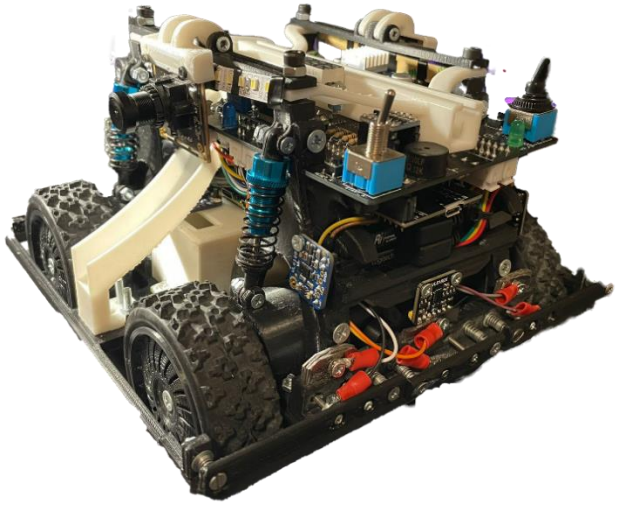
ROBOCUPJUNIOR RESCUE MAZE 2023

TEAM DESCRIPTION PAPER

SENSA SCHEI

Abstract

- Il nostro robot è stato ideato e costruito per essere versatile, permettendogli di adattarsi a qualsiasi tipo di situazione che incontra grazie alla rigida struttura e al sistema di sospensioni che facilita il superamento di qualsiasi ostacolo che si ritroverà davanti. Grazie alla cauta progettazione meccanica ed elettrica, è stato possibile compattare tutto il robot in dimensioni minime, garantendo però la massima accessibilità e manutenzione. Infine, grazie ad un rigoroso impegno nella programmazione, riesce a percorrere qualsiasi sentiero con la massima precisione attraverso il sistema PID, riuscendo così a riconoscere qualunque vittima che incontra utilizzando le due telecamere poste sui fianchi.



1. Introduction

a. Team

- Siamo una squadra di tre persone e abbiamo diviso i vari compiti in base alle nostre conoscenze e passioni:

  - Tuca A. Daniel (capitano): Hardware, Assemblaggio, Sensoristica, Test
  - Mendo Martino (co-capitano): Design, Software, Test
  - Pivirotto Riccardo (co-capitano): Software, Algoritmi, Mappatura, Test

2. Project Planning

a. Overall Project Plan

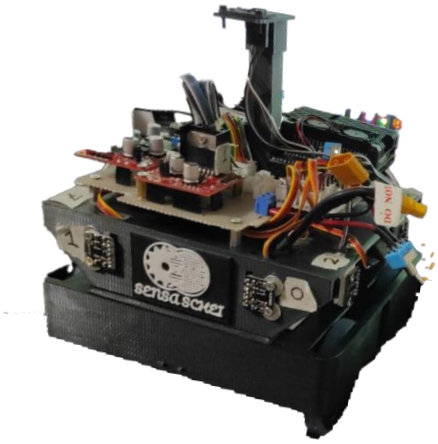
- Scopo principale per cui abbiamo deciso di intraprendere questo percorso è sicuramente per ampliare il nostro bagaglio di conoscenze, soprattutto informatiche ed elettroniche, per poi comprendere meglio il significato di team planning e di team work, fondamentale per questo tipo di

progetti. Oltre a gareggiare e vincere qualche tipo di premio, vogliamo incoraggiare grandi e piccoli ad appropriarsi al mondo della robotica, fondamentale al giorno d'oggi per il mondo in cui viviamo, incoraggiandoli a creare nuove invenzioni applicabili nel campo su cui è ideata anche questa competizione, il salvataggio.

- Essendo la nostra prima esperienza, creare un progetto da subito ideale e perfettamente funzionante era impossibile. Abbiamo deciso quindi di realizzare una struttura facile da montare e poi in caso modificabile, utilizzando componenti e sensori facili da ricambiare e programmare, lasciando porte di espansione aperte sulla PCB realizzata per altri componenti inizialmente non tenuti in considerazione. Anche a livello di programmazione abbiamo cercato di creare un codice facilmente modificabile per sistemare i problemi che si possono riscontrare. È stato soprattutto grazie alle squadre del nostro istituto che hanno partecipato gli scorsi anni che abbiamo compreso tanti dei problemi fondamentali di questi robot ed è stato piu semplice quindi individuarli nel nostro robot per poi sistemarli.

