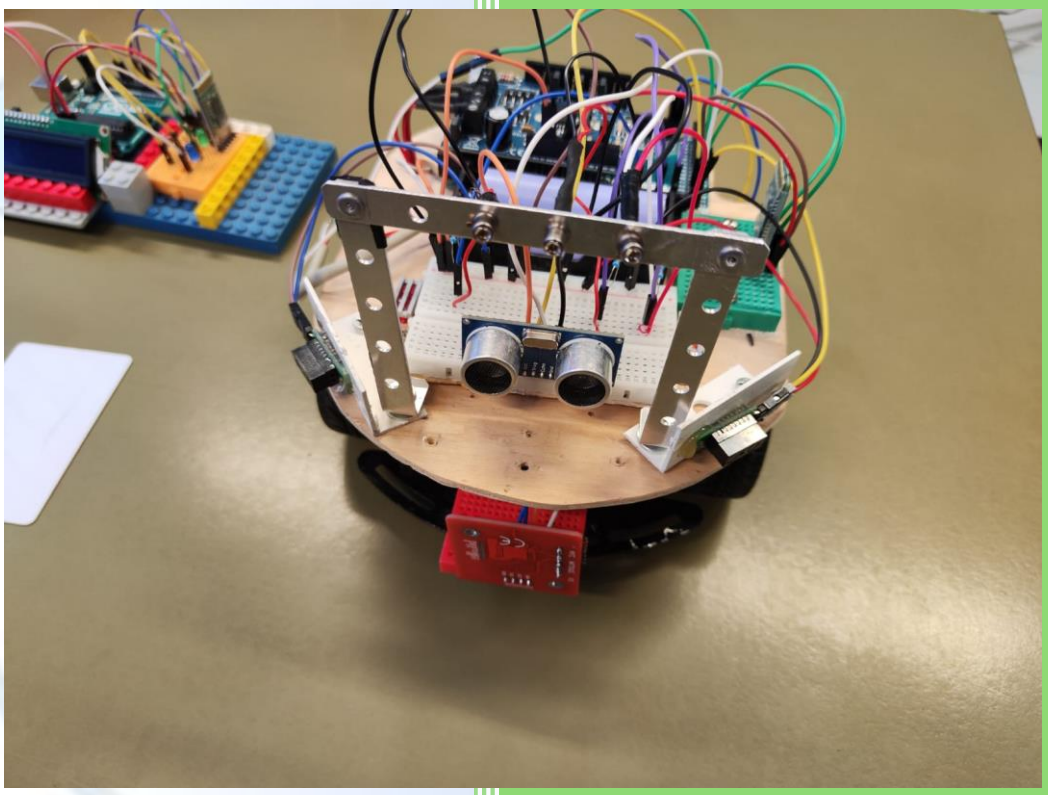
2024/2025

“ROBOTICA”



Luigi Zhou, Simone Vidotti, Grecu Lorenzo
I.S.I.S Arturo Malignani
18/10/2024
Fine:

Sommario

TESTO ORIGINALE

Progettare e realizzare un robot, partendo dalla struttura meccanica di un robot già presente in laboratorio, che dovrà essere capace:

- di muoversi autonomamente alla ricerca di sorgenti luminose collocate in posizioni fisse e sconosciute all'interno di un campo di movimento di dimensioni assegnate;
- di evitare gli ostacoli collocati in posizioni fisse e sconosciute all'interno del campo di movimento;
- di individuare tracciati segnati sul piano di movimento, e di seguire tali tracce (facoltativo);
- di rilevare la presenza di dispositivi TAG NFC, collocati in posizioni fisse in corrispondenza delle sorgenti luminose, e di leggere informazioni da tali dispositivi;
- di comunicare ad un sistema di controllo remoto, mediante dispositivi bluetooth, le informazioni
- sui TAG individuati, al fine di permettere lo spegnimento delle fonti di luce corrispondenti.

Il robot dovrà individuare correttamente tutti gli obiettivi (fonti luminose), leggere il codice del TAG NFC corrispondente a ciascuna sorgente luminosa ed inviarlo ad un dispositivo di controllo che provvederà, nel caso di codice corretto, a spegnere la luce corrispondente. Al termine della prova tutte le fonti luminose associate ai relativi codici dovranno risultare spente. La prova si considera conclusa quando tutte le luci associate ai codici dei TAG sono state correttamente rilevate e conseguentemente spente.

