# Introduzione

Informazioni sul progetto

Il progetto Tankino è stato realizzato da Adriano Chiriacò, Nicholas Pigni, Mosè Ferrazini e Michele Tomyslak alla Scuola Arti e Mestieri a Trevano durante la lezione del modulo 306.

I docenti responsabili sono:

- Luca Muggiasca (luca.muggiasca@edu.ti.ch)

- Geo Petrini (geo.petrini@edu.ti.ch)

La data d'inizio del progetto è il 17.01.2020 mentre la fine è prevista per il 08.05.2020.

## Abstract

*In questo documento si possono trovare tutte le informazioni che riguardano lo sviluppo del progetto di Tankino. Più nel dettaglio si possono trovare informazioni su come abbiamo gestito la progettazione del nostro prodotto, i mezzi utilizzati, i requisiti da rispettare, come è avvenuta l’effettiva realizzazione, il design dell’applicazione, i vari protocolli di comunicazione e i test eseguiti.*

## Scopo

Lo scopo del progetto Tankino è di realizzare un mini-carrarmato che sarà pilotabile grazie a un’applicazione disponibile per mobile. Il cuore centrale del nostro veicolo è un arduino che servirà a controllare tutti i motori. Il controllo da remoto viene effettuato stabilendo una connessione bluetooth.

# Analisi

## Analisi e specifica dei requisiti

|  |  |  |  |
| --- | --- | --- | --- |
| Id requisito | Requisito | Priorità | Versione |
| REQ-001 | Avere un’applicazione mobile | 1 | 1.0 |
| REQ-002 | Avere un arduino funzionante | 1 | 1.0 |
| REQ-003 | L’applicazione deve permettere di pilotare il veicolo | 2 | 1.0 |
| REQ-004 | Essere in grado di fare un pairing bluetooth | 2 | 1.0 |
| REQ-005 | Il veicolo deve essere in grado di muoversi avanti e indietro | 2 | 1.0 |
| REQ-006 | Il veicolo deve essere in grado di curvare | 3 | 1.0 |
| REQ-007 | Il veicolo deve avere delle luci di controllo, gestibili da applicazione | 4 | 1.0 |
| REQ-008 | Avere un sito dove scaricare le applicazioni | 3 | 1.0 |
| REQ-009 | Il veicolo deve essere pilotato in tempo reale con la minor latenza possibile (misurata). | 3 | 1.0 |
| REQ-010 | La velocità deve essere variabile | 4 | 1.0 |

## Use case

I casi d’uso rappresentano l’interazione tra i vari attori e le funzionalità del prodotto.

# Pianificazione

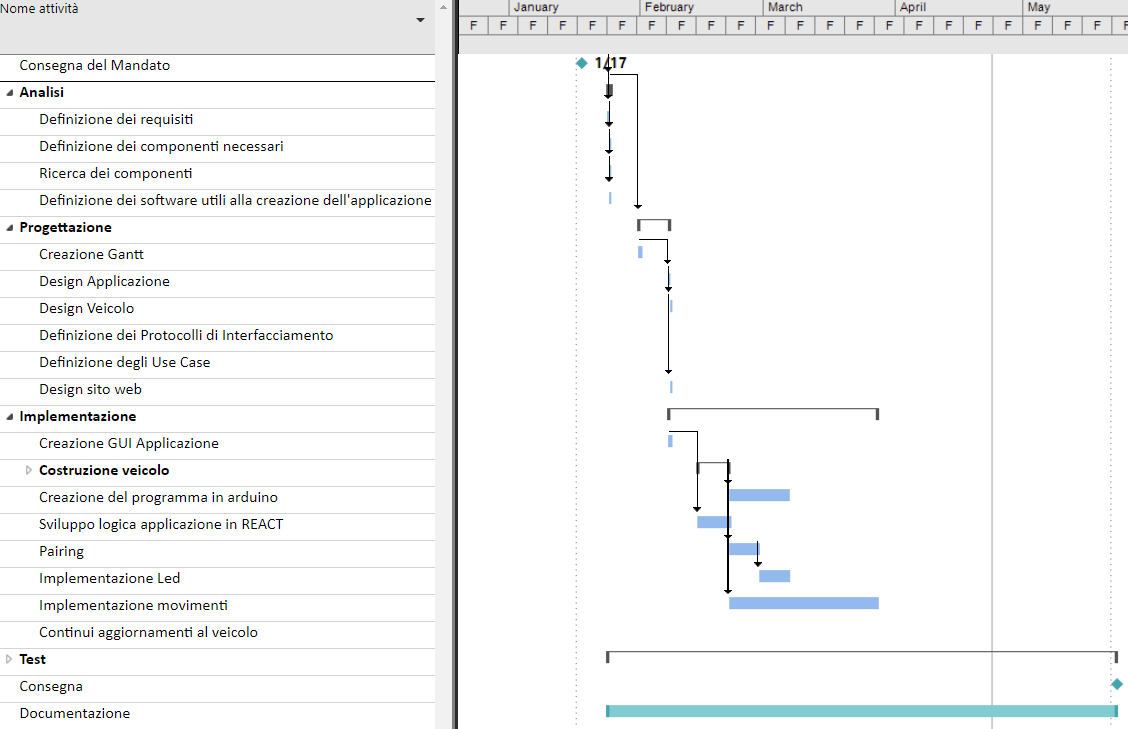
Per fare la pianificazione abbiamo creato un Gantt cercando di inserire all’interno tutte le attività che ci siamo posti, e suddividendocele.

