desiderata del servomotore o della velocità del motore, con impulsi di lunghezza variabile tra 1 mS e 2 mS, dove 1,5 mS rappresenta la posizione centrale o velocità zero.

Questa fase ha richiesto diverse sessioni di analisi per garantire che i segnali fossero correttamente interpretati. Il risultato è stato un set di dati che è servito da base per la successiva programmazione del microcontrollore Arduino Nano 33 BLE Sense.

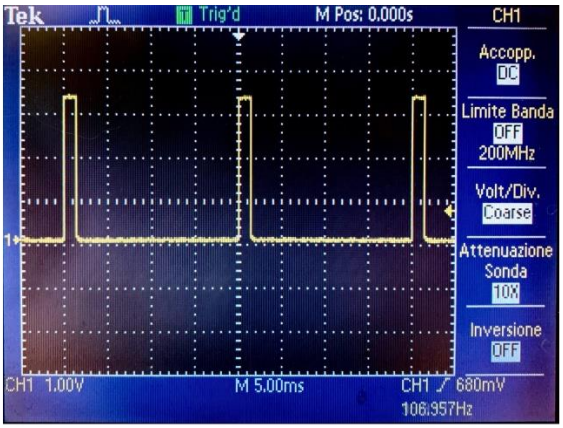

Figura 3: Segnale PWM rilevato

2.2 Funzionamento del veicolo

2.2.1 ESC (Electronic Speed Control)

L'ESC è un componente vitale nei veicoli RC, responsabile della gestione della potenza fornita al motore elettrico. Il suo funzionamento si basa sulla ricezione di segnali PWM, che determinano la velocità e la direzione del motore. In questo progetto, è stato necessario comprendere esattamente come l'ESC originale interpretava i segnali della ricevente e replicare questo comportamento con il microcontrollore Arduino.

L'ESC utilizzato nel veicolo RC esaminato è compatibile con motori DC, comunemente usati per la loro efficienza e potenza. L'ESC riceve il segnale PWM e lo traduce in una corrente modulata fornita al motore, controllando così la velocità di rotazione.

È stato necessario integrare correttamente l'ESC nel nuovo sistema di controllo, assicurando che i comandi inviati dall'Arduino fossero accuratamente interpretati dall'ESC per evitare problemi di controllo, come accelerazioni non lineari o comportamenti anomali.

Durante la progettazione, è stato necessario integrare l'ESC nel nuovo sistema di controllo mantenendo il cablaggio originale, che comprende tre collegamenti principali:

Il primo è collegato direttamente alla batteria del veicolo, fornendo l'alimentazione necessaria per il motore; il secondo è collegato al motore tramite due fili, positivo e negativo, che trasmettono la potenza modulata per

controllare la velocità di rotazione; il terzo connettore è un cavo a tre fili (positivo, massa e segnale) che originariamente si collegava alla ricevente del veicolo.

Nel nuovo sistema, questo terzo connettore è stato collegato all'Arduino, visibile in figura 4, che ora genera i segnali di controllo. Essendo l'ESC alimentato dalla batteria, non è stato necessario progettare componenti aggiuntivi per la gestione dell'alimentazione.

Nella foto sotto si può vedere lo schema di collegamento utilizzato, che è stato realizzato senza modifiche al cablaggio già presente sulla macchina, assicurando così una facile integrazione con il nuovo sistema di controllo basato su Arduino.

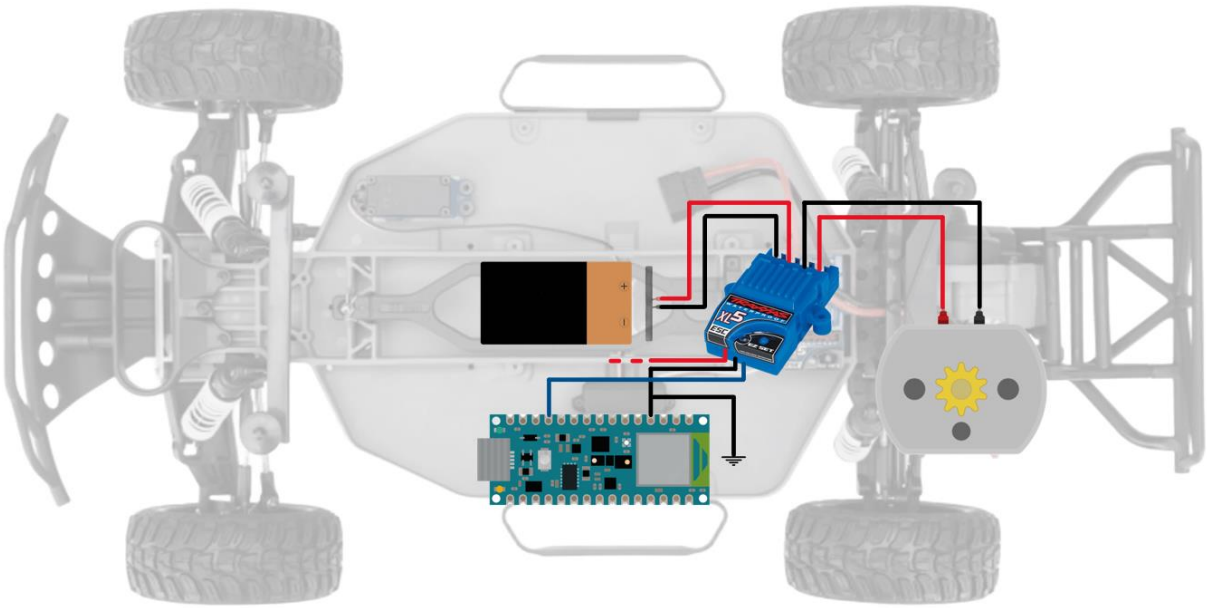

Figura 4: Schema di collegamento dell'ESC

2.2.2 Servomotore

Il servomotore, visibile in figura 5, è un componente cruciale per la sterzata del veicolo radiocomandato, fornendo una regolazione precisa dell'angolo di sterzo. A differenza dei motori tradizionali, progettati principalmente per la rotazione continua, i servomotori sono ottimizzati per offrire un controllo accurato della posizione angolare dell'albero motore.

Questo controllo è ottenuto tramite un feedback interno che confronta la posizione attuale dell'albero con quella desiderata e regola il movimento di conseguenza.

Il funzionamento di un servomotore si basa sul segnale PWM (Pulse Width Modulation), dove la larghezza dell'impulso determina la posizione angolare dell'albero motore. Per i veicoli radiocomandati, un impulso di 1 MS sposta il servomotore alla posizione minima (completamente a sinistra), mentre un impulso di 2 MS lo porta alla posizione massima (completamente a destra). Un impulso di 1,5 MS posiziona l'albero motore al centro.

