

ROBOCUPJUNIOR RESCUE 2021

TEAM DESCRIPTION PAPER

Salbeghi

Abstract

- Il nostro robot è stato progettato e costruito con l'obiettivo non solo di svolgere le prove, ma di farlo al meglio, ottimizzando ogni movimento eseguito dal robot e superando gli ostacoli nel miglior modo possibile. La macchina è stata progettata completamente al CAD per utilizzare al meglio tutti i volumi. È stato implementato un sistema di sospensioni e il sistema di rilascio kit a doppio motore. Il robot è progettato in modo da essere costruito su una piattaforma di alluminio tagliata al laser. Sono montate due telecamere, una per lato che tramite OpenCV ci consentono di riconoscere le vittime visuali. Durante l'esplorazione del labirinto il robot esegue la mappatura del labirinto e tramite l'algoritmo Dijkstra raggiunge la casella più vicina eseguendo il percorso più breve.



1. Introduction

a. Team

- Il nostro team è composto da tre persone, ci siamo suddivisi i compiti in base alle nostre passioni e competenze sviluppate nel tempo.
 - a. Pogetta Matteo (Capitano) -> Software - Algoritmi - Mappatura - Progettazione 3D
 - b. Berlato Matteo -> Software -> Sensoristica - Test
 - c. Grendene Marco -> Hardware -> Assemblaggio - Test
- Questo è il terzo anno che partecipiamo a questa competizione, sempre nella categoria Rescue Maze, con il nostro primo robot abbiamo partecipato ai campionati nazionali, qualificandoci per i campionati Europei ad Hannover, conquistando il settimo posto.
Questa prima esperienza ci ha permesso di acquisire l'esperienza necessaria per la creazione del nuovo robot, nonché sperimentare numerose soluzioni hardware e software innovative.



2. Project Planning

a. Overall Project Plan

- Il principale scopo per cui partecipiamo alla competizione è quello di sviluppare le nostre competenze tecnico-informatiche, ma soprattutto le competenze trasversali, a livello interpersonale. Inoltre, volevamo riuscire a costruire una macchina che potesse oltre che a gareggiare in queste

competizioni, essere in qualche modo d'ispirazione per applicazioni di salvataggio in ambienti di reale pericolosità dove un robot può salvare delle vite senza metterne in pericolo altre.

- Grazie alla prima esperienza, abbiamo appreso quali limiti e difficoltà il robot incontra a causa della sua struttura e programmazione; perciò abbiamo sviluppato questo robot cercando di risolvere i problemi del precedente (a livello strutturale), implementando anche nuove soluzioni. Queste soluzioni derivano sia sperimentazioni personali, sia dalle esperienze vissute dalle precedenti squadre dell'istituto, ma anche da quelle a livello internazionale.
- Con questo robot abbiamo cercato di introdurre delle innovazioni o migliorare delle funzioni già viste. Abbiamo voluto creare una struttura rigida che circondasse il robot, per fare questo abbiamo tagliato al laser un pianale in alluminio da 4mm su cui si agganciano tutte le altre strutture. In aggiunta, abbiamo implementato un sistema di sospensioni, rendendo indipendente il movimento di ogni motore e agganciandolo alla struttura principale tramite una sospensione da 60mm e due cuscinetti a sfera da Ø8/16mm. Oltre a questo abbiamo realizzato dei finecorsa a contatto diretto tra le sue due parti, la parte esterna tagliata al laser chiude il circuito toccando la base portata a massa. Grazie all'utilizzo della resina come materiale di stampa (ABS like) e alla completa progettazione in ambiente virtuale CAD 3D (Autodesk) abbiamo reso modulare il robot. La completa progettazione ci ha permesso di creare un rilascio kit di soccorso a doppia uscita, i kit vengono veicolati tramite due motori (DC e servo 6V) per poi essere espulsi tramite due rampe. Dal lato software abbiamo implementato vari algoritmi, tra cui due controlli a retroazione negativa PID, mappatura tramite Dijkstra e movimentazione tramite encoder e giroscopio.
- Il robot è stato sviluppato interamente per competere in questa categoria, affidandoci anche all'esperienza di altri gruppi passati. Le sue dimensioni, la forma, i materiali, e anche il software sono stati studiati in base ai regolamenti internazionali e al campo di gara.



b. Integration Plan

- Il robot è stato interamente progettato ed assemblato in ambiente virtuale, questo ci ha permesso di creare un assieme in cui tutte le parti interagissero tra loro nel miglior modo possibile. In aggiunta non va dimenticato che nel momento di sviluppo eravamo già consapevoli delle necessità hardware e software, dato dalle esperienze passate.