Figura Gantt

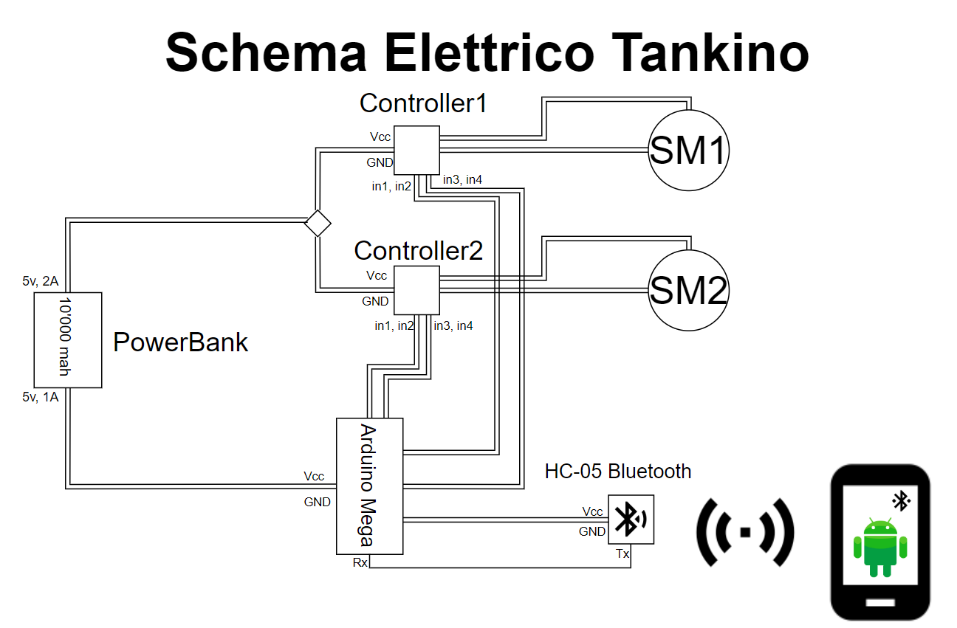
## Analisi dei mezzi

I mezzi da noi utilizzati per quanto riguarda la parte software e quindi la programmazione sono stati Android Studio e Arduino IDE. Per eseguire la creazione del Gantt abbiamo utilizzato un prodotto del pacchetto Office chiamato “Project” e un software libero chiamato “ProjectLibre”.

Mentre per quanto riguarda l’hardware ci siamo provvisti di due step motor “nema 8”, di un powerbank “Aukey” da 10’000mah e due controller per gli step motor “L298N”.

Il programma per controllare il Tankino bisognerà avviarlo su un qualsiasi dispositivo android.

# Progettazione

La progettazione del nostro prodotto è variata tanto nel tempo, ma dopo varie considerazioni siamo arrivati a questa conclusione. La base fisica dal Tankino verrà realizzata in legno. Le ruote, prodotte in plexiglass, saranno collegate tra loro con dei cingoli. I cingoli che andremo ad utilizzare sono originali “lego”. Per quanto riguarda l’alimentazione utilizzeremo una powerbank con due uscite: una da 1v e 1a e una da 1v e 2a. Nella prima ci collegheremo l’arduino mentre la seconda servirà ad alimentare i motori. All’arduino ci sarà anche collegato un ricettore bluetooth che servirà a controllare il Tankino tramite l’applicazione per smartphone. Inoltre, sempre all’arduino, ci saranno collegati i due controller che gli permetteranno di gestire gli step motor. Per spiegare meglio tutta questa parte di elettronica abbiamo deciso di sviluppare uno schema elettrico.

Un’altra cosa a cui abbiamo dovuto pensare è il protocollo di comunicazione tra l’applicazione all’arduino tramite bluetooth. La nostra idea è questa:

**Protocollo di comunicazione**

|  |  |  |
| --- | --- | --- |
| **Messaggio** | **Nome** | **Descrizione** |
| MSB = 0  0XXX XXXX | Cingolo destro | Il cingolo destro si muove in base alla percentuale data (da -100% a +100%)  L’MSB determina che il cingolo che si deve muovere è il destro, il resto è il valore passato che è compreso nel range [-64;63] (oppure [0-127]).  Il motore si muove in avanti se il valore è positivo e indietro se il valore è negativo. |
| MSB = 1  1XXX XXXX | Cingolo sinistro | L’MSB determina che il cingolo che si deve muovere è il sinistro, il resto è il valore passato che è compreso nel range [-64;63] (oppure [0-127]).  Il cingolo si muove in avanti se il valore è positivo e indietro se il valore è negativo. |