Normalmente, la frequenza del segnale PWM è di 50 Hz, ma nel nostro caso, il servomotore utilizzato opera con un segnale PWM a 100 Hz.

I servomotori sono dotati di un potenziometro interno collegato all'albero motore, che fornisce un feedback continuo sulla posizione attuale. Questo meccanismo consente al circuito di controllo interno del servomotore di correggere eventuali errori di posizione, assicurando che il servomotore raggiunga e mantenga con precisione la posizione desiderata.

Tale controllo è particolarmente vantaggioso in applicazioni che richiedono risposte rapide e accurate ai comandi di input, come la sterzata di veicoli RC, dove ogni piccola variazione dell'angolo di sterzo può influenzare significativamente la manovrabilità del veicolo.

Figura 5: Foto del servomotore

2.3 Sostituzione della ricevente con Arduino Nano

La sostituzione della ricevente originale del veicolo radiocomandato con un microcontrollore Arduino Nano 33 BLE Sense ha richiesto una riorganizzazione dei collegamenti hardware per integrare il nuovo sistema di controllo.

Il cablaggio è stato progettato per mantenere la compatibilità con l'ESC (Electronic Speed Control) e il servomotore esistenti e per garantire la possibilità di rendere reversibili le modifiche rimontando la ricevente originale nel caso in cui ce ne fosse la necessità.

2.3.1 Cablaggio dell'ESC

Il cablaggio dell'ESC (Electronic Speed Control) è stato modificato come segue:

Connessione alla Batteria e al Motore: Le due spine dell'ESC collegate direttamente alla batteria e al motore sono rimaste invariate.

Connessione alla ricevente: La spina dell'ESC precedentemente collegata alla ricevente originale, composta da tre fili (positivo, negativo e segnale), è stata adattata per il nuovo sistema. In particolare:

Positivo: il positivo è stato collegato direttamente al servomotore per alimentarlo.

Negativo: Il filo negativo dell'ESC è stato collegato alla massa dell'Arduino. Questo collegamento è essenziale per garantire un comune riferimento di terra tra l'ESC e l'Arduino, assicurando la corretta interpretazione e trasmissione dei segnali.

Segnale: Il filo di segnale dell'ESC è stato collegato a un pin specifico dell'Arduino. La scelta di questo pin e la sua configurazione saranno dettagliate nei paragrafi successivi, dove verranno spiegate le motivazioni tecniche di questa scelta.

2.3.2 Cablaggio del servomotore

Il cablaggio del servomotore è relativamente semplice rispetto a quello dell'ESC. Il servomotore è dotato di una spina a tre fili, che devono essere correttamente collegati per garantire un funzionamento ottimale. La configurazione del cablaggio è stata effettuata come segue:

Positivo: Il filo positivo del servomotore è stato collegato al positivo della spina a tre fili dell'ESC, come già indicato sopra. Questo collegamento è fondamentale per fornire la tensione necessaria al servomotore, permettendogli di operare correttamente.

Negativo e Segnale: Gli altri due fili, negativo e segnale, sono stati collegati all'Arduino. In particolare:

Negativo: Il filo negativo del servomotore è stato collegato alla massa dell'Arduino, assicurando un riferimento di terra comune tra il servomotore e il microcontrollore.

Segnale: Il filo di segnale del servomotore è stato collegato a un pin specifico dell'Arduino, simile a quanto fatto per l'ESC. Questo pin sarà utilizzato per trasmettere i segnali PWM che controllano la posizione del servomotore.

Per una maggior chiarezza è riportato in figura 6, visibile sotto, lo schema di collegamento completo.

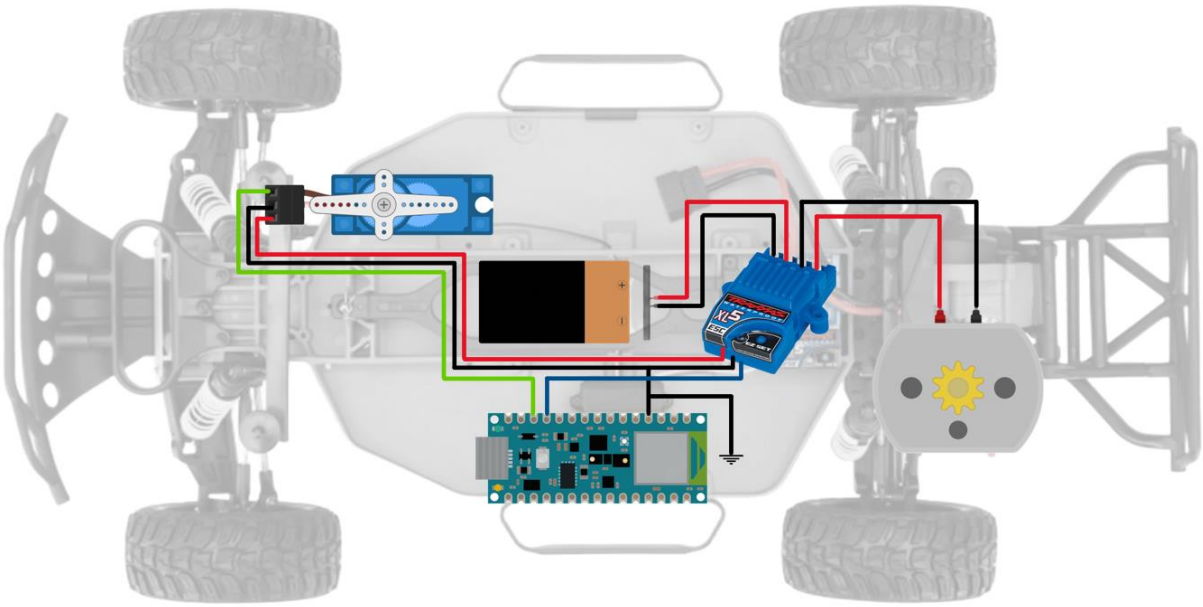

Figura 6: Schema di collegamento completo

2.3.3 Gestione dell'energia

La gestione dell'energia nel nuovo sistema ha richiesto un'attenzione particolare per garantire la compatibilità tra i diversi componenti e prevenire danni al microcontrollore.

Nel sistema originale, la ricevente del veicolo RC veniva alimentata direttamente dall'ESC, che a sua volta era collegato alla batteria principale del veicolo, con una tensione di 6 V. Questa tensione alimentava sia la ricevente che il servomotore, utilizzando un positivo comune.

Tuttavia, poiché l'Arduino Nano opera a una tensione di 3,3 V, collegarlo direttamente al sistema avrebbe potuto causare danni irreparabili al microcontrollore. Per risolvere questo problema, è stata implementata una scheda a batteria dedicata, sviluppata precedentemente in laboratorio durante una tesi, che è stata saldata sull'Arduino, visibile in figura 7.