- Il nostro traguardo principale è sicuramente rendere il robot il migliore in assoluto, capace di percorrere qualsiasi labirinto senza mai sbagliare un movimento o una vittima, facendolo poi nel minor tempo possibile e senza movimenti inutili. Desideriamo poi portare avanti il nome che si è creato il nostro istituto cercando di arrivare su tutti i podi possibili. Infine vogliamo essere pronti ad applicare le conoscenze acquisite durante questo progetto nel mondo del lavoro.

- La prima versione del nostro robot ci ha dato la possibilità di comprendere i vari problemi relativi alla progettazione meccanica ed elettrica. Inizialmente costruito su una base rigida, senza ammortizzatori e con un baricentro troppo elevato, abbiamo scoperto la facilità con cui il robot si bloccava negli ostacoli del labirinto, finendo per ribaltarsi con una semplice scalinata. Altro problema grave risultava la scheda elettronica che era stata creata su una comune millefori utilizzando fili rigidi e stagno per effettuare i vari collegamenti: le comunicazioni tra i vari dispositivi e sensori spesso falliva, rendendo impossibile la sua programmazione, scoprendo presto che era a causa dei continui disturbi generati sulla scheda "madre" a causa del lavoro svolto a mano. Infine, per come era stato creato l'intero robot, risultava assai difficile qualsiasi intervento di manutenzione regolare o straordinaria. Questo ci ha portati ad un radicale cambiamento, ridisegnando il robot da zero, aggiustando il baricentro e aggiungendo le sospensioni mentre per la scheda madre, è stata disegnata una PCB a software e poi stampata da ditta specializzata.



Prima versione del Robot

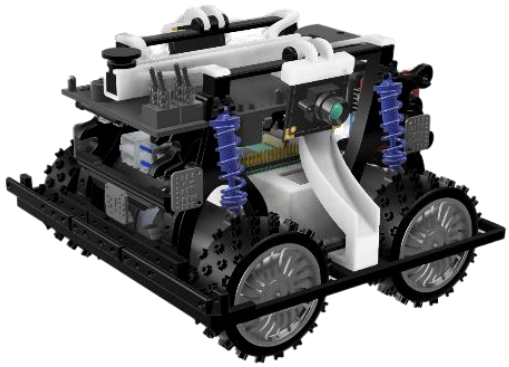
- Essendo il primo anno in cui partecipiamo a queste competizioni, non abbiamo nessuna esperienza precedente se non quelle dei nostri compagni che hanno vinto i mondiali due anni fa. Grazie a loro abbiamo avuto modo di comprendere fin dall'inizio alcuni problemi legati soprattutto alle telecamere e abbiamo anche intuito i primi passi da fare per creare un algoritmo funzionante per la mappatura del labirinto.

b. Integration Plan

- Il robot è stato inizialmente progettato via software appositi, sia per la componente fisica che elettrica. In questo modo è stato possibile gestire nel miglior modo possibile lo spazio a disposizione, garantendo anche successive espansioni hardware. Per garantire la migliore efficienza di tutto il sistema, sono stati utilizzati componenti individuali che si collegano esternamente tra di loro, in modo da garantire una immediata sostituzione in caso di guasto.
- Support your explanations with illustrations

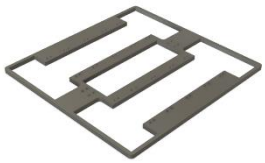
3. Hardware

- Il robot è stato realizzato fin da subito in ambiente virtuale 3D, in modo da poter studiare al meglio gli spazi a disposizione e garantire una solida struttura, dando ad ogni sensore e componente il suo spazio apposito. Abbiamo dato alla struttura una modularità per il successivo montaggio e smontaggio, per assicurare la comodità nell'accesso a tutti i sensori e per una ulteriore aggiunta di componenti inizialmente non previsti. La struttura è composta da una base e 4 colonne portanti che reggono il peso e mantengono l'equilibrio dell'intero sistema. Per garantire la modularità citata prima, la maggior parte dei pezzi sono di piccole dimensioni, facilmente smontabili senza interferire su altri componenti già installati. La dimensione di questi pezzi è stata limitata anche dal metodo con cui vengono ricavati: creati in PLA con l'utilizzo della stampante 3D, volevamo sprecare meno materiale possibile mentre lo rendevamo il più efficiente. Per il miglior funzionamento di questo sistema, anche i vari meccanismi che la compongono, per esempio il sistema di sospensioni ed il rilascio dei kit cubetti, sono stati ideati per essere il piu semplici possibili, garantendo comunque un funzionamento alla pari dei meccanismi più complessi.



a. Mechanical Design and Manufacturing

- BASE: è il pezzo più grande del robot, stampato in PLA grande 20x20cm e spessa 5mm, aiuta per l'aggancio dei motori e dei vari componenti del robot. Ideata per garantire anche la protezione delle ruote, che le circonda con il suo bordo esterno, aiutando così il robot a non bloccarsi o agganciarsi ad oggetti esterni.