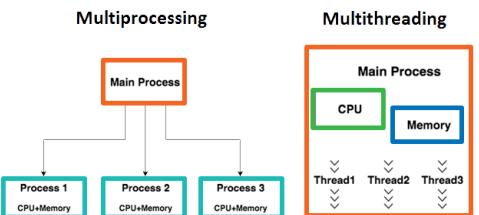
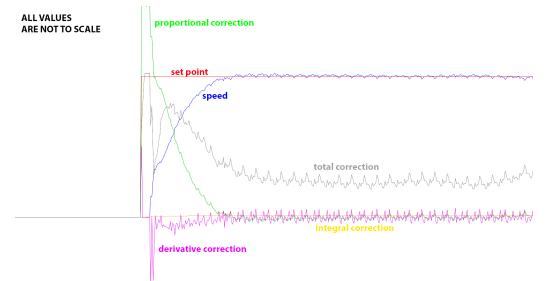
c. Testing

- Per testare le prestazioni e l'affidabilità del robot, abbiamo inizialmente provato ogni parte singolarmente per capire le sue prestazioni e poi creato l'insieme generale. Una volta che tutte le parti avessero funzionato singolarmente, abbiamo iniziato a unirle, per verificare man mano le prestazioni. Una volta che il robot è stato completato con tutte funzioni, abbiamo iniziato a testare tutto il sistema nei campi regolamentari, aggiungendo anche ostacoli e difficoltà non previste.
- Per gli algoritmi di mappatura, abbiamo creato anche un simulatore per riuscire ad analizzare le prestazioni dei vari programmi, il simulatore prende in ingresso una mappa di qualunque dimensione e ci mostra i vari passi, ed alla fine quanto è stata efficace l'esplorazione.
- I test della parte hardware sono stati eseguiti sia dall'ambiente virtuale 3D, sia in condizioni reali tramite prove realizzate per testare i limiti del robot (labirinti con molti speedbumps, oppure con vittime, ostacoli in posizioni inusuali ecc.).
Per quanto riguarda i test software, in particolare per verificare il funzionamento del sistema di mappatura, come detto prima, abbiamo creato un simulatore (usando la libreria PyGame) che eseguisse il codice caricato nelle schede del robot così da vedere ancora prima di portarle nella realtà, se le funzionalità aggiunte potevano essere valide.