Questa scheda, alimentata da una batteria a bottone, è in grado di fornire energia all'Arduino per circa tre ore, permettendo di disaccoppiare i due livelli di tensione e proteggere il microcontrollore. In questo modo, il veicolo ora ospita due sistemi di alimentazione separati:

la batteria principale, che alimenta i motori e l'ESC, e una batteria a bottone con una tensione più bassa, dedicata esclusivamente all'alimentazione dell'Arduino.

Questo approccio ha garantito un funzionamento sicuro e stabile del sistema, prevenendo interferenze tra i circuiti a tensioni diverse e assicurando che l'Arduino possa operare senza rischi di sovratensione.

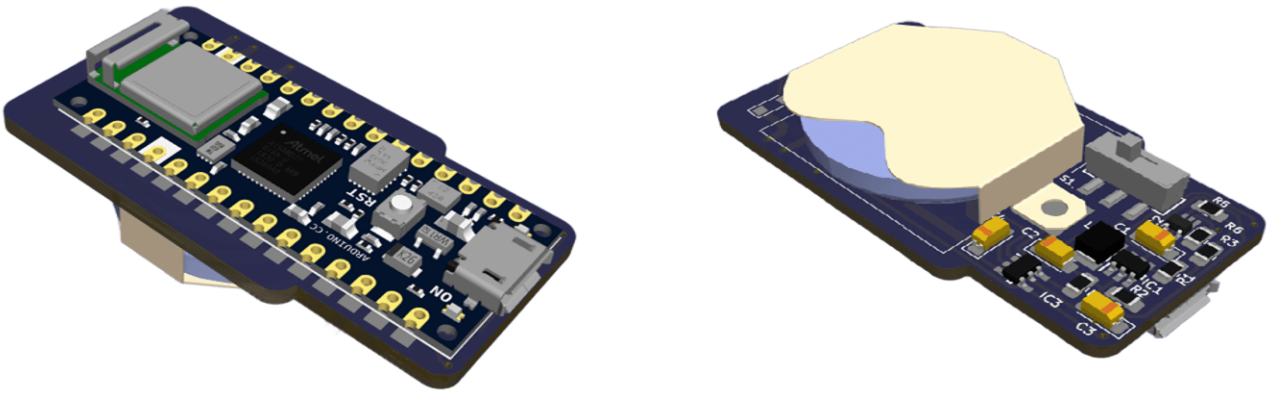

Figura 7: Arduino Nano 33 BLE Sense con scheda dedicata

2.4 Controllo dei motori tramite Arduino

Successivamente alla sostituzione della ricevente con l'Arduino Nano 33 BLE Sense, è stata avviata la fase di programmazione del microcontrollore per testare e replicare i movimenti del veicolo radiocomandato. Utilizzando il segnale PWM generato da Arduino, è stata implementata una routine iniziale per verificare la correttezza del sistema.

2.4.1 Test del servomotore

Questa routine, visibile in figura 8, prevedeva il movimento progressivo del servomotore da un'estrema posizione di sterzata a sinistra a un'estrema posizione di sterzata a destra, e viceversa, con incrementi di 5 gradi. Tale movimento a passi è stato scelto per simulare un comportamento controllato e fluido, consentendo di monitorare e valutare l'interpretazione del segnale PWM da parte del servomotore.

La routine ha fornito un primo feedback sulla capacità del microcontrollore di generare segnali PWM coerenti con le specifiche richieste dal servomotore e ha permesso di verificare che i segnali riprodotti fossero interpretati correttamente, garantendo così una risposta precisa e conforme alle aspettative.

Questo approccio ha facilitato una verifica preliminare dell'accuratezza e della reattività del servomotore, fornendo indicazioni utili per eventuali regolazioni, come la riduzione dell'angolo di sterzata.

È stato necessario ridurre l'angolo di sterzata poiché, nonostante la demoltiplicazione meccanica tra il servomotore e le ruote, il movimento completo delle ruote tendeva a bloccarle nel telaio della macchina, causando un eccessivo sforzo sul servomotore e rischiando danni al componente. Dopo alcuni test, è stato rilevato che il massimo angolo di sterzata rispetto alla posizione centrale di riposo era di circa 40 gradi, sia a destra che a sinistra.

Conclusi i test sul servomotore, si è passati ai test sull'ESC per valutare il controllo della velocità del motore e far muovere l'intero veicolo.

```arduino
#include <Servo.h>
Servo sterzo;
int sel=0;
int i=90;
void setup() {
  Serial.begin(9600);
  sterzo.attach(9);
  sterzo.write(i);
  Serial.println("accensione");
}

void loop() {
sel=Serial.read();
switch(sel){

  case(49)://5 degrees decrease
    i+=5;
    Serial.println(i);
    sterzo.write(i);
    break;

  case(50): //5 degrees increase
    i-=5;
    Serial.println(i);
    sterzo.write(i);
    break;
  }
}
```

Figura 8: Routine di test del servomotore

2.4.2 Test dell'ESC (marcia avanti)

Successivamente alla conclusione dei test sul servomotore, si è passati a testare l'ESC per verificare il funzionamento del motore e la capacità di controllarlo.

Analogamente ai test sul servomotore, è stata implementata una routine ciclica che aumentava progressivamente la velocità del motore. Per testare la marcia avanti, la routine partiva dalla posizione di riposo e, in quattro step, raggiungeva il 20% della potenza totale, per poi ritornare alla posizione iniziale seguendo gli stessi step. Questo test ha permesso di verificare il funzionamento del motore e di comprendere come comandarlo e con quanta potenza.

Considerando che la macchina, se comandata al 100% della sua potenza, è in grado di superare i 50 km/h, durante i test è stato necessario prestare particolare attenzione per evitare incidenti e prevenire danni sia a persone che all'ambiente circostante.

La gestione della potenza e la sicurezza sono stati quindi aspetti cruciali, richiedendo un controllo preciso e una valutazione continua delle condizioni di test per garantire operazioni sicure durante tutte le fasi di sperimentazione.

2.4.3 Test dell'ESC (marcia indietro)

Analogamente a quanto effettuato per la marcia avanti, è stata eseguita una prova per la retromarcia, utilizzando la stessa routine ciclica di incremento graduale della potenza.

Per ottenere questo risultato, è stato necessario invertire i valori di comando del segnale PWM. Partendo dalla posizione di riposo, il segnale PWM è stato diminuito progressivamente in vari step, consentendo all'ESC di invertire la direzione di rotazione del motore.