## Design

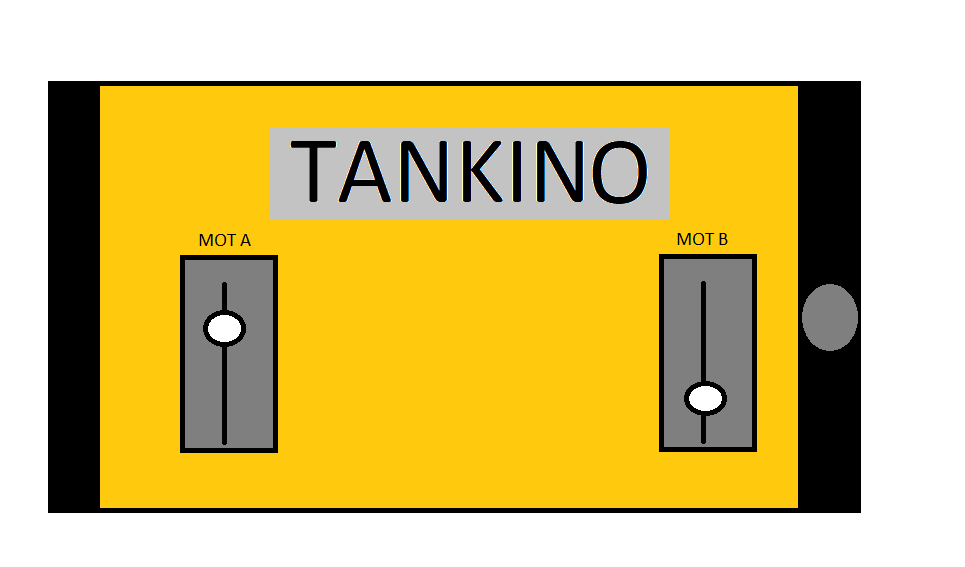
L’interfaccia dell’applicazione per mobile sarà molto semplice, sarà basata su due slider che permetteranno di gestire in modo autonomo i singoli motori. Ovviamente portando lo slider al massimo il motore raggiungerà la velocità massima e portandolo al minimo il motore sarà fermo. Questo è il design per quanto riguarda il design dell’applicazione:

Figura Progetto interfaccia app

Per quanto invece riguarda il design del tankino vero e proprio sarà formato da due tavole di legno poste una sopra l’altra nel quale centralmente c’è un buco per far passare i cavi da sopra a sotto. Le ruote saranno realizzate in plexiglass e con dimensioni uguali tra loro.

# Implementazione

## Modulo HC-05

Per gestire il bluetooth abbiamo utilizzato il modulo HC-05 e per implementarlo ci siamo basati su questo schema:

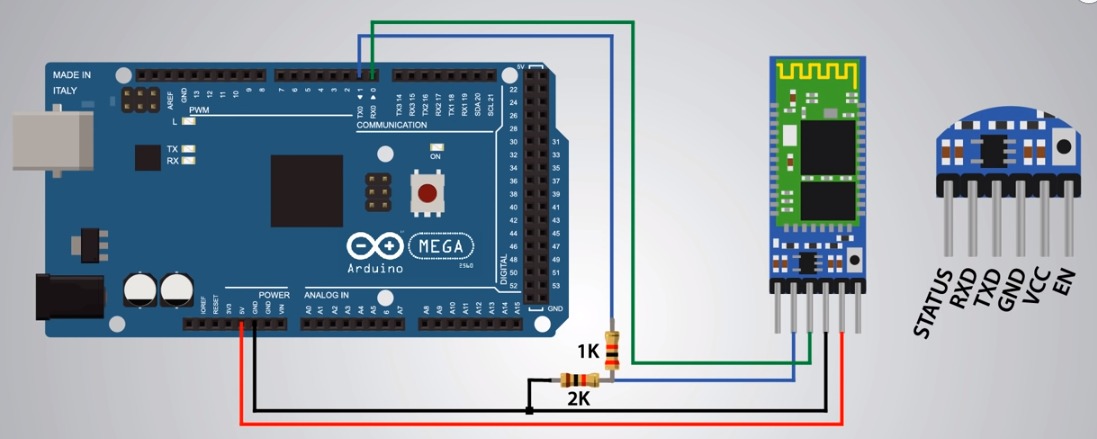


Figura Schema Arduino-HC-05

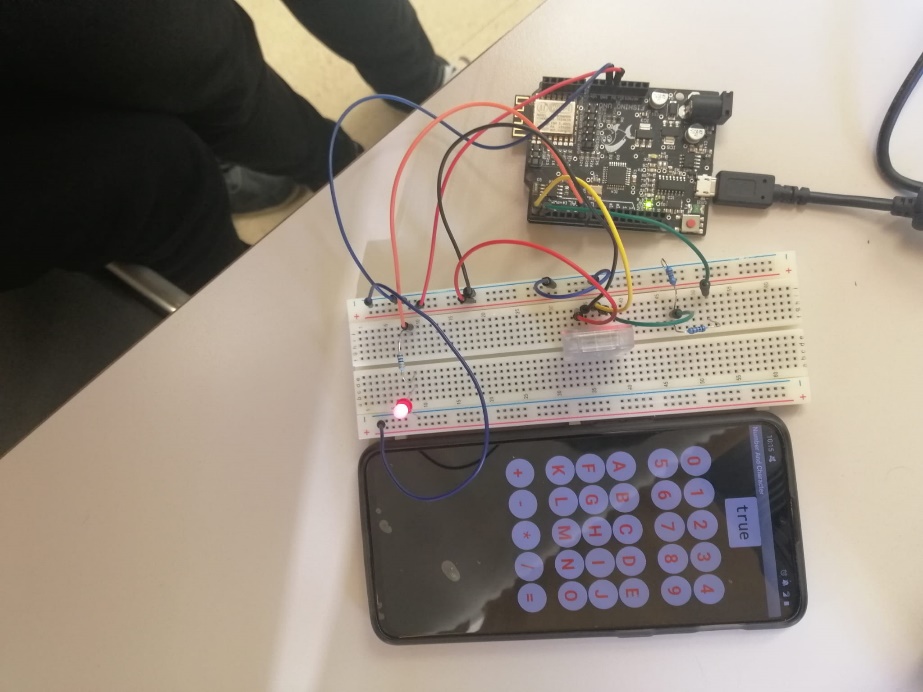
Collegando l’HC-05 all’arduino in questo modo siamo in grado di comunicare con il dispositivo per il controllo remoto. La comunicazione tra i due componenti avviene tramite i pin RX e TX dell’arduino.

Figura Prime prove HC-05

Per fare le prime prove nell’utilizzarlo abbiamo semplicemente acceso e spento un led tramite un’applicazione trovata sul play store.

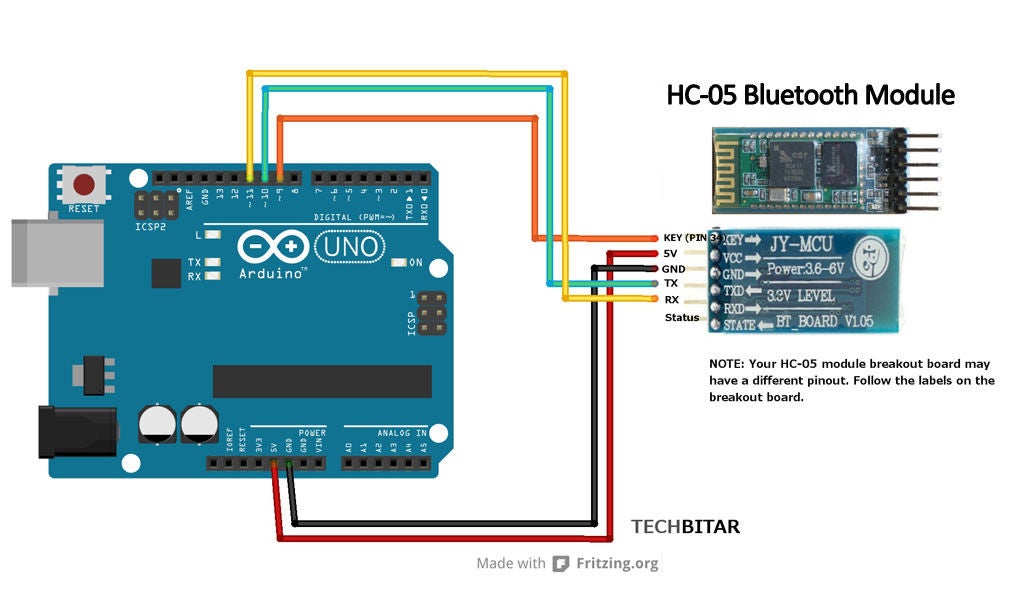
Dopodiché abbiamo rinominato il modulo HC-05 in “Tankino” per renderlo più riconoscibile alla connessione con il dispositivo mobile. Per farlo abbiamo dovuto utilizzare uno schema diverso perché per modificare il nome e le altre impostazioni del modulo bisogna attivare la “Command Mode”.

Figura Schema HC-05 Command Mode

Seguendo questo schema abbiamo attivato la command mode e lo abbiamo ipostato come volevamo grazie a dei comandi inseriti nella console seriale.

Per quanto riguarda il codice scritto su android studio abbiamo creato una classe Thread per gestire gli slider chiamata CrowlerThread. Questa classe è stata implementata in modo da poter utilizzare nella main activity due slider che si occupano di assegnare ai motori la potenza desiderata. CrowlerThread inoltre invia anche i pacchetti di dati direttamene tramite bluetooth al ricevitore dell’arduino. A sua volta l’arduino, tramite un protocollo precedentemente descritto, interpreta i dati e comunica con i controller degli step motor.