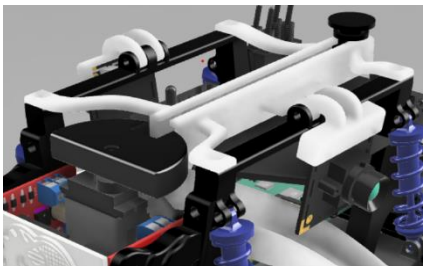


- SOSPENSIONI: come menzionato inizialmente, con la prima versione del robot abbiamo riscontrato problemi nel superare gli ostacoli, a causa della rigidità del robot. Per questo con la seconda versione abbiamo subito deciso di introdurre un sistema di ammortizzazione per ogni ruota presente. Queste sospensioni vengono agganciate tra l'estremità alta del supporto che avvolge i motori (che a loro volta sono agganciati alla base attraverso dei cuscinetti per dare loro mobilità) e la cima della colonna portante. In normale regime, cioè quando il robot è poggiato su terreno pianeggiante, gli ammortizzatori sono già in fase di compressione, cosi in caso di superamento di un ostacolo, con la conseguente elevazione di una delle ruote, nessun'altra rischierà di girare a vuoto in quanto con la massima estensione della molla riuscirà a toccare terra. Questo garantisce un superamento omogeneo di qualsiasi ostacolo ed una migliore distribuzione delle forze all'interno del robot.

- MOTORI: Il nostro robot utilizza 4 motori DC a 12 Volt JGB37-520 con motoriduttore che porta ciascuno di loro a 110rpm. Sono equipaggiati con encoder incrementali che contano le rotazioni di ogni motore, permettendoci di ricavare la singola velocità e con l'utilizzo del sistema PID si riesce a mantenere identica la velocità tra di loro. Questi motori sono gestiti attraverso due driver motore L298N che ci permettono di inviare due segnali digitali per indicare il verso di marcia ed un segnale PWM per la velocità in bassa tensione, convertendolo poi nella tensione che viene applicata ai capi del motore. Inoltre, sul robot è presente un servomotore a 5 Volt che gestito tramite segnale PWM, che permette lo spostamento dei cubetti sul corretto scivolo in fase di rilascio.

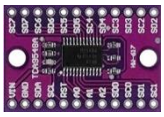
Driver L298N "ponte ad H"

- RILASCIO KIT CUBETTI: Per il rilascio dei kit una volta trovata una vittima, abbiamo ideato un meccanismo semplice in modo che non vi siano mai inceppi nella fase di rilascio. Esso infatti è composto da una canaletta orizzontale posta sopra il robot, che immagazzina i 12 cubetti e subito dopo di loro un tappo a cui viene agganciato un elastico che spinge questi cubetti fino allo slot del servo motore, mantenendo così questo slot sempre caricato grazie alla forza dell'elastico. In fase di rilascio, il servomotore sposta il cubetto sullo scivolo del lato interessato facilitando così la loro caduta. Anche i cubetti a loro volta sono stati studiati: i lati di ognuno di loro è stato smussato per diminuire l'effetto del rimbalzo garantendo così la sua caduta nell'area della vittima.

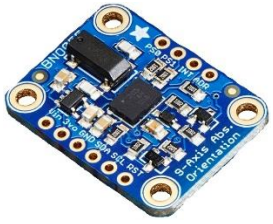


b. Electronic Design and Manufacturing

- SENSORI: Abbiamo deciso di montare vari sensori sul robot, in modo da avere una totale conoscenza dell'ambiente circostante aiutando così il suo orientamento e rilevamento di ostacoli:
  - VL6180X: abbiamo montato 6 di questi sensori, uno davanti e dietro e due per i lati. Sono sensori di prossimità laser e rilevano una distanza fino a 255mm che è più che sufficiente per il nostro scopo e sono collegati attraverso un multiplexer I2C TCA9548A alla rete I2C del Raspberry Pi 4. Abbiamo scelto questi sensori perché hanno una alta precisione, non si lasciano influenzare da fattori esterni come la temperatura e soprattutto sono molto comodi da programmare e gestire.