2.4.4 Test dell'ESC (marcia avanti e marcia indietro)

In seguito, è stata creata una routine, visibile in figura 9, che combinava marcia avanti e retromarcia per osservare il comportamento complessivo del veicolo. Da questo test è emerso un comportamento particolare: mentre la marcia avanti funzionava correttamente e senza impedimenti, indipendentemente dal comando precedente, la retromarcia richiedeva una specifica sequenza per essere attivata.

Infatti, a veicolo fermo, la retromarcia veniva inserita regolarmente e la macchina si muoveva indietro. Tuttavia, se il veicolo era in movimento in avanti, l'attivazione del comando di retromarcia non invertiva immediatamente il senso di marcia, ma frenava progressivamente la macchina fino a fermarla.

Questo comportamento suggerisce che, probabilmente per ragioni di sicurezza e per prevenire danni meccanici, la retromarcia non può essere attivata finché il veicolo non è completamente fermo. Pertanto, per inserire correttamente la retromarcia, è necessario attivare il comando due volte: la prima per arrestare il veicolo e la seconda per invertire effettivamente il senso di marcia.

Questo test ha fornito una comprensione più approfondita del sistema di controllo del motore, evidenziando l'importanza delle misure di sicurezza integrate nell'ESC e la necessità di tener conto di queste dinamiche durante l'utilizzo del veicolo.

```
#include <Servo.h>
Servo acc;
int sel=0;
int j=90;
void setup() {
  Serial.begin(9600);
  acc.attach(10);
  acc.write(j);
  Serial.println("accensione");
}

void loop() {
sel=Serial.read();
switch(sel){

  case(51)://1
    j+=5;
    Serial.println(j);
    acc.write(j);
    break;

  case(52):
    j-=5;
    Serial.println(j);
    acc.write(j);
    break;
  }
}
```

Figura 9: Routine di test dell'ESC

2.5 Segnale PWM e libreria Servo

Un segnale PWM (Pulse Width Modulation) è una forma di modulazione digitale in cui la durata di un impulso viene variata per rappresentare l'informazione trasportata dal segnale. In pratica, un segnale PWM alterna tra uno stato alto (ON) e uno stato basso (OFF) a una frequenza fissa, con la durata del periodo in cui il segnale è alto, chiamata "larghezza dell'impulso", che determina l'intensità del segnale. La proporzione di tempo in cui il segnale è alto rispetto al tempo totale di un ciclo è nota come "duty cycle" e viene espressa in percentuale.

Nella progettazione del sistema di controllo è stata scelta la libreria Servo di Arduino per generare segnali PWM essenziali per il comando sia del servomotore che dell'ESC.

La libreria Servo è stata selezionata per la sua semplicità d'uso e per la sua capacità di produrre segnali PWM precisi e facilmente configurabili, inoltre è una libreria ben collaudata e supportata in quanto viene direttamente fornita da Arduino e ciò ci dà maggior sicurezza.

Essa consente di specificare direttamente l'angolo desiderato per il servomotore o la velocità per l'ESC, traducendo tali comandi in segnali PWM con impulsi variabili tra 1 MS e 2 MS come visibile in figura 10. Questi impulsi determinano la posizione angolare del servomotore e la potenza erogata dall'ESC. Nel nostro caso, la libreria Servo imposta una frequenza di 50 Hz per i segnali PWM, differente dai 100 Hz utilizzati nel sistema originale del veicolo.

La gestione dei segnali PWM è realizzata tramite timer interni dell'Arduino, che garantiscono un ciclo di lavoro preciso e una sincronizzazione accurata degli impulsi. Gli Arduino Nano, come molti altri modelli di Arduino, utilizzano timer hardware per generare segnali PWM.

Questi timer funzionano con alta precisione e sono programmati per emettere segnali a intervalli regolari. In particolare, la libreria Servo sfrutta i timer dell'Arduino per ottenere un'accuratezza di temporizzazione che riduce al minimo il rischio di alterazioni o variazioni nei segnali PWM.

La gestione accurata dei timer è fondamentale per mantenere una frequenza di PWM stabile e per garantire che il servomotore e l'ESC ricevano segnali coerenti e di alta qualità, inoltre ciò evita comportamenti imprevedibili della macchina dovuti a comandi PWM errati che possono provocare danni ed incidenti.

Nonostante la libreria Servo gestisca segnali PWM a una frequenza di 50 Hz, invece dei 100 Hz previsti, il servomotore e l'ESC hanno interpretato i segnali correttamente, dimostrando una certa tolleranza verso la variazione di frequenza. Tuttavia, è importante notare che la riduzione della frequenza da 100 Hz a 50 Hz ha comportato una leggera diminuzione nella reattività del sistema.

Questo si traduce in una risposta leggermente più lenta ai comandi, che potrebbe influenzare la precisione e la fluidità delle manovre del veicolo. La riduzione della frequenza può anche influire sulla capacità del sistema di rispondere a cambiamenti rapidi nei comandi, risultando in una maggiore latenza durante le operazioni di controllo, ma nel nostro caso questa differenza di frequenza non ha portato alcun cambiamento nella risposta del veicolo a cambiamenti di direzione o accelerazioni rapide.

In aggiunta, la frequenza del clock del microcontrollore può variare leggermente a causa di tolleranze di fabbricazione e condizioni operative, il che può influenzare la frequenza del segnale PWM generato. Variazioni significative nella frequenza del segnale possono portare a movimenti imprevisti del servomotore o a un comportamento non ottimale dell'ESC.

La scelta della libreria Servo si è rivelata efficace, sebbene con una leggera perdita di reattività, i risultati confermano la validità dell'approccio adottato e offrono spunti per ulteriori ottimizzazioni future.

Tra queste, la regolazione della frequenza PWM e una gestione più fine delle variazioni di clock possono migliorare ulteriormente le prestazioni e la stabilità del sistema di controllo del veicolo, assicurando un controllo più preciso e affidabile delle operazioni del servomotore e dell'ESC. Questi miglioramenti non solo ottimizzano le performance, ma contribuiscono anche a una maggiore affidabilità e durata del sistema in ambienti di utilizzo diversi, garantendo che il veicolo radiocomandato operi in modo efficiente e consistente anche in condizioni estreme.

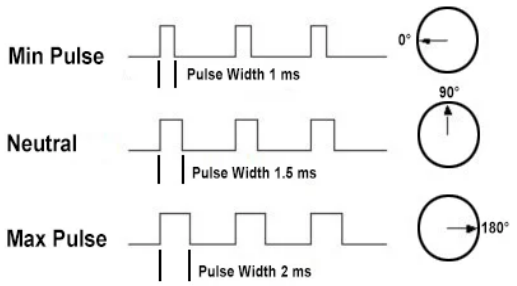

Figura 10: Traduzione del valore di input in segnale PWM

2.6 Framework Flutter

In questo paragrafo prendiamo in considerazione il framework Flutter e l'app con esso sviluppata.