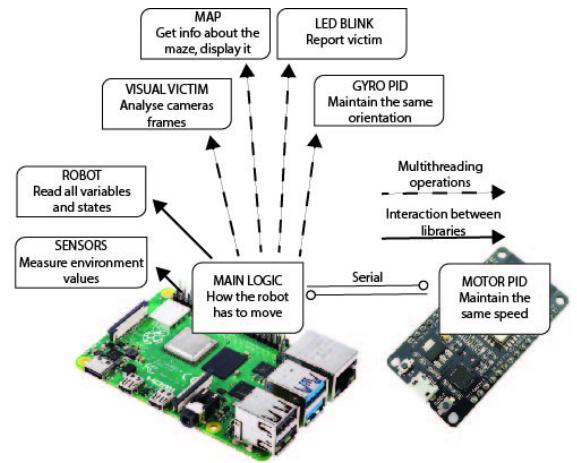
3. Software

a. General software architecture

- Il software del nostro robot si divide principalmente in due parti, quella relativa al controllo PID dei motori effettuato nella scheda ESP32 e la restante parte eseguita nella scheda Raspberry Pi.
 - ESP32: programmato in C con il suo framework ESP-IDF, è utilizzato per il controllo dei motori e la lettura dei sensori con uscita analogica. Comunica tramite seriale al raspberry pi. Questa scheda è collegata a uno shift register 74hc595 per l'espansione delle porte digitali in output, in particolare le uscite dello shift register comandano i segnali digitali dei driver per i 4 motori principali. Per rendere il robot il più controllato possibile abbiamo avuto bisogno di implementare vari controlli, tra cui un controllo della velocità dei motori di trazione. Per fare questo abbiamo montato su tutti i motori un encoder per la misurazione della velocità e realizzato un controllo PID. Quindi i quattro motori sono controllati tramite un controllore software Proporzionale- Integrale- Derivativo (PID), che mantiene costante la velocità voluta. Viene eseguito sulla scheda ESP32, i 4 motori sono indipendenti, ovvero si calcola un controllo PID per ogni singolo motore. Il PID è un controllo a retroazione negativa che agisce da rete correttiva, il processo acquisisce in ingresso la velocità istantanea del motore e lo confronta con il valore di riferimento, ricava l'errore come sottrazione dei due e procede a calcolare le tre correzioni, proporzionale, derivata e integrale.
 - COMUNICAZIONE SERIALE: La comunicazione tra Esp32 e Raspberry Pi avviene tramite un bus seriale di tipo RS-232 opportunamente gestito, si tratta di un protocollo a 1.000.000 baud, 8 bits di dati per trasmissione, 1 bit di controllo di parità e 1 bit di stop. La comunicazione è bidirezionale, ma solo il raspberryPi fa le richieste all'Esp, e questo risponde con i valori desiderati. Il protocollo dati è stato strutturato esattamente sui dati voluti, permette poca espansione ma riduce i byte inviati. L'ESP32 è programmato in modo tale che se non riceve nulla per più di un secondo blocca i motori impostando la loro velocità a 0. La comunicazione è completata anche da un segnale digitale condiviso tra le due schede, questo viene portato alto dal ESP32 quando è pronto, ovvero se è acceso e avviato correttamente, così il raspberry pi riesce a sapere quando l'ESP è spento, e anche quando ha problemi di avviamento, così da poterlo resettare tramite un altro segnale digitale collegato sul piedino di reset dell'ESP.
 - RASPBERRY PI: La struttura del codice del Raspberry Pi è organizzata secondo diversi file, contenenti diverse classi (sensori, mappa, movimenti, interpretazione delle immagini e logica principale). Il raspberry Pi quindi deve eseguire numerosi programmi in parallelo, tra cui la logica principale, due riconoscimenti di immagini e la visualizzazione della mappa. Abbiamo quindi deciso di dividere i programma in più processi paralleli, usano multiprocessing, che permette a costo dell'isolamento delle varie strutture di dati delle varie sezioni, una maggiore fruibilità delle risorse del raspberry, tra cui CPU e Ram. Questo poiché i processi vengono considerati come completamente separati, al contrario del multithreading che esegue più parti del codice in "parallelo", ma queste sono considerate lo stesso processo. La comunicazione tra i vari processi avviene tramite l'utilizzo di apposite queue che creano dei ponti tra un processo e l'altro. Si utilizzano poi svariati multithreading all'interno delle varie sezioni generali, in particolare la parte di logica si compone di un multithreading per il PID del giroscopio e uno per l'attivazione dei led di segnalazione vittima, ogni programma per il



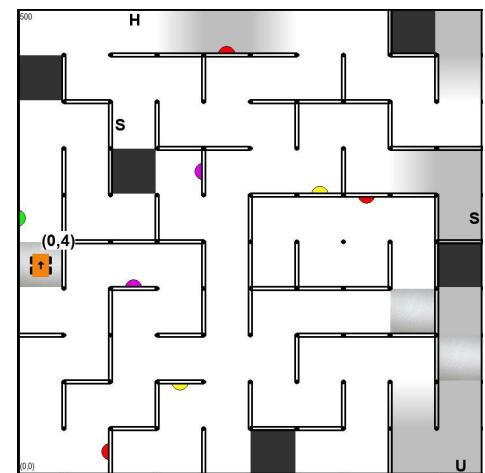
- riconoscimento delle telecamere fa uso di un thread per la presa continua dei frame dalla telecamera.
- d. **LOGICA A STATI FINITI:** La logica principale del robot è stata programmata seguendo una logica a stati finiti che permette di frammentare il codice in moltissime sezioni e farle collaborare. Queste piccole sezioni di codice possono essere eseguite una dopo l'altra in modo continuo così che sia possibile seguire contemporaneamente più linee logiche. Questo tipo di programmazione ci permette di non dover ripetere molte parti del codice, ad esempio la comunicazione con la scheda ESP avviene una volta per ciclo, questo consente di essere sicuri del funzionamento dell'ESP stesso, che i motori abbiamo la velocità voluta esatta, e di avere distanza e il valore della fotoresistenza sempre aggiornati. Questo discorso vale anche per i sensori laser, di temperatura, delle vittime visive trovate dai riconoscimenti in multiprocessing, di avere la visualizzazione della mappa sempre aggiornata. La struttura di dati che è stata creata per questa logica è suddivisa in numerose classi, in cui vengono salvate tutte le variabili corrispondenti alla funzione assegnata. Il comando di tutte le sezioni di codice è gestito tramite una logica molto complessa che fa uso di molte liste e livelli.
- e. **MAPPA E ORIENTAMENTO:** La creazione della mappa ha richiesto una particolare logica di orientamento, lo scopo è quello di riuscire a far muovere il robot all'interno della mappa esattamente come si muove all'interno del labirinto. Il compito risulta abbastanza complesso data la natura stessa del labirinto, rampe, ostacoli e speed bumps, sono elementi che causano al robot movimenti difficilmente controllabili e rilevabili. Per questo abbiamo implementato un metodo per trasformare i dati di distanza e di giroscopio nei valori di X, Y, Z (coordinate considerando il labirinto in un piano cartesiano), G (rotazione del robot) nel piano della mappa. Per il calcolo delle tre componenti cartesiane a ogni ciclo viene aggiunto lo spostamento compiuto ciclo per ciclo, questo è dato dalla differenza di distanza percorsa, quella attuale meno quella rilevata dallo scorso ciclo, per le componenti trigonometriche.
- f. **MOVIMENTI:** I movimenti del robot sono stati progettati per essere delle funzioni della logica principale richiamabili quando se ne ha necessità, facendo parte di una logica a stati sono più complessi ma molto efficienti. Le funzioni programmate sono per esempio dritto, gira, finecorsa, allinea, centra, vittime ecc.. Queste possono essere attivate modificando dei flag e impostando su apposite variabili le impostazioni desiderate, come nel caso del gira la velocità e la distanza da percorrere.
- g. **MAPPATURA:** Il robot deve essere in grado di esplorare in autonomia tutto il labirinto, per fare questo abbiamo dovuto implementare un algoritmo di mappatura, ovvero un processo che salva il labirinto, mano che viene esplorato, all'interno di un'apposita struttura di dati, e calcola il percorso migliore per esplorare il labirinto. Il robot non può sapere la mappa a priori, perciò devono essere implementate regole dinamiche. La regola utilizzata di base è quella della "mano destra" per cui il robot esplora il labirinto il labirinto dando la precedenza sempre alla sua destra, in pratica il robot segue sempre il muro alla sua destra, questo permette in labirinti perfetti di esplorare tutte le zone e di tornare al punto iniziale, ma per labirinti non perfetti, con stanze e cicli come nel nostro caso, c'è bisogno di qualcosa in più. Noi abbiamo risolto questo implementando insieme alla regola della mano destra anche un algoritmo per trovare i cammini minimi. L'algoritmo scelto è quello di Dijkstra, derivante