Per la valutazione del progetto sarà data particolare importanza:

- alla corretta individuazione delle sorgenti luminose;
- all'individuazione del maggior numero di sorgenti luminose;
- alla capacità di seguire senza incertezze le tracce indicate (facoltativo);
- alla precisione e fluidità del movimento del robot;
- alla corretta lettura e conseguente invio del dato letto al dispositivo remoto di controllo delle luci;
- allo studio di adeguati comportamenti ed efficaci strategie di controllo nel movimento del robot;
- all'efficacia del controllo, da parte del dispositivo remoto, delle fonti luminose.

Per lo sviluppo del progetto si considerino i seguenti requisiti:

Robot

- Il robot deve poter essere interamente contenuto all'interno di un parallelepipedo di dimensioni 190 x 190 x 160 millimetri. Da queste misure sono esclusi eventuali dispositivi di contatto ("baffi") per l'individuazione di ostacoli.

- Il robot deve essere autonomo: non può essere direttamente radiocomandato o filoguidato da un operatore esterno.

Campo di Movimento

- Il campo di movimento è formato da una superficie piana di dimensioni pari a 1,5 x 1,5 metri, di colore bianco.
- Il perimetro del campo di movimento è delimitato da un bordo, di colore bianco, di altezza pari a 260 millimetri.
- All'interno del campo di movimento sono presenti ostacoli di colore bianco a forma di parallelepipedo le cui dimensioni sono pari a 200 x 200 x 260 millimetri.
- Gli ostacoli presenti nel campo di movimento possono essere disposti in modo da formare dei corridoi di larghezza non inferiore a 300 millimetri.
- Sono considerati ostacoli, a tutti gli effetti, anche i bordi del campo di movimento.

Tracciato evidenziato

- Sul piano di movimento, in corrispondenza di ogni fonte luminosa, sarà disposta una traccia nera, di 25 mm di larghezza, che partirà dalla base ove risulta collocata la luce e procederà perpendicolarmente verso la parte opposta alla luce, per un tratto di circa 40 cm. Lo scopo di tale traccia sarà quello di agevolare il movimento del robot al fine di posizionarsi correttamente difronte alla fonte luminosa (il riconoscimento della traccia ed il suo inseguimento sono facoltativi).

Obiettivi (fonti luminose e TAG NFC)

- Gli obiettivi che il robot deve rilevare sono costituiti da:
  - Sorgenti luminose direzionali poste ad un'altezza di 160 millimetri dal piano di movimento e fissate agli ostacoli (bordi del campo inclusi) senza nessun tipo di sporgenza (sono inglobate negli ostacoli e la luce esce da un foro di circa 20 mm di diametro.
  - TAG NFC disposti ciascuno alla base dell’alloggiamento di ciascuna luce LED, ad una altezza di 35 mm dal piano.

Dispositivo di controllo delle fonti luminose

- Il dispositivo di controllo sarà basato sull’uso di una scheda Arduino UNO.
- Impiegherà un dispositivo BT per la comunicazione con il robot.
- Sarà equipaggiato con un display LCD a due righe di 16 caratteri per la visualizzazione dei codici inviati da e verso il robot.
- Presenterà una serie di linee digitali per il controllo delle fonti luminose presenti nell’area di movimento del robot.
- Controllerà un LED per l’indicazione di “termine attività”.
- Funzionamento:
  - Il dispositivo sarà normalmente predisposto alla ricezione dei dati provenienti dal robot.

- Ogni fonte luminosa sarà associata ad un corrispondente codice identificativo.
- Ogni dato acquisito, proveniente dal robot, sarà visualizzato sul display LCD e quindi confrontato con il codice identificativo relativo ad ogni fonte luminosa. Si verificheranno due condizioni:
    1. Il confronto è andato a buon fine: il dispositivo provvederà allo spegnimento della relativa fonte luminosa e all'invio al robot di un opportuno codice di risposta, che potrà assumere due valori, in base allo stato delle luci:
        - Luci parzialmente spente: il valore indicherà al robot che potrà riprendere il movimento alla ricerca di altre fonti luminose associate a codici validi.
        - Luci tutte spente: il valore indicherà che l'attività di ricerca si considera conclusa.
    2. Se, al contrario, il codice rilevato non corrisponde ad un codice valido (riconosciuto), il controllore provvederà ad inviare immediatamente al robot una risposta di codice non corretto, e questo potrà riprendere la ricerca di ulteriori fonti luminose.