Flutter è un framework open source di Google utilizzato per sviluppare applicazioni multipiattaforma da una sola codebase, il linguaggio utilizzato è Dart sviluppato da Google.

Inizialmente il framework era limitato alla creazione di interfacce native iOS, Android, Linux, Windows e MacOS ma successivamente è stato introdotto anche il supporto allo sviluppo di applicazioni web per siti statici.

L'architettura di Flutter è suddivisa in quattro componenti principali, ognuno di questi componenti svolge un ruolo cruciale nel funzionamento e nella capacità di Flutter di creare applicazioni cross-platform.

- Dart Platform: il compilatore del linguaggio Dart utilizza due diverse piattaforme per lo sviluppo delle due diverse interfacce. Una piattaforma Dart Native usata per lo sviluppo su dispositivi (come smartphone, desktop, server, ecc.…)

  L'altra piattaforma invece è la piattaforma Dart Web per lo sviluppo di interfacce web con una compilazione per lo sviluppo e una per la produzione

- Flutter Engine: Il Flutter Engine è il nucleo di Flutter, responsabile della gestione del rendering, del layout, del testo, delle animazioni, e dell'input dell'utente. Basato su Skia, il motore grafico open-

source utilizzato da Google Chrome, il Flutter Engine permette di ottenere un rendering veloce e fluido dell'interfaccia utente, indipendentemente dal dispositivo o dalla piattaforma.

Inoltre, il Flutter Engine supporta la gestione della grafica 2D, delle animazioni e del testo, offrendo una vasta gamma di strumenti per creare interfacce utente dinamiche e interattive.

- Foundation Library: Questa libreria include una serie di funzioni essenziali, come la gestione dello stato, delle animazioni e delle transizioni, che facilitano la creazione di applicazioni complesse. Ad esempio, la gestione dello stato è fondamentale per costruire interfacce utente reattive, in cui i cambiamenti nello stato dell'applicazione si riflettono immediatamente nell'interfaccia utente.
- Widget: I widget sono il cuore dell'interfaccia utente di Flutter. Ogni elemento, dai layout più complessi ai singoli pulsanti, è un widget. Flutter adotta un approccio dichiarativo, in cui l'interfaccia utente è costruita come una gerarchia di widget. Questa struttura gerarchica permette di gestire facilmente la composizione e la disposizione degli elementi dell'interfaccia. I widget possono essere di due tipi principali: StatelessWidget, che rappresenta componenti immutabili dell'interfaccia utente, e StatefulWidget, che possono cambiare stato durante l'esecuzione dell'app.

L'applicazione utilizza un plugin di Flutter di nome quickblue in grado di:

- Ricercare dispositivi BLE nei dintorni
- Connettersi con i dispositivi rilevati
- Scoprire i servizi che vengono offerti dai dispositivi
- Trasferire i dati fra i due dispositivi

Utilizzando quindi queste funzionalità, è stato possibile integrare in un'unica applicazione il joystick per comandare la macchina e l'interfaccia relativa al collegamento e alla comunicazione dei comandi via BLE.

Questa integrazione non solo dimostra la potenza di Flutter nell'interfacciarsi con hardware esterno, ma evidenzia anche la sua capacità di gestire comunicazioni complesse e di fornire un'esperienza utente coerente e senza interruzioni.

La user interface è composta da una serie di pagine, fra cui l'utente deve navigare per collegarsi tramite BLE e comandare la macchina.

Questa interfaccia è stata progettata per essere intuitiva e facile da usare, riducendo al minimo il numero di passaggi necessari per la connessione e l'invio dei comandi, migliorando così l'efficienza e l'esperienza utente complessiva.

2.7 Applicazione sviluppata con Flutter

La pagina di comando dell'app Flutter è progettata per consentire agli utenti di controllare la macchina utilizzando due joystick per comandare acceleratore e sterzo e due slider per regolare l'offset sullo sterzo e la potenza totale.

Di seguito sono descritte ed analizzate le varie parti di codice che compongono i widget implementati, che lavorano insieme per fornire un'interfaccia utente interattiva e come il loro utilizzo influenza l'andamento della macchina.

2.7.1 Struttura della pagina

Scaffold: Il widget Scaffold fornisce una struttura di base per la pagina, includendo elementi essenziali come l'AppBar, il corpo della pagina e la gestione del colore di sfondo. Questo widget è fondamentale per costruire una struttura coerente e reattiva dell'app, garantendo che il contenuto si adatti correttamente a diverse dimensioni dello schermo e risoluzioni.

AppBar: L'AppBar visualizza il titolo della pagina e può includere altre azioni, come pulsanti o icone, per migliorare l'usabilità. In questo caso, è impostato su "Measurify Car", fornendo agli utenti un chiaro riferimento all'applicazione specifica e al suo scopo.

SafeArea: Il widget SafeArea è utilizzato per garantire che il contenuto non venga sovrapposto da aree di sistema come la barra di stato o le bande di navigazione. Questo widget è essenziale per evitare problemi di visualizzazione sui dispositivi con diverse configurazioni hardware.

Stack: Il widget Stack consente di sovrapporre i widget in modo flessibile. In questa applicazione, viene utilizzato per posizionare i joystick e altri controlli su uno sfondo, permettendo una disposizione personalizzata degli elementi senza compromettere la funzionalità dell'interfaccia utente.

Align: Il widget Align è impiegato per posizionare i joystick all'interno dello Stack. Grazie a questo widget, i joystick per lo sterzo e l'acceleratore possono essere posizionati in modo preciso e intuitivo, migliorando l'esperienza dell'utente nella gestione del veicolo.

Joystick: Un widget personalizzato per l'interazione dell'utente. Viene usato per controllare lo sterzo e l'acceleratore del veicolo. Il movimento del joystick per lo sterzo è configurato in modalità orizzontale, mentre quello per l'acceleratore è in modalità verticale.

Positioned: Il widget Positioned viene utilizzato per posizionare i pulsanti e i controlli in posizioni specifiche all'interno dello Stack. Due widget Positioned sono impiegati per i pulsanti di inversione del comando e per i cursori di configurazione, permettendo una disposizione ordinata e funzionale degli elementi dell'interfaccia utente.