dalla teoria dei grafi, permette di calcolare i cammini minimi in un grafo a pesi positivi. Il labirinto è un particolare tipo di grafo, dove ogni casella corrisponde ad un nodo e il peso di ogni arco è uguale a 1. Data la casella di partenza, ovvero la posizione attuale del robot, l'algoritmo calcola il percorso più breve da percorrere per arrivare in qualsiasi altra casella inesplorata, o a quella di partenza. Perciò l'algoritmo di esplorazione totale si basa su due piani, se è direttamente disponibile una casella inesplorata a fianco alla casella attuale del robot, segue le regole della mano destra, altrimenti esegue l'algoritmo di Dijkstra per arrivare alla casella inesplorata più vicina, o se non c'è ne sono più, per tornare alla casella iniziale. Il vantaggio di questa soluzione è che funziona senza una "memoria" delle azioni precedentemente eseguite, ogni volta che il robot entra in una nuova casella, se non ci sono caselle inesplorate disponibili affianco, esegue Dijkstra da capo senza tenere conto delle azioni precedenti.

- h. SIMULATORE: Il codice della mappa è stato programmato e poi testato su un PC per poi essere trasferito sul Raspberry Pi ed essere integrato al resto del codice. Per simulare il funzionamento del robot abbiamo creato un simulatore basato sulla libreria pygame. Il visualizzatore non viene usato solo in fase di simulazione nel PC, ma anche nel reale funzionamento del robot, mostrando quello che il robot sa, dove pensa di essere, e le decisioni che prende, tutto questo anche per una questione di debug.

- i. RICONOSCIMENTO TELECAMERE: Le vittime visive vengono rilevate dal robot tramite due telecamere, una a destra e una a sinistra, queste vengono lette dal raspberry pi con le dovute procedure di computer vision. La libreria utilizzata è opencv, che permette sia la cattura dell'immagine sia la sua elaborazione. Per riuscire a "vedere" il più possibile, possibilmente tutto il muro del labirinto, così da poter rilevare le vittime solo a centro casella e non in continuazione, liberando potenza di calcolo per il resto dei programmi, sono state usate delle telecamere a 180°. Queste telecamere hanno un elevatissimo effetto fisheye, che rende molto difficile il riconoscimento degli elementi, poiché variano molto la loro forma a seconda della distanza dal centro dell'obiettivo, per risolvere questo problema abbiamo mappato l'obiettivo della telecamera con le dovute procedure e sviluppato le matrici di correzione. Queste matrici, calcolate una sola volta, vengono applicate ad ogni immagine in ingresso così da 'raddrizzare' l'immagine e avere una visuale non deformata. Il riconoscimento delle lettere e delle forme colorate è stata sperimentata in più modi, si tratta comunque sempre di binarizzare l'immagine (portare l'immagine in bianco e nero), rilevare i bordi, poi la forma e il colore. Una volta eseguita la binarizzazione si procede all'individuazione dei contorni dell'immagine, questa è data dalla funzione findContours di opencv, che rilascia in output una lista di liste di tutti i vertici dei contorni rilevati, questa gerarchia di contorni viene analizzata intorno per intorno. Ogni singolo contorno chiuso viene poi approssimato da una funzione contenuta in opencv, che a seconda della tolleranza inserita diminuisce sensibilmente il numero di vertici. I controlli applicati ai colori prevedono principalmente il calcolo del rettangolo circoscritto al contorno, ovvero che comprende tutti i bordi, e delle sue dimensioni rispetto all'immagine, ovvero non deve essere troppo piccolo o troppo grande, deve essere il più possibile quadrato, ovvero non avere una forma troppo