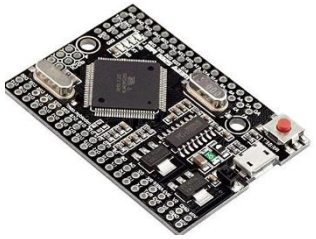


  - BNO055: è giroscopio triassiale, accelerometro con magnetometro a 9 assi, utilizzato nella modalità IMU (Inertial Measurement Unit) che calcola l'orientamento nello spazio, come angoli di Eulero, partendo dai dati di accelerazione del giroscopio e dell'accelerometro. Abbiamo scelto questo giroscopio 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, svolgiamo perfette virate di 90 gradi e rileviamo salite e discese.

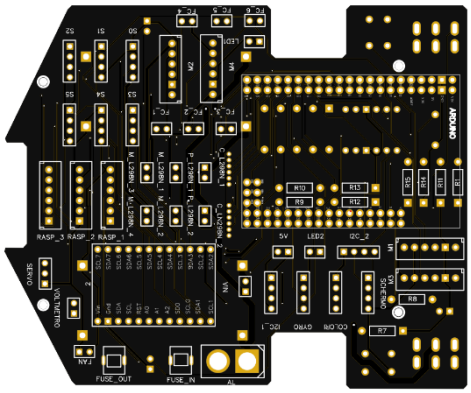


- APDS9960: è il sensore di colori che ci permette di distinguere le placche blu, nere e argento per scegliere i movimenti che deve poi svolgere il robot. Comunica attraverso il protocollo I2C con il Raspberry ed è stato scelto dopo aver provato una vastità di sensori, in quanto è il più veloce a rilevare il corretto valore e non subisce troppe variazioni con diversa illuminazione, rendendolo cosi la scelta più sicura da intraprendere.

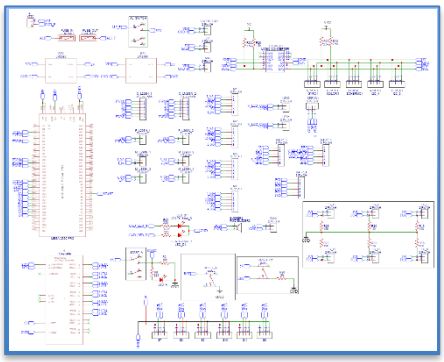
- MICROPROCESSORE/MICROCONTROLLORE: il robot è equipaggiato con due tipi di schede. Il primo è il microprocessore Raspberry Pi 4, che ha il compito di leggere e gestire tutta la logica dei sensori, rilevare le vittime attraverso le telecamere poste ai lati ed inviare attraverso la comunicazione I2C i comandi per muovere il robot. È stato scelto per la sua grandissima capacità di elaborazione dati, gestione dei segnali video e nell'interfacciarsi con sensori di ogni tipo. Per la gestione dei movimenti, quindi per il controllo dei motori, è stato utilizzato un Arduino Mega 2560 Pro che esegue il controllo della velocità attraverso il sistema PID. È stato scelto per la quantità e stabilità dei segnali PWM facilitando così la loro programmazione e per la dimensione ridotta facilitando così la sua installazione. Come menzionato precedentemente, le due schede comunicano attraverso il protocollo I2C, dove Il Raspberry agisce da Master mentre Arduino da Slave, creando una serie di comandi e funzioni necessarie per i movimenti del robot.