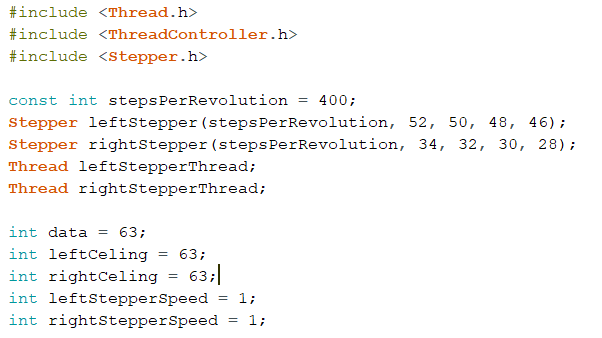
## 

Il metodo connect invece viene invocato al click del button connect. Come prima cosa egesue un ciclo di tutti i dispositivi bluetooth disponibili in cerca del ricevitore bluetooth del Tankino. Una volta trovato stabilisce una connessione aprendo un canale stream di uscita e se non si verificano errori esegue la thread che comincerà a mandare i dati degli slider.

## 

Il metodo run della classe CrowlerThread invia ogni 100 ms i dati in formato binario al bluetooth dell’arduino seguendo il protocollo prestabilito.

Ora invece passiamo al lato di programmazione svolto su arduino.  
Effettivamente in arduino IDE abbiamo scritto il codice che tramite il protocollo di comunicazione traduce i dati ricevuti dal bluetooth in grandezze utilizzabili come velocità per i motori.

All’inizio del programma abbiamo importato le librerie necessarie per lo sviluppo. Infatti, per riconoscere un oggetto di tipo *Stepper* (rappresentazione logica dello step motor) si necessitava di scaricare e includere tale libreria (riga 3). Questo ci ha facilitato di molto il controllo dei motori; tramite i parametri passati alla creazione di un nuovo oggetto stepper (step per revolution, in1,in2,in3,in4) e i suoi due principali metodi (setSpeed(int) e step(int)) è stato un gioco da ragazzi.

Per controllare singolarmente e senza interferenza i motori abbiamo dovuto utilizzare due Thread in modo che si muovano in modo asincrono.

Questa porzione di codice viene eseguita ripetitivamente durante tutto il ciclo di vita del programma. Qui avviene la vera propria traduzione dei dati ricevuti dal bluetooth e il controllo dei motori. Innanzitutto, se necessario le due Thread vengono avviate. In seguito, si leggono i dati ricevuti e vengono modificati di conseguenza le velocità dei due motori.

Purtroppo, per modificare la velocità dei motori non basta impostarla nel metodo *set speed* ma bensì si deve richiamare con una frequenza consona il metodo *step*. Per fare ciò abbiamo utilizzato il metodo *setInterval* che varia l’intervallo tra una chiamata e l’altra al metodo *step* (in ms).

## Assemblaggio Tankino

Grazie al contributo del professore Petrini abbiamo potuto avere dei pezzi su misura per il nostro Tankino. Abbiamo così ottenuto le due tavole di legno e le ruote in plexiglas.

Figura Motori nema 8 con ruote personalizzate

In questa foto si possono vedere le due incollate ai due motori nema 8.

Abbiamo preferito delle ruote personalizzate rispetto quelle lego per ragioni di estetica e di compatibilità con i motori.

Per quanto riguarda i cingoli invece abbiamo tenuto quelli lego e perciò anche la forma delle nostre ruote combacia in modo da aderire bene ad essi.



Figura Base inferiore con powerbank

Passando alla base, nella parte inferiore c’è un’incavatura che permette l’inserimento della powerbank che alimenterà il Tankino.

Fatto ciò abbiamo attaccato la base superiore, essa presenta degli alloggi in cui fissare l’arduino mega e i controller.

Figura Basi attaccate con la powerbank all’interno

Per stabilizzare ci siamo provvisti di due piccole assi di legno della lunghezza necessaria per lasciare la possibilità di utilizzare le porte della powerbank con facilità.