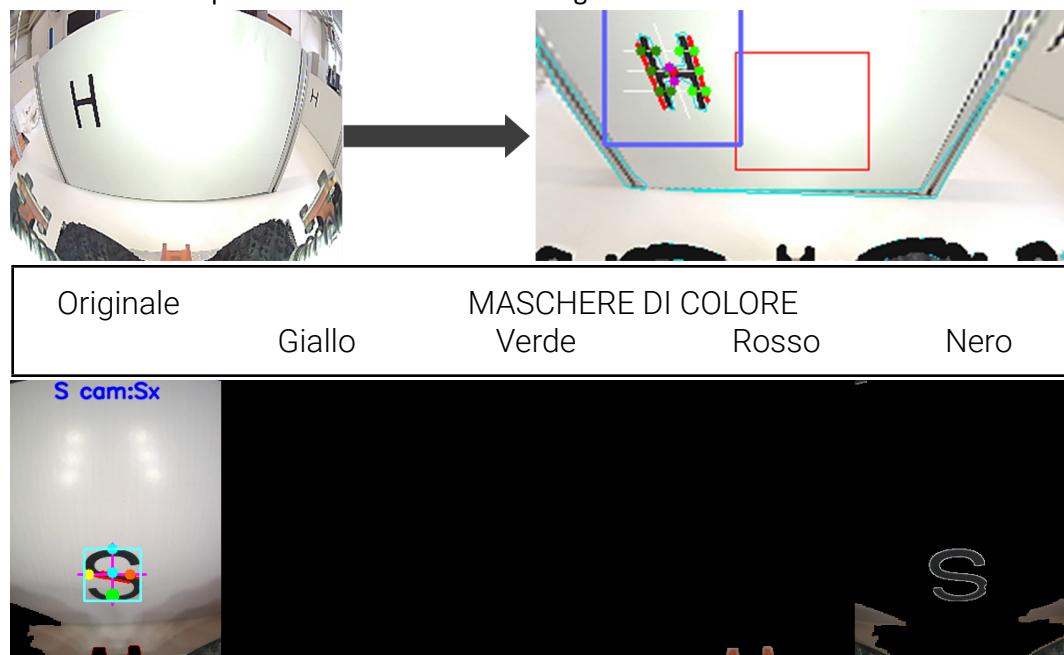
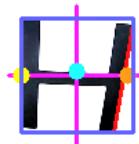


Legend:

- Victims:
 - red shape
 - yellow shape
 - green shape
- letterH H
- letterU U
- letterS S
- heat
- Ramp
- Checkpoint
- Black tiles
- Second floor
- Unexplored



allungata, non deve toccare la parte superiore o inferiore del frame. Il numero di lati non viene preso in considerazione in quanto secondo le ultime versioni del regolamento le vittime visive colorate possono avere qualsiasi forma. I controlli per le vittime invece si compongono in una prima fase degli stessi delle forme colorate, per poi passare a capire di che lettera si tratta. Per fare questo vengono esplicitati i due lati del contorno approssimato più lunghi, che per la H e per la U dovranno essere ai lati verticali del rettangolo circoscritto ed avere circa la sua stessa altezza, questo non vale per la S. Un altro controllo vede il tracciamento di due direttive rispettivamente in orizzontale e verticale che tagliano in 4 parti uguali in rettangolo circoscritto. Queste direttive vengono percorse, e vengono individuati i punti in cui il colore passa da bianco a nero, la loro posizione e il loro numero determinano insieme alla condizione di prima di cosa si tratta. Il riconoscimento tramite le telecamere non è solo in grado di riconoscere le vittime ma anche di vedere le caselle nere, e modificare il comportamento del robot di conseguenza.



b. Innovative solutions

- Abbiamo sperimentato numerose soluzioni non convenzionali per il codice di questo robot, dalla creazione di numerosissimi processi paralleli alla creazione di una logica completamente a stati, quest'ultima che comprende una gestione molto onerosa e complicata.
- Anche per il rilascio kit abbiamo implementato un sistema in grado di capire sempre quanti kit di soccorso sono presenti nel caricatore e se un eventuale rilascio è andato a buon fine, tutto questo grazie ad un motore servo e ad un motore dc con encoder incrementale.
- Anche per le telecamere abbiamo studiato e sviluppato le trasformazioni per la correzione dell'obiettivo fisheye. Oltre a questo siamo in grado di riconoscere le lettere senza nessuna funzione particolare ma soltanto dalla semplice analisi dell'immagine binarizzata. Inoltre abbiamo appena imparato la possibilità di riconoscere le caselle nere tramite le telecamere.