    Complessivamente, quindi, i codici di risposta del dispositivo di controllo al robot saranno tre. Sul display dovranno essere visualizzati, in particolare:

    - il codice del tag proveniente dal robot
    - il codice di risposta verso il robot.

    Tali codici saranno visualizzati fino al successivo aggiornamento.
- E' importante che il robot sia programmato all'attività di ricerca di un numero non definito di fonti luminose, ovvero, soltanto il dispositivo di controllo potrà notificargli la fine della ricerca.
- Soltanto quando tutte le luci all'interno dell'area di movimento saranno spente il controllore accenderà un LED ad indicare che l'attività sarà conclusa.

Comportamento del Robot

- Il robot viene posizionato sul piano di movimento in un punto che verrà definito solo in fase di collaudo (pertanto da considerarsi casuale).
- Nel momento iniziale il robot deve compiere una rotazione con un angolo casuale (anche multi-giri).
- Il robot inizia il movimento alla ricerca di sorgenti luminose.
- Quando il robot individua una fonte luminosa dovrà orientarsi e procedere verso questa. Per facilitare l'avvicinamento del robot alla fonte luminosa, e quindi al tag NFC posto al di sotto di questa, il robot incontra una traccia nera che conduce perpendicolarmente al TAG stesso.

- Il robot eseguirà la lettura del TAG NFC ed invierà l'UID del TAG al dispositivo di controllo.
- Se il dato inviato risulta corretto il dispositivo di controllo provvederà allo spegnimento della luce individuata dal robot ed invierà a questo una notifica con un codice il cui valore indicherà il proseguimento della ricerca di altre fonti luminose. Il robot potrà quindi continuare il suo movimento di ricerca.
- Se la fonte luminosa spenta sarà l'ultima, il robot riceverà dal dispositivo di controllo tale notifica con un opportuno codice e quindi considererà conclusa la sua attività: il robot arretrerà di qualche centimetro, eseguirà una rotazione di 540° (360° + 180°), quindi si arresterà.

Tempi di progetto

progetto si riterrà concluso, salvo eventuali necessarie proroghe concordate, venerdì 20 dicembre, e sarà collaudato la lezione successiva.

ANALISI DEL PROBLEMMA

Il progetto prevede la costruzione di un robot autonomo in grado di muoversi in un ambiente delimitato e svolgere una serie di compiti complessi. L'obiettivo principale del robot è quello di individuare e spegnere delle sorgenti luminose collegate a dispositivi "TAG NFC", inviare i dati relativi al sistema di controllo tramite Bluetooth e terminare l'attività una volta completata l'operazione su tutte le luci presenti.

Compiti principali del robot:

1. Rilevamento delle sorgenti luminose:

Il robot deve individuare luci poste in posizioni sconosciute e fisse all'interno del campo di movimento. La luce emessa è direzionale e la sorgente è montata a un'altezza definita dal piano di movimento (160 mm).

- **Sensori di prossimità e ricerca delle fonti luminose**

  - **Sensore di luce**: Usa un sensore di luminosità come un fototransistor o una LDR (resistenza dipendente dalla luce) per rilevare le sorgenti luminose.
  - **Inseguimento di traccia nera (opzionale)**: Puoi usare un sensore a infrarossi o fotodiodi per seguire la traccia nera che facilita l'avvicinamento alla fonte luminosa.

2. Lettura di TAG NFC:

Ogni fonte luminosa ha associato un TAG NFC posizionato a 35 mm dal piano. Il robot deve avvicinarsi alla luce, leggere il codice univoco del TAG e inviarlo al sistema di controllo remoto.