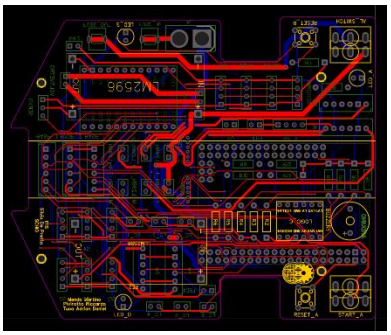
- PCB: Per il collegamento di tutti i sensori e delle due schede, abbiamo progettato una "scheda madre" mandata poi a stampare da azienda specializzata. È stata realizzata in modo da occupare nella maniera più efficiente lo spazio ristretto che avevamo a disposizione ed essere la più sicura dal punto di vista elettrico. Infatti è composta da svariati connettori JST-HX per collegare tutti i sensori individualmente mentre i componenti che sono direttamente collegati sulla PCB sono facilmente rimovibili e sostituibili in caso di guasto, non dovendo cambiare tutta la scheda madre. La PCB è spessa 1,6mm ed è composta da due strati, che contengono tutti i tipi di tracciati necessari al robot, dai circuiti di alimentazione dei motori fino alla comunicazione di segnali. Come precedentemente spiegato nel paragrafo 2.a con la iniziale scheda stagnata manualmente con fili rigidi, risultava difficile la comunicazione tra le due schede ipotizzando come causa le interferenze tra segnali a bassa tensione ed il circuito di potenza che alimentava i motori. Per questo è stata fatta particolare attenzione con la nuova PCB per mantenere separati questi tipi di circuiti, confermando la nostra ipotesi, in quanto non si sono più verificate nessun tipo di interferenze.



- ALIMENTAZIONE: Tutto il circuito è alimentato da una batteria LiPo 14.8V 2300mAh, interrotta attraverso un interruttore principale per poi collegarsi direttamente ai driver dei motori ed ai due buck-boost LM2596, che abbassano la tensione da 5V per alimentare tutti i sensori e le due schede. I motori sono quindi alimentati alla tensione della batteria ma grazie al controllore PID essi non vengono più influenzati superando una certa soglia di tensione. È presente un secondo interruttore sulla PCB, che funge



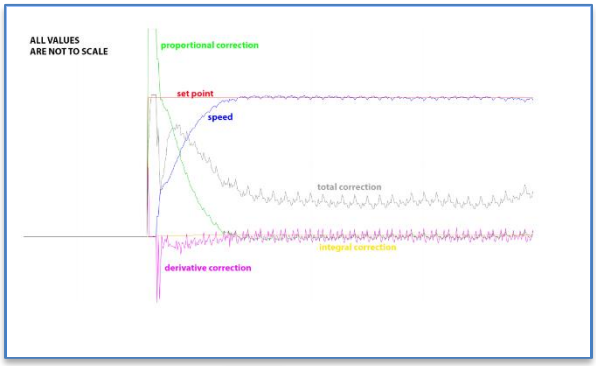
da avvio del codice di Arduino, senza mai spegnerlo, in questo modo si può continuare a mantenere tutto il circuito acceso senza che i motori inizino a muovere il robot. Infine tutto il circuito è protetto con un fusibile da 6A collegato ancora prima dell'interruttore generale.



4. Software

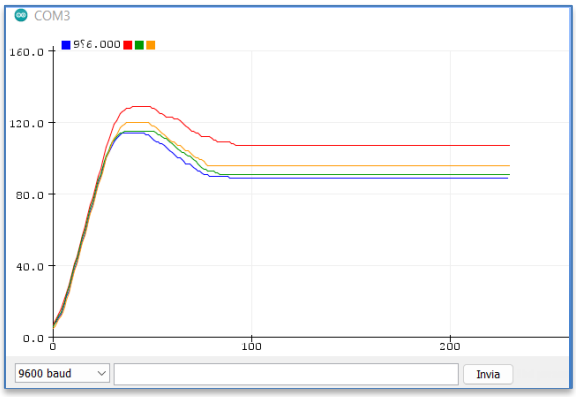
a. General software architecture

- Il software del robot è suddiviso in due parti distinte, una presente su Arduino che gestisce i motori attraverso il controllo PID e l'altra sul Raspberry che gestisce tutta la logica. I due software comunicano con il protocollo I2C come precedentemente citato.



Calcolo PID su singolo motore

- ARDUINO: è programmato interamente in C attraverso l'applicazione [Microchip Studio IDE] per la programmazione di AVR, utilizzato per avere una migliore gestione delle porte accessibili e dei registri del microcontrollore. Viene utilizzato esclusivamente per gestire i 4 motori presenti sul robot per i suoi movimenti che necessitano di un controllo della velocità per garantire un migliore controllo degli spostamenti. Per questo abbiamo utilizzato dei motori con montati degli encoder utili alla misurazione della velocità per regolarla in modo che sia sincrona tra tutti i motori. Per realizzare questa sincronizzazione abbiamo utilizzato il controllo Proporzionale-Integrale-Derivato (PID) via software, che permette di mantenere costante la velocità desiderata. I 4 motori sono indipendenti e per ognuno viene calcolato il proprio valore con il controllo. Il PID è un controllo a retroazione negativa che agisce da rete correttrice, 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.