4. Hardware

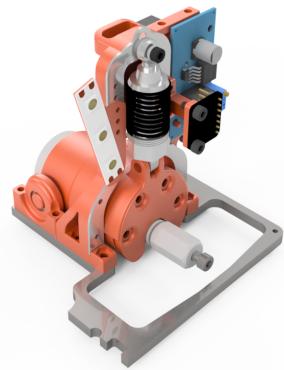
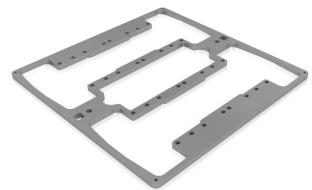
- Questo robot è stato da noi interamente disegnato in 3d prima di essere realizzato, questo ci ha permesso di sperimentare varie soluzioni e di avere una visione del robot nella sua completezza.

Anche se sono state apportate numerose modifiche in corso d'opera, il progetto iniziale non è mai variato. Lo sviluppo in 3D ci ha consentito di calcolare tutti gli ingombri, il posizionamento dei sensori e degli attuatori per garantire il massimo utilizzo di ogni spazio ma anche di garantire comodità e accessibilità ai vari componenti. La progettazione è stata fatta con la consapevolezza del materiale di stampa: la resina, come vedremo in seguito, e dei volumi di stampa; abbiamo creato una struttura modulare, con molti pezzi ma di piccole dimensioni. Questa modularità ci è servita per riuscire a modificare delle parti anche successivamente e di aggiungere moduli prima non considerati. Come si può vedere dai vari render non è un robot classico, abbiamo deciso di sperimentare nuove soluzioni, come ad esempio le sospensioni, oppure il rilascio kit a doppio motore.



a. Mechanical Design and Manufacturing

- **PIANALE:** Il pianale del robot, dove si fissano tutte le altre parti, è costituito da una lamiera di alluminio da 4mm, di dimensioni massime di 200x200 mm, progettata in 3d e tagliata al laser da un'azienda affiliata. Questa ci ha permesso di avere un bordo esterno al robot liscio, nel quale fosse difficile incastrarsi, ed una grande solidità strutturale. Sulla base sono stati avvitati tutti gli altri componenti del robot, dai supporti delle sospensioni ai finecorsa.
- **SOSPENSIONI:** Le sospensioni sono un esperimento che abbiamo voluto introdurre per portare qualcosa di nuovo. Esse si agganciano da una parte all'estremità superiore degli "archi" e dall'altra ai motori con un supporto in resina che li ricopre quasi interamente, questo poi va ad agganciarsi alla lamiera tramite due mini cuscinetti. Sono utili per superare efficacemente gli ostacoli, soprattutto in collaborazione con il controllo PID della velocità eseguito su ogni singolo motore.
- **ATTUATORI:** Il robot è movimentato da 4 motori DC da 12V, ognuno con un motoriduttore che lo porta a 111 rpm, sono azionati da due driver L298N a ponte ad H comandati a loro volta per ogni motore da una coppia di segnali digitali, per il senso di marcia, e un segnale pwm, per la velocità. Un encoder incrementale è accoppiato ad ogni motore al fine di determinare la velocità esatta. Il rilascio kit invece è azionato da un motore DC da 6V con motoriduttore che lo porta a 750 rpm, il motore è azionato da un driver L293D a ponte ad H comandato sempre da una coppia di segnali digitali, per il senso di marcia, e un segnale pwm, per la velocità. Anche a questo motore è accoppiato ad un encoder incrementale così da permetterci di rilevare la velocità, ma soprattutto le rotazioni per il calcolo dei kit presenti sul caricatore. Lo spostamento poi del kit per la selezione dello scivolo di destra o sinistra è azionato da un servomotore TowerPro MG92B.
- **FINECORSO:** finecorsa sono stati costruiti da noi con delle lamiere tagliate al laser e piegate, con delle molle e delle viti.



- MECCANISMO DI RILASCIO KIT: Il meccanismo per rilasciare i kit di soccorso è composto da un caricatore centrale che può contenere fino a 12 kit di dimensioni 10x10x10 mm, i kit vengono spinti verso lo scarico tramite un cursore che avanza tramite la rotazione di una vite senza fine. Questa vite è una barra filettata fatta ruotare da un mini motore DC a 6V equipaggiato con un encoder incrementale. Lo scarico è formato da un settore di circonferenza con un incasso apposito per il kit, questo scarico può ruotare grazie ad un servomotore a destra o a sinistra, così che il kit possa cadere all'interno di uno dei due scivoli. La parte del rilascio superiore è indipendente meccanicamente dal resto del robot così da permetterne facilmente la rimozione per accedere alla pcb e ai componenti sottostanti. Principalmente è costituito da elementi stampati in resina, che permette una grande precisione, soprattutto per le rampe che devono essere perfette e molto levigate vista la poca pendenza. Sono presenti anche delle lamiere da 3mm per il fissaggio del motore e della barra filettata, questi danno anche un sostegno strutturale all'elemento principale. Il cursore è sempre costituito da della lamiera da 3mm tagliata al laser e successivamente filettata per essere in grado di scorrere alla rotazione della vite. Il sistema permette una veloce e semplice gestione dei kit, infatti grazie all'encoder è possibile calcolare il numero di kit all'interno del caricatore e se un eventuale rilascio è stato completato.