- **Sistema di lettura TAG NFC**

  - Colloca un lettore NFC sul robot per leggere i TAG alla base delle sorgenti luminose.
  - **UID del TAG**: Ogni TAG avrà un codice identificativo unico. Una volta letto, il robot invierà l'UID al sistema di controllo tramite Bluetooth.

3. Evita ostacoli:

Nel campo di movimento sono presenti ostacoli (di colore bianco) disposti in maniera fissa ma sconosciuta. Il robot deve essere in grado di rilevarli e aggirarli per continuare la sua esplorazione.

4. Comunicazione con il sistema di controllo remoto:

Tramite un modulo Bluetooth, il robot deve trasmettere al sistema di controllo i codici dei TAG NFC rilevati. In base alla correttezza dei codici, il sistema provvede allo spegnimento delle fonti luminose.

- **Modulo Bluetooth**: Utilizza un modulo HC-05 o HC-06 per la comunicazione bidirezionale con la scheda Arduino.
- **Arduino**: Programma Arduino per ricevere i dati dal robot, confrontarli con i codici delle sorgenti luminose e spegnere le luci se il codice è corretto.
- **Display LCD**: Utilizza un LCD a due righe per visualizzare i codici dei TAG ricevuti e l'esito del confronto.

5. Ricerca autonoma:

Il robot deve agire in modo completamente autonomo, senza interventi esterni, gestendo tutte le operazioni (movimento, evitamento ostacoli, rilevamento luci, lettura dei TAG e comunicazione dei dati) in autonomia.

Requisiti del robot

- Movimento autonomo: Il robot non può essere radiocomandato. Deve gestire in autonomia tutti gli aspetti della navigazione, dall'orientamento alla ricerca delle luci.

- Dimensioni: Il robot deve rispettare le dimensioni massime di 190 x 190 x 160 mm (esclusi eventuali dispositivi di contatto).

- Ambiente di lavoro:

1. Campo di movimento: 1,5 x 1,5 metri.

2.Ostacoli di 200 x 200 x 260 mm.

3.Larghezza minima dei corridoi: 300 mm.

4. Presenza di tracce nere facoltative che facilitano il movimento verso le fonti luminose.

Funzionalità facoltative

- Inseguimento di traccia nera: Se implementato, il robot può usare un sensore per seguire una traccia nera larga 25 mm che facilita il movimento verso le luci.

Vincoli e risorse

1. Spazio fisico limitato: Il campo di movimento è relativamente piccolo, il che richiede movimenti precisi e una gestione ottimizzata dello spazio disponibile.

2. Numero sconosciuto di sorgenti luminose e TAG: Il robot deve essere programmato per gestire una quantità indeterminata di obiettivi, continuando la ricerca fino a quando il sistema remoto non conferma la fine dell'attività.

3. Interferenze visive: Gli ostacoli e i bordi del campo di movimento sono dello stesso colore (bianco), rendendo importante che i sensori siano in grado di distinguere correttamente questi elementi.

4. Interazione remota con Arduino: Il sistema di controllo è basato su Arduino UNO, che riceve i dati inviati dal robot, verifica la correttezza dei TAG NFC e spegne le luci. Il robot deve inviare correttamente i codici e ricevere i feedback dallo stesso sistema per proseguire con l'attività.

Sfide tecniche principali

- Navigazione e gestione degli ostacoli:

Il robot deve essere in grado di evitare ostacoli in modo autonomo utilizzando sensori di prossimità (IR o a ultrasuoni) e deve essere abbastanza preciso da evitare urti e blocchi, soprattutto in un ambiente con corridoi stretti.

- Rilevamento e lettura delle fonti luminose:

Deve rilevare le sorgenti luminose a distanza e avvicinarsi correttamente al TAG NFC per leggerne il codice. La lettura NFC richiede un posizionamento preciso del robot per garantire la corretta acquisizione dei dati.

- Comunicazione e controllo remoto:

La comunicazione Bluetooth tra il robot e Arduino è cruciale per il corretto spegnimento delle luci. Il sistema deve essere in grado di inviare e ricevere dati in modo affidabile per garantire che la luce corretta venga spenta.

- Fluidità del movimento:

Oltre alla semplice navigazione, il robot deve muoversi in modo fluido, senza interruzioni o movimenti casuali che potrebbero influire sull'efficacia della ricerca delle luci.