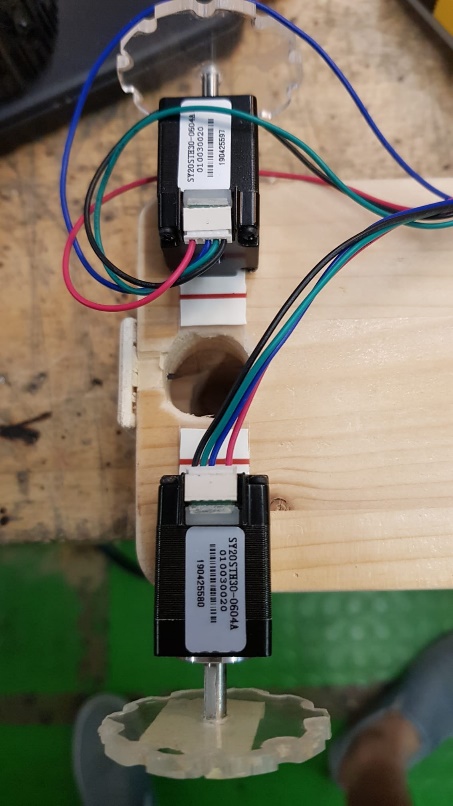
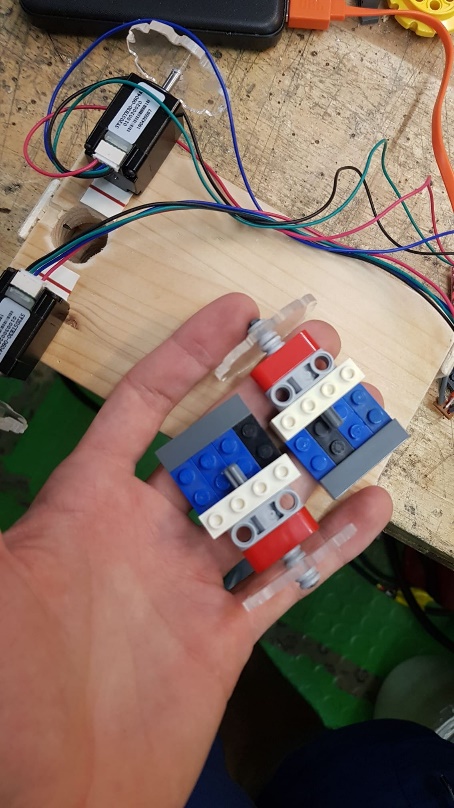
Successivamente abbiamo attaccato i motori sotto alla base inferiore.

Figura Supporti ruote senza motore

Figura Motori nema 8 affrancati sotto alla base inferiore

Inoltre, dato che i cingoli hanno bisogno di due ruote ciascuno per funzionare correttamente, abbiamo aggiunto due supporti senza motore.

Attivati a questo punto ci siamo accorti che c’era un ulteriore problema: un paio di ruote avevano una forma che non permetteva di far girare bene il cingolo. Per migliorare un po’ la situazione le abbiamo limate.

Il risultato finale della parte inferiore è questa.

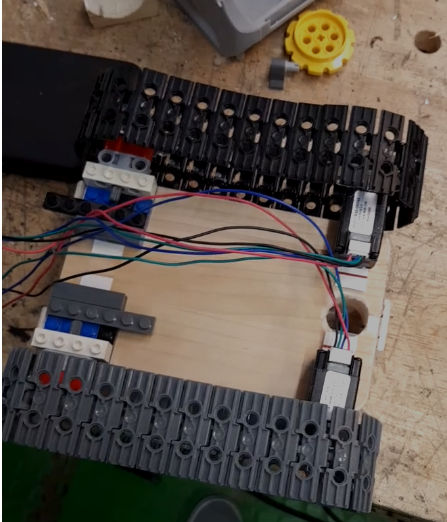


Figura Motori e supporti assemblati con la base inferiore

Per quanto riguarda invece la parte superiore abbiamo affrancato l’arduino utilizzando i buchi appositi mettendoci delle viti.   
Per fare funzionare l’arduino e i motori abbiamo ovviamente collegato il tutto all’alimentazione.

Questo è il risultato della parte superiore del Tankino.