b. Electronic Design and Manufacturing

- Sensori:
 - Il robot è dotato di una sensoristica avanzata, che ci permette di avere delle misure dall'ambiente circostante molto precise. Per quanto riguarda l'orientamento nello spazio, la rilevazione di muri o di eventuali ostacoli sono stati utilizzati 5 VL6180X, sensori di prossimità laser di range fino a 255mm che comunicano tramite protocollo I2C alla scheda Raspberry Pi 4, sono stati scelti questi sensori per l'alto grado di precisione e di velocità in lettura (modificata secondo quanto indicato nel datasheet), nonché per il fatto che altri sensori IR sono molto influenzabili dalle condizioni di temperatura dell'ambiente, cosa che non accade con questi;
 - Sensori di temperatura TPA81, sono dei sensori composti da una matrice ad 8 pixel che rileva la temperatura in un range di circa 120 gradi e permettono quindi di rilevare una vittima riscaldata dalla distanza, dopo l'utilizzo di altri sensori ci siamo soffermati su questi perché avevano una precisione molto più alta rispetto ad altri sensori commerciali, inoltre l'utilizzo tramite I2C è molto semplice.
 - L'orientamento nello spazio è dato dal modulo BNO055 un giroscopio triassiale, accelerometro con magnetometro a 9 assi, utilizzato nella modalità IMU (Inertial Measurement Unit) che calcola l'orientamento nello spazio, come angoli di Eulero, partendo dai dati di accelerazione del giroscopio e dell'accelerometro. Abbiamo scelto questo giroscopio per la sua facilità di utilizzo tramite comunicazione I2C e la vastità di dati che ci riesce a fornire, tramite questo giroscopio capiamo come siamo posizionati nel labirinto e se ci stiamo muovendo o meno.

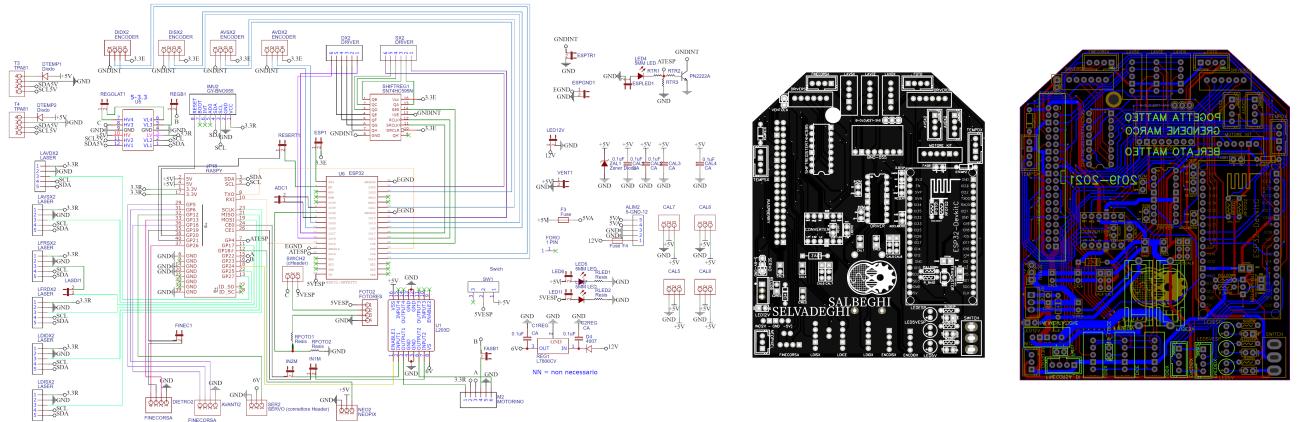
La lettura del colore è eseguita tramite una fotoresistenza posta in serie ad una resistenza fissa, saldata in una pcb creata appositamente da noi, ciò ci permette di scegliere quale valore di tensione leggeremo in base al colore della casella.

Tutti i motori DC utilizzati sul robot sono equipaggiati con un encoder incrementale rotativo a due fasi sfasate di 90 gradi, l'encoder dei motori principali compie circa 990 rotazioni per 1 rotazione della ruota, questo ci permette di calcolare non solo la velocità con segno del motore, ma anche la distanza percorsa dal robot.