La foto in figura 11 qua sotto riporta una parte di codice Dart relativa ai due joystick.

```dart
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Color.fromARGB(255, 180, 87, 21),
    appBar: AppBar(
      title: const Text('Measurify Car'),
    ), // AppBar
    body: SafeArea(
      child: Stack(
        children: [
          Container(
            color: Color.fromARGB(255, 255, 255, 255),
          ), // Container
          Align(
              //joystick steering
              alignment: Alignment(stateCommand * (0.8), 0.5),
              child: Joystick(
                mode: JoystickMode.horizontal,
                listener: (details) {
                  setState(() {
                    steering = idleValueSteering +
                        ((details.x * rangeValueSteering).round());
                  });
                },
              )), // Joystick // Align
          Align(
              //joystick throttle
              alignment: Alignment(stateCommand * (-0.8), 0.5),
              child: Joystick(
                mode: JoystickMode.vertical,
                listener: (details) {
                  setState(() {
                    throttle = idleValueThrottle -
                        ((details.y * rangeValueThrottle * powerSliderValue).round());
                  });
                },
              )), // Joystick // Align
```

Figura 11: codice Dart

2.7.2 Funzionamento dei widget

Joystick per sterzo e acceleratore: I joystick rilevano i movimenti dell'utente e aggiornano i valori di sterzo e acceleratore.

Sterzo: con JoystickMode.horizontal si configura il joystick per controllare lo sterzo. I dati di movimento vengono utilizzati per calcolare il valore di sterzo, che viene aggiornato in base alla posizione del joystick.

Acceleratore: con JoystickMode.vertical si configura il joystick per controllare l'acceleratore. I dati di movimento sono utilizzati per calcolare il valore di acceleratore, che varia a seconda della posizione del joystick.

Slider di Potenza: consente all'utente di regolare la potenza totale della macchina, è utile in quanto evita movimenti bruschi dovuti ad accelerazioni repentine ed evita ad esempio di colpire oggetti in luoghi stretti o con poco spazio di manovra. Inoltre, il valore numerico sopra lo slider indica la percentuale di potenza a cui è settata la macchina.

Slider dell'offset di Sterzo: Lo slider dell'offset di sterzo permette all'utente di correggere eventuali deviazioni nel movimento del veicolo.

Questa funzionalità è stata implementata per replicare l'esperienza del telecomando originale e per migliorare il controllo. È particolarmente utile quando la macchina tende a deviare da un lato e richiede una regolazione fine per mantenere un movimento dritto.

Pulsante di Inversione Comando: Il pulsante di inversione comando consente di scambiare la posizione dei comandi dell'acceleratore e dello sterzo.

Questa opzione è utile per adattarsi a diverse preferenze di guida e per facilitare il controllo in scenari di guida variabili. La possibilità di invertire i comandi offre una maggiore flessibilità e personalizzazione dell'esperienza di guida.

Si può vedere nella figura 12 il risultato finale dell'interfaccia grafica.

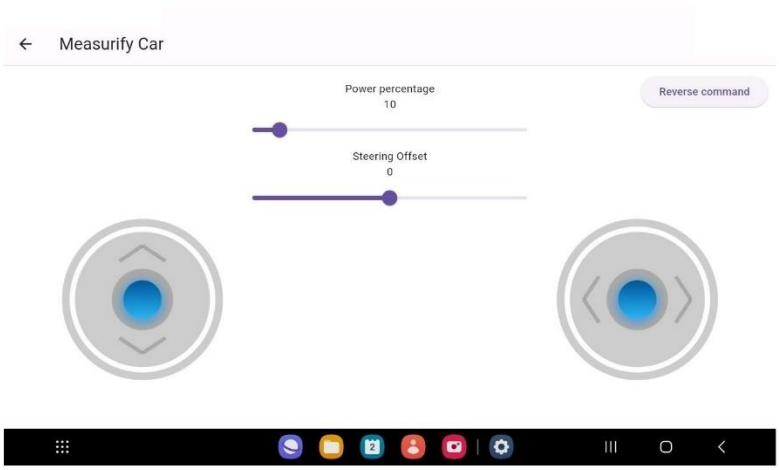

Figura 12: Interfaccia grafica

2.7.3 Gestione dello stato

La pagina utilizza variabili di stato per gestire i valori del joystick e degli slider. Ecco come cambia lo stato in risposta alle azioni dell'utente:

steering e throttle: Questi valori rappresentano lo stato attuale dello sterzo e dell'acceleratore. Sono aggiornati in tempo reale ogni volta che l'utente muove i joystick, i valori sono poi inviati periodicamente tramite BLE, garantendo che le modifiche siano riflesse immediatamente nel comportamento del veicolo.

powerSliderValue e steeringOffsetSliderValue: Questi valori rappresentano le posizioni degli slider di potenza e offset di sterzo. Ogni volta che l'utente modifica questi slider, i valori sono aggiornati e utilizzati per regolare i comandi inviati al dispositivo.

stateCommand: è una variabile utilizzata per invertire la posizione dei comandi nell'applicazione; infatti, cambiandola si può notare che il comando dello sterzo viene posizionato al posto del comando dell'acceleratore e viceversa.

2.8 Comunicazione BLE

2.8.1 Il bluetooth low energy

Il Bluetooth Low Energy (BLE) è una tecnologia di comunicazione wireless progettata per operare con un basso dispendio di energie, ideata per dispositivi che richiedono una durata della batteria prolungata e una comunicazione non continua.

BLE, una parte del Bluetooth 4.0 e successivi, è stato sviluppato per risolvere i limiti delle versioni precedenti del Bluetooth, in particolare per applicazioni a basso consumo come sensori, dispositivi indossabili e IoT (Internet of Things).

Il BLE è stato introdotto per la prima volta nel 2010 come una parte delle specifiche Bluetooth 4.0, conosciute anche come Bluetooth Smart. L'idea alla base di BLE era quella di creare una tecnologia che offrisse un consumo energetico estremamente ridotto, permettendo ai dispositivi a batteria di funzionare per periodi molto più lunghi rispetto al Bluetooth tradizionale.

BLE è progettato per essere particolarmente adatto a scenari di comunicazione intermittente e a bassa velocità di trasferimento dei dati, dove il dispositivo può restare in uno stato di sospensione la maggior parte del tempo e riattivarsi solo per brevi periodi per scambi di dati.

Opera su frequenze di 2,4 GHz, come il Bluetooth tradizionale, ma utilizza un diverso schema di comunicazione per ridurre il consumo energetico. La tecnologia si basa su un modello di "pubblicazione" e "sottoscrizione" per la comunicazione, dove un dispositivo (chiamato "centrale") invia pacchetti di dati che possono essere ricevuti da uno o più dispositivi "periferici".

Il BLE utilizza una serie di servizi e caratteristiche definiti da standard che consentono la trasmissione di dati in modo flessibile e personalizzabile.

2.8.2 Connessione di un dispositivo

Ricerca e Connessione ai Dispositivi BLE:

Per stabilire una connessione BLE tra un'applicazione Flutter e un Arduino Nano 33 BLE Sense, è necessario utilizzare una libreria che gestisca la comunicazione BLE. In questo caso, viene utilizzata la libreria quickblue che fornisce una serie di funzionalità per gestire la scoperta e la connessione ai dispositivi BLE.