Conclusioni

Per concludere con successo il progetto, il robot dovrà affrontare una serie di sfide tecniche:

- Navigazione autonoma con evitamento degli ostacoli.

- Rilevamento preciso delle sorgenti luminose e lettura dei TAG NFC.

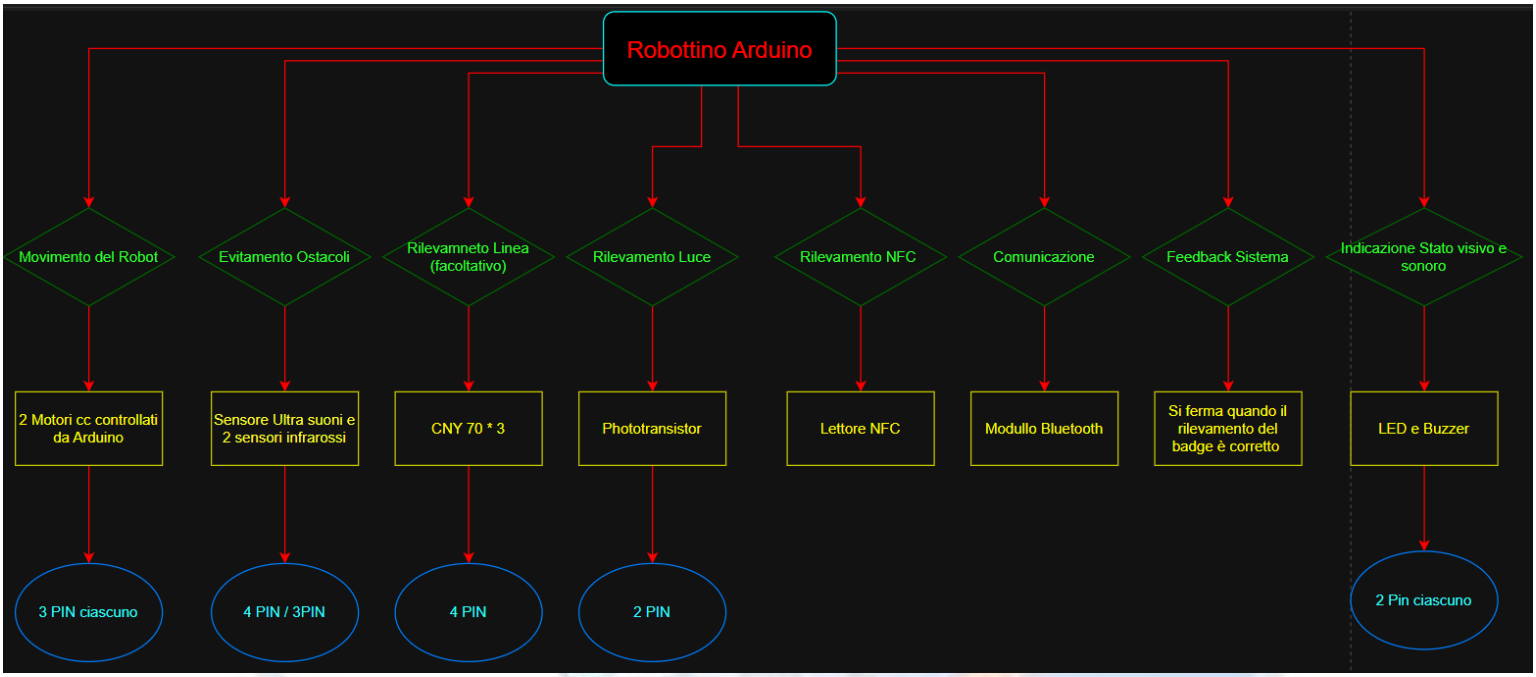
- Comunicazione efficace con il sistema remoto per spegnere correttamente le luci.

- Capacità di esplorare l'intera area di movimento, garantendo che tutte le fonti luminose siano spente al termine dell'attività.

L'accurata pianificazione della strategia di movimento, l'integrazione dei sensori e l'implementazione della logica di controllo saranno essenziali per il successo del progetto.