I finecorsa sono stati costruiti da noi con delle lamiere tagliate al laser e piegate, con delle molle e delle viti.

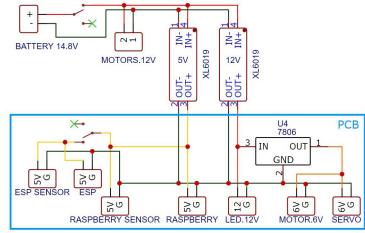
È costituito da due parti distinte, elettricamente isolate, una a contatto con la lamiera di base portata a GND, e l'altra mobile, e tenuta staccata dalle molle, collegata ad un ingresso digitale in pullup del raspberry. Questo fa in modo che quando le parti di un finecorsa vengono in contatto, il pin del raspberry viene portato a GND e quindi rilevabile come livello basso.

- Schede / microcontrollori:
 - Il robot è dotato di due schede, un Raspberry Pi 4, che si occupa di gestire la logica, l'analisi delle telecamere e la lettura dei sensori, ed un Esp32 che invece esegue il controllo di velocità motori PID e la lettura della fotoresistenza, le due schede comunicano tramite comunicazione seriale, tramite un protocollo di istruzioni creato appositamente da noi. La scheda Raspberry Pi 4 è stata scelta perché è una delle migliori schede che si possano utilizzare per interfacciarsi a delle webcam ed ha delle dimensioni assai contenute. La scheda Esp32 è stata utilizzata invece per la sua enorme capacità di calcolo, la possibilità di programmazione tramite il suo framework dà un controllo completo su ciò che si sta utilizzando, ed ovviamente per il fatto che possiede dei pin di lettura degli input Hardware.
 - PCB:
 - Per collegare tra loro tutte le schede ed i sensori è stata progettata e realizzata una scheda stampata su misura che occupasse tutto lo spazio disponibile nella struttura del robot e che facilitasse il collegamento di tutti i componenti elettronici. Avevamo in precedenza creato una scheda tramite la saldatura di cavi, ma questa diventava molto complessa da gestire e poteva capitare che col passare del tempo i cavi si interrompessero o staccassero, rendendo la fase di manutenzione molto complicata.
Il circuito stampato invece ha uno spessore di 1.6mm, si compone di due strati e permette di montare una serie di connettori JST 2,54mm, collocati in base alla posizione dei sensori, dei motori e delle strutture di base del robot.



- Alimentazione:

- Il robot è alimentato da una batteria lipo 14.8V 2300mAh, questa è interrotta da un interruttore principale per poi collegarsi direttamente ai driver, i motori sono alimentati quindi con la stessa tensione della batteria, ma essendoci il controllore PID la prestazione non è influenzata dalla tensione di alimentazione, se sopra i 10V.
Invece il circuito (PCB) è alimentato a 5V e 12V,
l'alimentazione dell'interruttore principale passa per due convertitori buck-boost XL6019 che abbassano quindi a 5V e 12V, il 5V non viene interrotto per l'alimentazione del raspberry, invece per l'esp32 è stato inserito un interruttore direttamente in pcb, questo interrompe i 5V all'esp e funge quindi anche da interruttore di funzionamento del robot, solo questo viene utilizzato in gara.



5. Performance evaluation

- Dopo questi anni di esperienza, di continue ricerche, innovazioni ed ore spese sullo sviluppo di hardware e software che si integrassero perfettamente tra loro, così da poter portare ad un livello sempre più alto il progetto finale, possiamo dire di aver creato una macchina molto performante in grado di superare anche le sfide più difficili. Come è possibile vedere dai video inviati, il robot segue sempre una logica precisa e non lascia spazio agli errori più comuni commessi, questo perché la ricerca ed i test effettuati sono stati pensati per eliminare queste tipologie di errore. Può ovviamente capitare che possa sbagliare in alcune occasioni, ma abbiamo cercato di ridurre al minimo questo, potendo invece affermare che nella grande maggioranza dei casi è in grado di esplorare completamente un campo di gara al 100%.

6. Conclusion

- In questa relazione non siamo riusciti a documentare con precisione tutto il lavoro, e perciò a rispondere esaustivamente a tutte le vostre domande. Per questo vi lasciamo in appendice i link diretti alla documentazione completa, al codice sorgente, a tutte le immagini e i video, nonché alla nostra pagine github. Così da condividere anche il nostro lavoro con la comunità di robocup e che quindi possa essere utile a futuri progetti.

Appendix and References

- Pagina GitHub del gruppo: <https://github.com/robocup-depre/Salbeghi>
- Canale Youtube del gruppo: https://youtube.com/playlist?list=PL4E3h3ClAotzZJWBBYdVjQWadXayW_QSV
- Cartella Drive del gruppo: [SALBEGHI_WEB](#)