Scoperta dei Dispositivi:

La ricerca dei dispositivi BLE nelle vicinanze viene eseguita utilizzando le capacità della libreria di scansionare le reti BLE disponibili. Durante questa fase, l'app Flutter invia richieste di scansione che identificano i dispositivi BLE nel raggio d'azione. Ogni dispositivo BLE broadcast ha un identificatore univoco e, eventualmente, informazioni aggiuntive come il nome del dispositivo e la potenza del segnale.

Connessione al Dispositivo:

Una volta identificato il dispositivo Arduino Nano 33 BLE Sense tra quelli disponibili, viene stabilita una connessione con esso. Questa fase include la verifica dell'autenticità del dispositivo e la configurazione delle caratteristiche di comunicazione.

Si può vedere nella foto sotto il processo di connessione tra i due dispositivi.

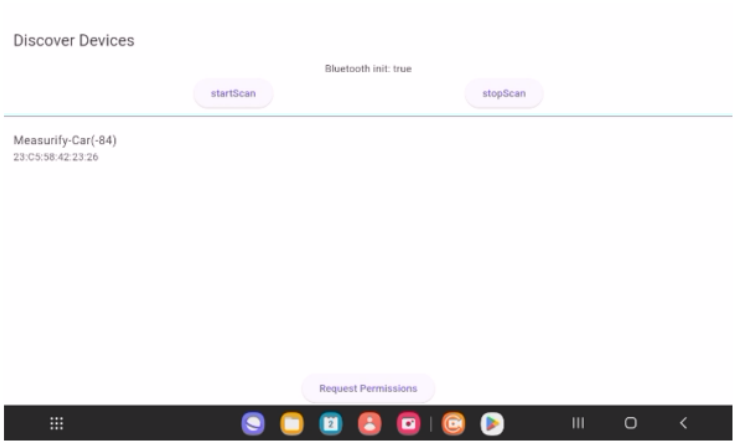

Figura 13: Connessione tra due dispositivi

Scoperta dei Servizi e delle Caratteristiche:

Dopo la connessione, l'applicazione esplora i servizi e le caratteristiche del dispositivo BLE. Questi servizi includono gruppi di caratteristiche che permettono la lettura o la scrittura di dati. L'Arduino Nano 33 BLE Sense espone servizi specifici per il controllo del veicolo, che l'app Flutter utilizza per inviare e ricevere comandi.

Una volta stabilita la connessione e scoperti i servizi, l'app Flutter utilizza BLE per inviare comandi e ricevere dati dall'Arduino Nano 33 BLE Sense. Questo processo si suddivide in diverse fasi:

Invio dei Comandi: i comandi inviati tramite l'interfaccia utente dell'app Flutter vengono convertiti in pacchetti di dati BLE.

Questi pacchetti contengono istruzioni per controllare il veicolo, come l'input per lo sterzo o la regolazione della potenza. La libreria quickblue gestisce la trasmissione di questi pacchetti al dispositivo Arduino, che li riceve attraverso le sue caratteristiche BLE.

Decodifica e Controllo: l'Arduino Nano 33 BLE Sense riceve i pacchetti di dati e li decodifica per estrarre le informazioni necessarie. I comandi vengono tradotti in segnali PWM per il controllo dell'ESC e del servomotore, permettendo al veicolo di rispondere ai comandi in modo preciso e tempestivo.

Vantaggi: l'integrazione del BLE con l'Arduino Nano 33 BLE Sense e l'app Flutter offre numerosi vantaggi. La scelta del BLE garantisce una comunicazione efficace e a basso consumo energetico, migliorando la durata della batteria del dispositivo.

L'uso della libreria quickblue consente una gestione fluida e precisa della connessione e del trasferimento dei dati, offrendo un'esperienza utente ottimale e un controllo affidabile del veicolo.

2.9 Software di sicurezza

Il sistema di sicurezza software implementato sul microcontrollore è progettato per garantire la sicurezza operativa del veicolo attraverso una serie di meccanismi di protezione e controllo.

Uno dei principali strumenti di sicurezza è il monitoraggio del tempo di inattività della connessione BLE, se il veicolo non riceve comandi tramite BLE per un periodo che supera il valore predefinito di idle_period, il sistema entra automaticamente in uno stato di inattività.

In questo stato, il servomotore e l'ESC vengono riportati alle loro posizioni di sicurezza predefinite, ossia gli angoli di sterzo e accelerazione vengono impostati ai valori di idle (IDLE_STEERING e IDLE_THROTTLE). Questo processo è essenziale per evitare che il veicolo continui a muoversi senza controllo in assenza di input validi, riducendo al minimo il rischio di comportamenti imprevisti o incidenti.

Un ulteriore livello di protezione è fornito dal contatore di errori. Ogni volta che il sistema entra nello stato di inattività a causa della mancanza di comandi, il contatore di errori (error) viene incrementato. Se il contatore supera una soglia critica (MAX_ERROR), che indica che il sistema ha subito un numero eccessivo di timeout, il microcontrollore disconnette automaticamente la connessione BLE e riavvia il processo di pubblicità per cercare nuove connessioni. Questa misura garantisce che il veicolo non rimanga in uno stato di errore prolungato, prevenendo potenziali danni e garantendo che il dispositivo rimanga operabile e sicuro.

La gestione degli errori è ulteriormente raffinata dal fatto che, alla ricezione di un comando valido, il contatore di errori viene azzerato (error = 0), e il timer di inattività (idle_previous) viene aggiornato. Questo assicura che il sistema continui a operare correttamente senza subire interruzioni o malfunzionamenti a causa di errori di comunicazione precedenti.

Il software include quindi una logica di "failsafe" che garantisce il controllo continuo del veicolo in condizioni di errore. Se si verificano errori ripetuti o prolungati nella comunicazione, il sistema si spegne automaticamente. Questa funzionalità di sicurezza è fondamentale per mantenere l'affidabilità e l'integrità del sistema in scenari operativi complessi o in caso di problemi imprevisti.

Infine, anche se disabilitati di default, i messaggi di debug sono disponibili per fornire un feedback dettagliato durante lo sviluppo e la diagnostica del sistema. Questi messaggi permettono di monitorare in tempo reale i valori dei comandi e lo stato del veicolo, offrendo informazioni preziose per il debugging e l'ottimizzazione del sistema, la combinazione di questi meccanismi di sicurezza assicura che il veicolo operi in modo sicuro.