SCHEMA LOGICA ROBOTINO

SCHEMA A FUNZIONALE



Lo schema rappresenta le principali funzioni e i relativi componenti che il robot Arduino deve integrare per operare correttamente. Ogni funzionalità è associata a specifici sensori o moduli, con i dettagli dei collegamenti ai PIN di Arduino:

- **Movimento del Robot:** Due motori controllati da Arduino, ciascuno connesso a 3 PIN, permettono al robot di spostarsi e cambiare direzione.
- **Evitamento Ostacoli:** Un sensore a ultrasuoni (4 PIN) rileva la presenza di ostacoli e invia comandi per evitarli.
- **Rilevamento Linea:** Tre sensori a infrarossi (4 PIN totali) seguono il percorso tracciato da una linea, influenzando la velocità e direzione delle ruote.
- **Rilevamento Luce:** Un sensore a infrarossi (2 PIN) rileva l'intensità luminosa per eseguire azioni basate su condizioni ambientali.
- **Rilevamento NFC:** Un lettore NFC permette al robot di leggere tag e interagire con segnali specifici.
- **Comunicazione:** Un modulo Bluetooth consente la comunicazione remota tra il robot e altri dispositivi.
- **Feedback Sistema:** Il robot si arresta automaticamente quando rileva che tutte le luci sono spente.
- **Indicazione Stato Visivo e Sonoro:** Un LED e un buzzer, ciascuno connesso a 2 PIN, forniscono feedback visivi e sonori in base alle condizioni operative.

Questa configurazione consente al robot di seguire percorsi definiti, evitare ostacoli, reagire a condizioni esterne, e fornire interazione con l'ambiente circostante in modo autonomo ed efficiente.

(La parte del Rilevamento Linea non è stata implementata)

SCHEMA SULLA LOGICA DEL ROBOT

Inizio del Sistema

- **Avvio iniziale del robot**
  - Inizializzazione dei moduli e dei sensori:
    - Sensori di Movimento (motori, giroscopi)
    - Sensori di Prossimità (ultrasuoni o IR per evitare ostacoli)
    - Sensore NFC (per leggere TAG)
    - Sensore di Luce (per rilevare le sorgenti luminose)
  - Avvio del Modulo Bluetooth:
    - Connessione al sistema di controllo remoto (Arduino Uno)
    - Verifica della connessione
  - Diagnostica Iniziale:
    - Verifica del funzionamento dei sensori
    - Test della capacità di movimento (rotazioni e movimenti avanti ecc.)

Inizio della Navigazione

- **Ricerca e rilevamento delle sorgenti luminose**
  - Movimento autonomo all'interno del campo delimitato:
    - Scansione dell'ambiente alla ricerca di sorgenti luminose
    - Uso del sensore di luce per indentificare le direzioni delle fonti
  - Se viene **rilevata una sorgente luminosa**:
    - Orientamento verso la luce
    - **Avvicinamento alla sorgente** con movimenti precisi
      - Verifica continua della distanza per evitare collisioni
  - Se **non viene rilevata** alcuna luce:
    - **Scansione continua** dell'area di movimento
      - Seguire una possibile traccia (**FACOLTATIVO**)
      - Utilizzo di un algoritmo di esplorazione
- **Rilevamento e lettura dei TAG NFC**
  - Posizionamento preciso del robot vicino alla luce:
    - Distanza del TAG NFC: il sensore deve essere a circa 35 mm dal piano
  - **Lettura TAG NFC:**
    - Se il codice NFC viene letto correttamente:
      - Invio immediato del codice al sistema di controllo remoto via Bluetooth
      - Attesa del feedback dal sistema per confermare la validità del codice
        - Se il codice è corretto:

                - Il sistema remoto spegne la luce associata
                - Il robot si prepara a lasciare l'area
            - Se il codice è errato o la luce non viene spenta:
                - Nuovo tentativo di lettura o riposizionamento
        - Se il TAG NFC non viene letto:
            - Il robot si riposiziona in modo da avvicinarsi correttamente al TAG
            - Effettua una nuova lettura
- **Rilevamento e gestione degli ostacoli**
    - Durante il movimento, il robot utilizza sensori a ultrasuoni o infrarossi per rilevare ostacoli fissi:
        - Se **viene rilevato un ostacolo** (ad esempio blocco bianco):
            - Fermo immediato per evitare collisione
            - Attivazione del calcolo del percorso alternativo:
                - Eseguire una manovra di aggiramento dell'ostacolo
                - Continuare la scansione e la ricerca di sorgenti luminose dopo l'ostacolo
        - Se **non vengono rilevati ostacolo:**
            - Continuazione del movimento lungo la traiettoria pianificata