Curva velocità dei 4 motori

- RASPBERRY PI: Il codice principale del robot è strutturato su una moltitudine di librerie e codici secondari contenenti le classi (movimenti, servomotore, telecamere, finecorsa, mappatura) che confluiscono in un unico codice principale che permettere di eseguirli in parallelo grazie ad "asyncio" e "multiprocessing" e attraverso le queue, questi processi paralleli riescono a comunicare tra di loro. Per esempio, la classe "movimenti", che comprende tutto il necessario per muovere in autonomia il robot, dalla lettura dei sensori fino all'invio dei dati ad Arduino, attende un risultato positivo da parte delle telecamere quando rilevano una vittima per potersi fermare in tempo.

- COMUNICAZIONE I2C: per qualsiasi tipo di comunicazione del robot è stato utilizzato il protocollo I2C, configurandolo con successo anche su Arduino, in modo da avere un sistema omogeneo di comunicazione tra sensori e schede.

- MOVIMENTI: I movimenti del robot sono stati progettati per essere delle funzioni della logica principale richiamabili quando se ne ha necessità. Nel codice di Arduino sono state create delle funzioni base come avanti, destra, sinistra, fermo, etc. Queste funzioni sono richiamabili dal Raspberry all'invio di un determinato byte (es. Avanti == 0x01, Retromarcia == 0x09) e alla loro fine rinviano un byte al master per segnalare il loro completamento.

- TELECAMERE*:* Per il rilevamento delle vittime all'interno della mappa abbiamo utilizzato due telecamere USB100W04H con obiettivo grandangolare, poste una sul fianco destro e l'altra su quello sinistro della macchina. Collegando quindi queste al raspberry pi e usando la libreria opencv, abbiamo potuto catturare le immagini, analizzarle, e rilevare eventuali colori o lettere. L'obiettivo a 180° è stato scelto per cercare di vedere il più possibile all'interno del labirinto, tuttavia esso presenta un elevato effetto fisheye, il quale storcendo la forma delle vittime trovate, ne rende difficile il riconoscimento. Per questo motivo abbiamo deciso di calcolare delle matrici da applicare all'immagine catturata, così da 'raddrizzarla' e migliorare l'analisi. Per ottenere queste matrici di correzione è stato necessario scattare una trentina di foto di una scacchiera in bianco e nero, rilevata attraverso la funzione "cv2.findChessboardCorners". Successivamente, utilizzando un'altra funzione della libreria OpenCv, "cv2.fisheye.calibrate", abbiamo potuto calcolare le correzioni necessarie per rendere l'immagine ortogonale, ovvero con tutte le linee parallele e perpendicolari fra loro. Infine per mezzo delle funzioni "cv2.fisheye.initUndistortRectifyMap" e "cv2.remap()", applichiamo queste matrici ad ogni immagine catturata dalle telecamere.

Immagine distorta (originale)

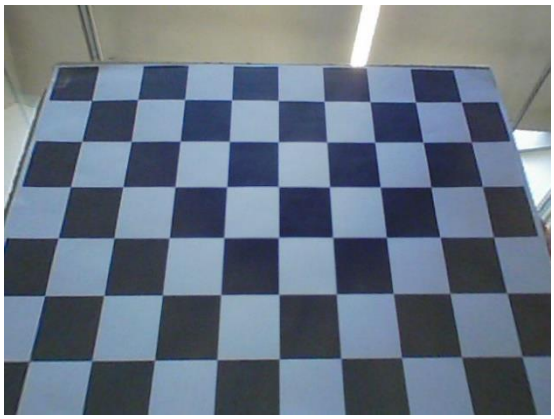


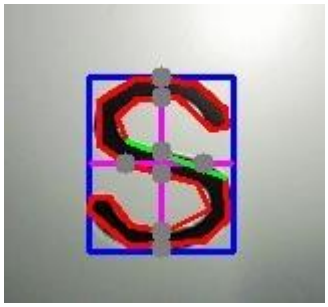
Immagine raddrizzata (ricalcolata)