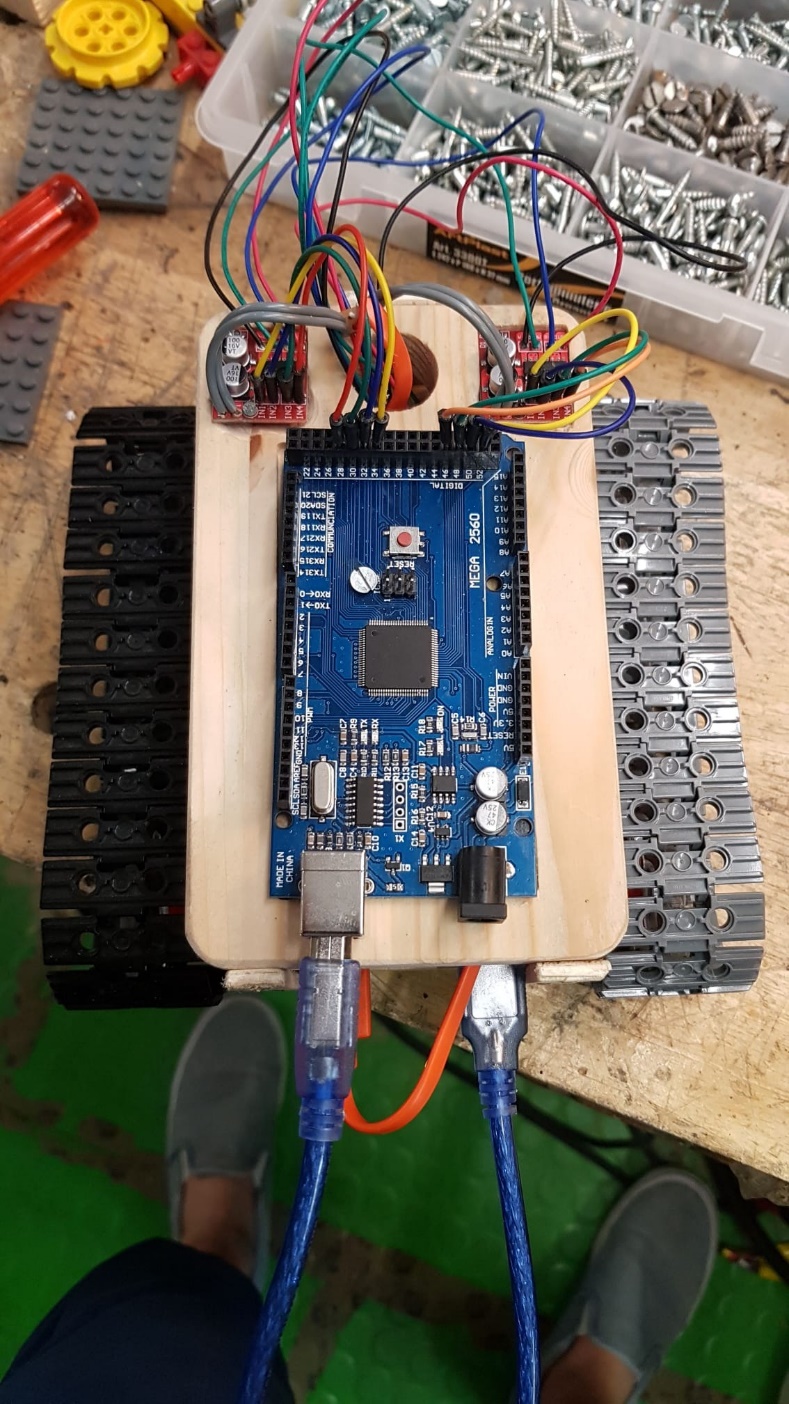
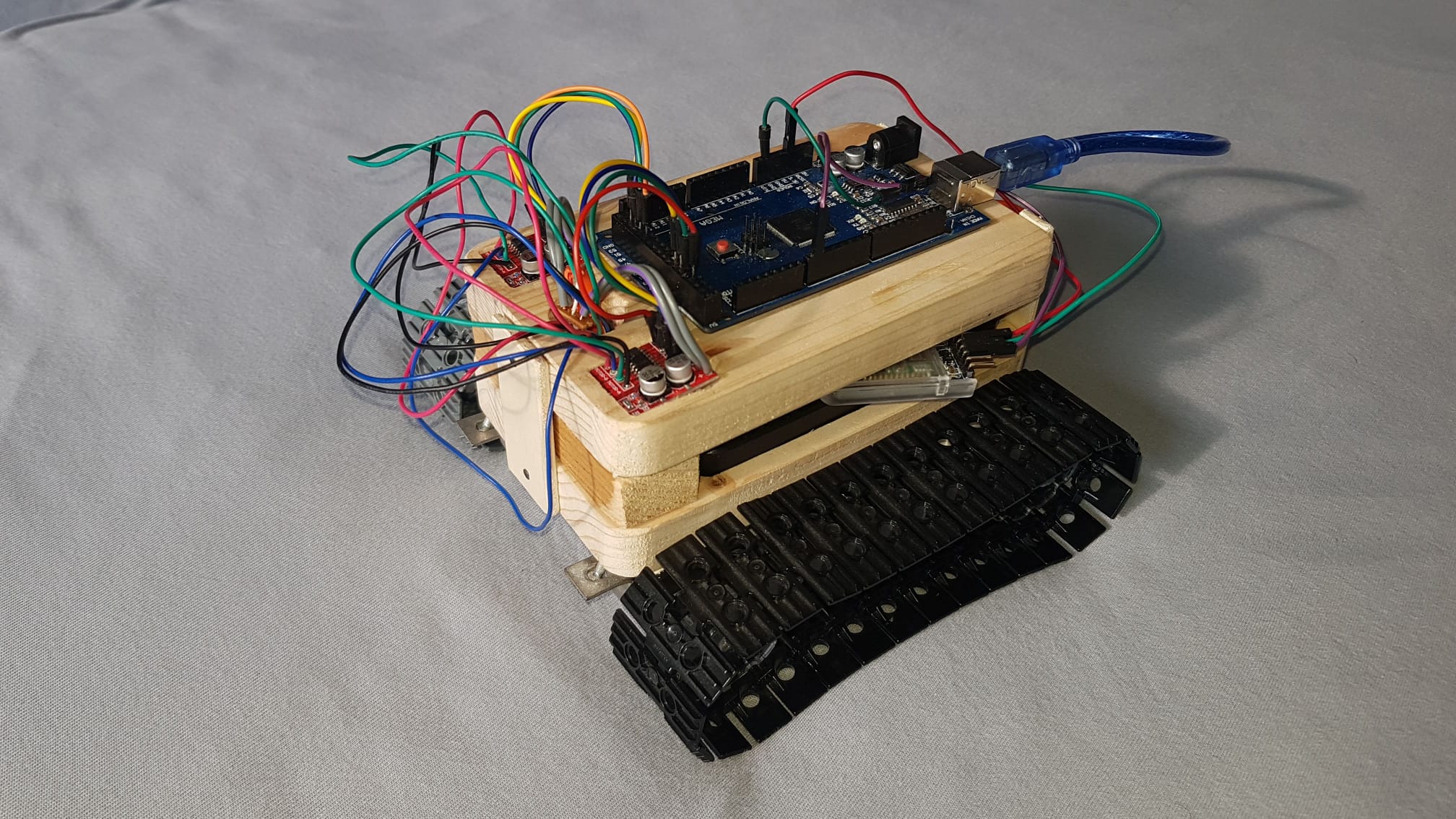