3 Sperimentazione e risultati

Il processo di sperimentazione del progetto ha seguito un approccio modulare, che ha permesso di testare e ottimizzare le singole componenti hardware e software in parallelo con lo sviluppo. Questo metodo ha garantito un'affidabile integrazione dei vari moduli, riducendo il rischio di problemi complessi durante la fase finale di assemblaggio. La sperimentazione è stata fondamentale per identificare e risolvere problemi specifici, ottimizzare le performance e garantire la robustezza del sistema complessivo.

Durante le fasi iniziali dello sviluppo, il controllo dello sterzo e dell'acceleratore sono stati testati separatamente per assicurare che ciascun componente funzionasse correttamente e rispondesse ai comandi in modo preciso. I servomotori e l'ESC sono stati sottoposti a una serie di test, che includevano la verifica della linearità dei movimenti e della sensibilità delle risposte ai comandi. I risultati hanno dimostrato che entrambi i componenti erano in grado di raggiungere e mantenere le posizioni desiderate senza errori significativi. Tuttavia, sono stati identificati alcuni problemi minori nella risposta a comandi estremi, che sono stati prontamente risolti attraverso regolazioni nei parametri di controllo.

La sperimentazione ha anche messo in luce l'efficacia del sistema di sicurezza software implementato. I test hanno confermato che la logica di "failsafe" rispondeva adeguatamente in caso di errori di comunicazione prolungati, spegnendo automaticamente il sistema per evitare danni.

Nel complesso, i test hanno confermato che l'approccio modulare e il collaudo continuo durante lo sviluppo hanno contribuito significativamente al successo del progetto, consentendo di affrontare e risolvere problemi in modo mirato. I dati raccolti hanno mostrato che il sistema di controllo rispondeva in modo adeguato ai comandi dell'utente e che le modifiche agli slider e ai joystick erano riflesse in tempo reale nel comportamento del veicolo. La stabilità e l'affidabilità del sistema BLE si sono dimostrate soddisfacenti in tutte le condizioni operative.

Le modifiche e le ottimizzazioni future saranno orientate a migliorare la stabilità complessiva e a garantire una performance affidabile in tutte le condizioni operative. I piani includono l'implementazione di algoritmi avanzati per la gestione degli errori, l'espansione dei test in ambienti variabili e l'ottimizzazione della comunicazione BLE per affrontare condizioni di interferenze più severe. La sperimentazione continua e l'adattamento alle nuove sfide rappresentano una parte fondamentale del processo di sviluppo, assicurando che il sistema possa evolversi e migliorare continuamente per soddisfare le esigenze degli utenti e le aspettative di prestazioni.

Le prossime fasi del progetto si concentreranno su ulteriori miglioramenti della stabilità e dell'affidabilità del sistema, per assicurare prestazioni ottimali in una varietà di condizioni operative. Tra le modifiche pianificate vi è l'implementazione di algoritmi avanzati per la gestione degli errori, progettati per garantire che il sistema possa affrontare e recuperare da malfunzionamenti imprevisti in modo più efficace. Questo include l'adozione di strategie più robuste per il rilevamento e la correzione degli errori, nonché il miglioramento della verifica dell'integrità dei dati trasmessi.

Inoltre, si prevede di ampliare il campo dei test, includendo una gamma più vasta di ambienti operativi per valutare il comportamento del sistema sotto diverse condizioni. Questo approccio aiuterà a identificare e risolvere potenziali problemi che potrebbero non emergere in scenari di test più limitati, assicurando una maggiore affidabilità e robustezza del sistema in situazioni reali.

Un altro obiettivo importante sarà l'ottimizzazione della comunicazione BLE. Sebbene il sistema si sia dimostrato generalmente stabile, ci sono margini di miglioramento nella gestione delle interferenze e nella stabilità della connessione in ambienti con alta densità di segnali. L'ottimizzazione della comunicazione BLE includerà l'adozione di tecniche avanzate per migliorare la resilienza del segnale e ridurre il rischio di disconnessioni o perdite di dati.

Il continuo processo di sperimentazione e l'adattamento alle sfide emergenti rappresentano un elemento cruciale dello sviluppo del progetto. Questo approccio dinamico assicurerà che il sistema non solo soddisfi le aspettative attuali, ma si adatti e migliori costantemente per rispondere alle esigenze future degli utenti e alle evoluzioni del contesto tecnologico.

4 Contributo personale e considerazioni conclusive

Il progetto di tesi si è sviluppato con l'obiettivo di creare un sistema di controllo avanzato per un veicolo radiocomandato utilizzando la comunicazione Bluetooth Low Energy (BLE) e un'applicazione mobile sviluppata in Flutter. Il progetto ha rappresentato una sfida interessante nella realizzazione di un'interfaccia utente intuitiva e nella gestione della comunicazione wireless con un microcontrollore. Un possibile sviluppo futuro per questo sistema potrebbe essere l'integrazione di funzionalità di analisi dei dati per monitorare l'ambiente circostante in tempo reale fornendo dati di ogni genere.

Inoltre, si potrebbero implementare algoritmi di apprendimento automatico per ottimizzare i comandi inviati al veicolo, migliorando la precisione del controllo e adattando la risposta del sistema alle diverse condizioni di guida e di realizzare successivamente un veicolo a completa guida autonoma.

Un ulteriore passo avanti potrebbe includere lo sviluppo di un'applicazione desktop o web che permetta agli utenti di visualizzare e analizzare in modo più approfondito i dati raccolti, offrendo report e grafici sull'andamento delle sessioni di guida.

Gli strumenti principali utilizzati per questo progetto sono stati il linguaggio di programmazione C++ per la programmazione dell'Arduino e Dart per lo sviluppo dell'app Flutter. Il mio contributo personale ha riguardato la decodifica dei segnali di comando, la progettazione e l'implementazione dell'interfaccia utente dell'app, la configurazione della comunicazione BLE, la gestione dei dati inviati al microcontrollore ed il relativo comando dei motori mediante quest'ultimo.

5 Riferimenti bibliografici

[1] Arduino Nano 33 BLE Sense Rev2: https://docs.arduino.cc/hardware/nano-33-ble-sense-rev2

[2] Arduino IDE: https://www.arduino.cc/en/software

[3] Arduino BLE (libreria): https://www.arduino.cc/reference/en/libraries/arduinoble/

[4] Arduino servo (libreria): https://www.arduino.cc/reference/en/libraries/servo/

[5] Flutter: https://it.wikipedia.org/wiki/Flutter_(software)

[6] Traxxas: https://traxxas.com/products/models/electric/58276-74-slash-vxl

6 Ringraziamenti

Un ringraziamento al professore Riccardo Berta ed al dottor Matteo Fresta per avermi aiutato nello svolgimento di questa tesi. Grazie alla mia famiglia, alla mia ragazza ed ai miei amici per essermi stati vicini e avermi sostenuto durante il percorso universitario.