- RILEVAMNETO VITTIME COLORATE: Ottenuta quindi l'immagine 'raddrizzata', è necessario verificare se sono presenti vittime colorate o lettere. Per quanto riguarda le prime, l'immagine viene inizialmente convertita dal RGB al HSV (Hue, Saturation, Value). Questo spazio colore descrive i colori (hue) in termini di tonalità (saturazione o quantità di grigio) e di luminosità, dunque per ottenere un risultato il più preciso possibile, sono stati applicati dei led bianchi. In seguito, attraverso delle soglie realizzate separatamente e la funzione "cv2.inRange", viene svolta una binarizzazione dell'immagine, trasformando i pixel all'interno delle soglie in bianco e i restanti in nero. Abbiamo realizzato una soglia per ogni colore da rilevare (rosso, giallo e verde), la quale è composta da due colori hsv, uno minimo e uno massimo, per cui avremo tre immagini binarizzate. Successivamente, dopo aver effettuato una leggera sfocatura a quest'ultima volta, utilizzare la funzione "cv2.findContours" in modo tale da ottenere una lista con i vertici dei contorni rilevati.

Infine calcoliamo l'area per ogni contorno, e se questa è maggiore di un determinato valore (così da evitare riconoscimenti errati), viene identificato il colore in funzione dell'immagine binarizzata in analisi.

- RILEVAMENTO LETTERE: Per le lettere invece l'immagine 'raddrizzata' viene convertita in tonalità di grigi e sfuocata, per poi applicare la funzione “cv2.Canny”, la quale svolge una binarizzazione dell'immagine, convertendo i pixel dei contorni delle figure in bianco e i restanti in nero. In seguito ci ricaviamo i vertici di questi contorni sempre attraverso la funzione “cv2.findContours” e calcoliamo l'area per ognuno. Se una di queste è compresa tra due determinati valori, indicando perciò la probabile presenza di una lettera, il contorno associato viene salvato. Mediante quindi quest'ultimo usiamo la funzione “cv2.boundingRect” per costruire un rettangolo intorno alla lettera e le sue direttrici, e la funzione “cv2.approxPolyDP” per realizzare un poligono che riprenda la forma della lettera. Successivamente andiamo a trovare il lato più lungo di questo poligono, calcolando il suo angolo di inclinazione e la differenza tra il lato verticale del rettangolo ed esso. Attraverso dunque questi due valori siamo in grado di ipotizzare quale possa essere la lettera presente. A questo punto andiamo ad analizzare il numero e la posizione dei punti di cambio colore nell'intersezione tra le direttrici e la lettera. Anche in questo caso, come nel passaggio precedente, ogni lettera avrà le proprie caratteristiche, le quali variano limitatamente. Perciò andando proprio a 'sfruttare' le differenze tra una vittima e l'altra, siamo in grado di identificare con sufficiente precisione la lettera rilevata.

Funzione “Canny”



Analisi finale

b. Innovative solutions

- Sono state sperimentate numerose soluzioni, anche poco sicure e pratiche che hanno portato a diversi risultati, aiutandoci ad arrivare dove siamo ora, soprattutto nella programmazione dove abbiamo scoperto solo mentre scrivevamo i codici come funziona veramente un linguaggio di programmazione.
- Punto cardine del nostro progetto è cercare la soluzione piu semplice e di conseguenza piu innovativa per svolgere il compito richiesto: i sensori di distanza VL6180X sono stati collegati attraverso un multiplexer I2C per poter raggirare con ingegno il problema degli indirizzi dei sensori, che non erano facilmente modificabili.
- Anche con le telecamere sono state cercate soluzioni che possano facilitare il carico di lavoro e una di queste è sicuramente la funzione che permette di correggere l’immagine distorta dall’obbiettivo fisheye.

5. Performance evaluation

- A causa del poco tempo avuto a disposizione per la creazione di questo robot (poco più di 6 mesi), non abbiamo avuto modo di scoprire a fondo e risolvere tutti i problemi esistenti, infatti è carente sotto molti punti di vista ma riesce comunque a svolgere il suo compito con minimi errori, riuscendo sempre a concludere la sua gara. Siamo già al lavoro per sistemare questi problemi e rendere il nostro robot ancora più performante di tutti i robot passati e siamo pronti a guadagnarci il titolo di campioni del mondo nel minor tempo possibile.

6. Conclusion

- In questa relazione non siamo riusciti a documentare con precisione tutto il lavoro, e perciò a rispondere esaustivamente a tutte le vostre domane. 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 pagina 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 per source code:**
  https://github.com/Danituca2154/Sensa-Schei

- **EasyEda PCB schema e tracce:**
  https://oshwlab.com/danituca2154/main-pcb_copy