Figura Risultato finale parte superiore Tankino

Con questo abbiamo terminato l’assemblaggio del Tankino, ed ora che è assemblato abbiamo scoperto che i motori sembrano non avere la potenza necessaria a far muovere il mezzo. Il risultato finale del Tankino è questo:

# Test

## Protocollo di test

|  |  |
| --- | --- |
| Test Case | TC-001 |
| Nome | Avere un’applicazione mobile android |
| Riferimento | REQ-001 |
| Descrizione | Test case per verificare di avere un’applicazione android funzionante. |
| Prerequisiti | Avere un dispositivo android. |
| Procedura | Installare l’applicazione tramite l’apk su un dispositivo android e aprirla. |
| Risultati attesi | L’applicazione viene installata e aperta senza problemi. |

|  |  |
| --- | --- |
| Test Case | TC-002 |
| Nome | Avere un’arduino funzionante |
| Riferimento | REQ-002 |
| Descrizione | Test case per verificare di avere un’arduino funzionante. |
| Prerequisiti | Avere un pc a disposizione. |
| Procedura | Collegare l’arduino al computer e provare a caricarci un programma. |
| Risultati attesi | L’arduino viene individuato dal computer e esegue il programma correttamente. |

|  |  |
| --- | --- |
| Test Case | TC-003 |
| Nome | Test guidabilità veicolo |
| Riferimento | REQ-003 |
| Descrizione | Test case se L’applicazione permette di pilotare il veicolo |
| Prerequisiti | Essere connessi via bluetooth al tankino |
| Procedura | Provare a muovere entrambe gli slider sull’applicazione |
| Risultati attesi | Il tankino dovrebbe muoversi. |

|  |  |
| --- | --- |
| Test Case | TC-004 |
| Nome | Pairing bluetooth |
| Riferimento | REQ-004 |
| Descrizione | Essere in grado di fare un pairing bluetooth |
| Prerequisiti | Avere l’applicazione installata |
| Procedura | Cliccare sul tasto “connect” e provare a muovere gli slider |
| Risultati attesi | Il tankino dovrebbe muoversi. |

|  |  |
| --- | --- |
| Test Case | TC-005 |
| Nome | Test curve |
| Riferimento | REQ-006 |
| Descrizione | Il veicolo deve essere in grado di curvare |
| Prerequisiti | Essere connessi via bluetooth al tankino |
| Procedura | Provare a impostare gli slider mettendo quello a sinistra a un livello superiore e poi fare il contrario |
| Risultati attesi | Il tankino dovrebbe prima muoversi verso destra e poi verso sinistra. |

|  |  |
| --- | --- |
| Test Case | TC-006 |
| Nome | Download applicazione |
| Riferimento | REQ-008 |
| Descrizione | Avere un sito dove scaricare le applicazioni |
| Prerequisiti | Nessuno |
| Procedura | Andare sul sito ufficiale di Tankino e cliccare sul tasto download nella homepage. |
| Risultati attesi | Dovrebbe partire un download dell’applicazione. |

|  |  |
| --- | --- |
| Test Case | TC-007 |
| Nome | Test velocità |
| Riferimento | REQ-010 |
| Descrizione | La velocità deve essere variabile |
| Prerequisiti | Essere connessi via bluetooth al tankino |
| Procedura | Muovere gli slider verso l’alto e il basso |
| Risultati attesi | I motori del Tankino girano a velocità diversa in base al valore impostato con gli slider |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Alla fine di questo progetto ci troviamo con alcune mancanze. Questo dovuto o alle mancanze di risorse o per mancanza effettiva di tempo. La prima caratteristica che non abbiamo implementato è l’ausilio di luci sul tankino.

# Consuntivo

Questo è il nostro

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc

# Bibliografia

## Bibliografia per articoli di riviste

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo,

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

Esempio:

* <http://standards.ieee.org/guides/style/section7.html>, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o Qdc
* Prodotto
* …