Comunicazione con il sistema di controllo remoto

- **Trasmissione dei dati:**
    - Il robot invia i codici dei TAG NFC al sistema remoto tramite Bluetooth:
        - Ogni volta che un TAG viene letto, il codice viene trasmesso al sistema di controllo Arduino
        - Il sistema remoto verifica il codice e:
            - Se corretto, spegne la luce associata
            - Se errato, richiede una nuova lettura
    - Feedback di conferma:
        - Il robot riceve una conferma dal sistema remoto che la luce è stata spenta
        - Se non riceve la conferma:
            - Ripete l'operazione di lettura del TAG NFC e tentare nuovamente la trasmissione
    - Registrazione dei TAG letti:
        - Il robot tiene traccia dei TAG già letti per evitare di rileggere gli stessi TAG

Ricerca Autonoma

- **Gestione dell'esplorazione del campo:**
    - Il robot esplora l'intero campo di movimento (1,5*1,5 metri):
        - Utilizza algoritmi di ricerca per coprire l'intera area
        - Evita di ripetere la scansione delle stesse aree già esplorate

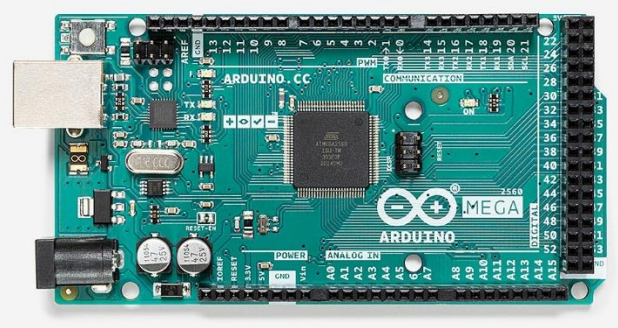
    - **Utilizzo delle tracce nere (facoltativo):**
        - Se una traccia nera è presente, il robot può seguire questa linea per facilitare la navigazione verso le sorgenti luminose
        - Se non è presente una traccia nera, il robot esplora in modo indipendente
- **Gestione dell'autonomia:**
    - Il robot continua la ricerca fino a quando non vengono rilevate tutte le sorgenti luminose e i TAG NFC
    - Il sistema remoto conferma la conclusione dell'attività quando tutte le luci sono spente
    - Se il numero di sorgenti luminose è sconosciuto, il robot continua a esplorare fiano a quando non riceve un segnale di fine attività

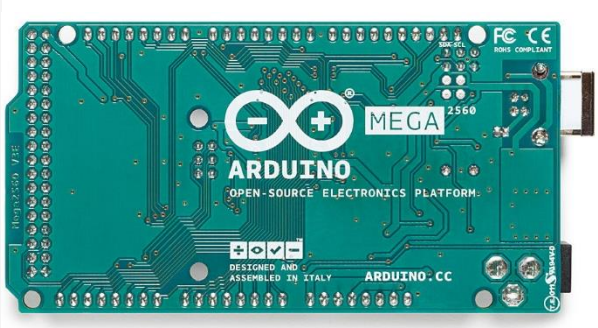
Fine della navigazione

- **Conclusione dell'attività:**
    - Quando tutte le sorgenti luminose sono state spente e il sistema remoto conferma la conclusione:
        - Il robot interrompe la ricerca
        - Ritorna alla posizione di partenza o si ferma
    - **Spegnimento dei sistemi:**
        - Spegnimento dei motori
        - Disconnessione del modulo Bluetooth
        - Spegnimento dei sensori e chiusura del sistema
    - **Report Finale:**
        - Il robot invia un report finale al sistema di controllo remoto per confermare che tutte le operazioni sono state completate correttamente

COMPONENTTI PROGETTO

Arduino Mega 2560 rev3





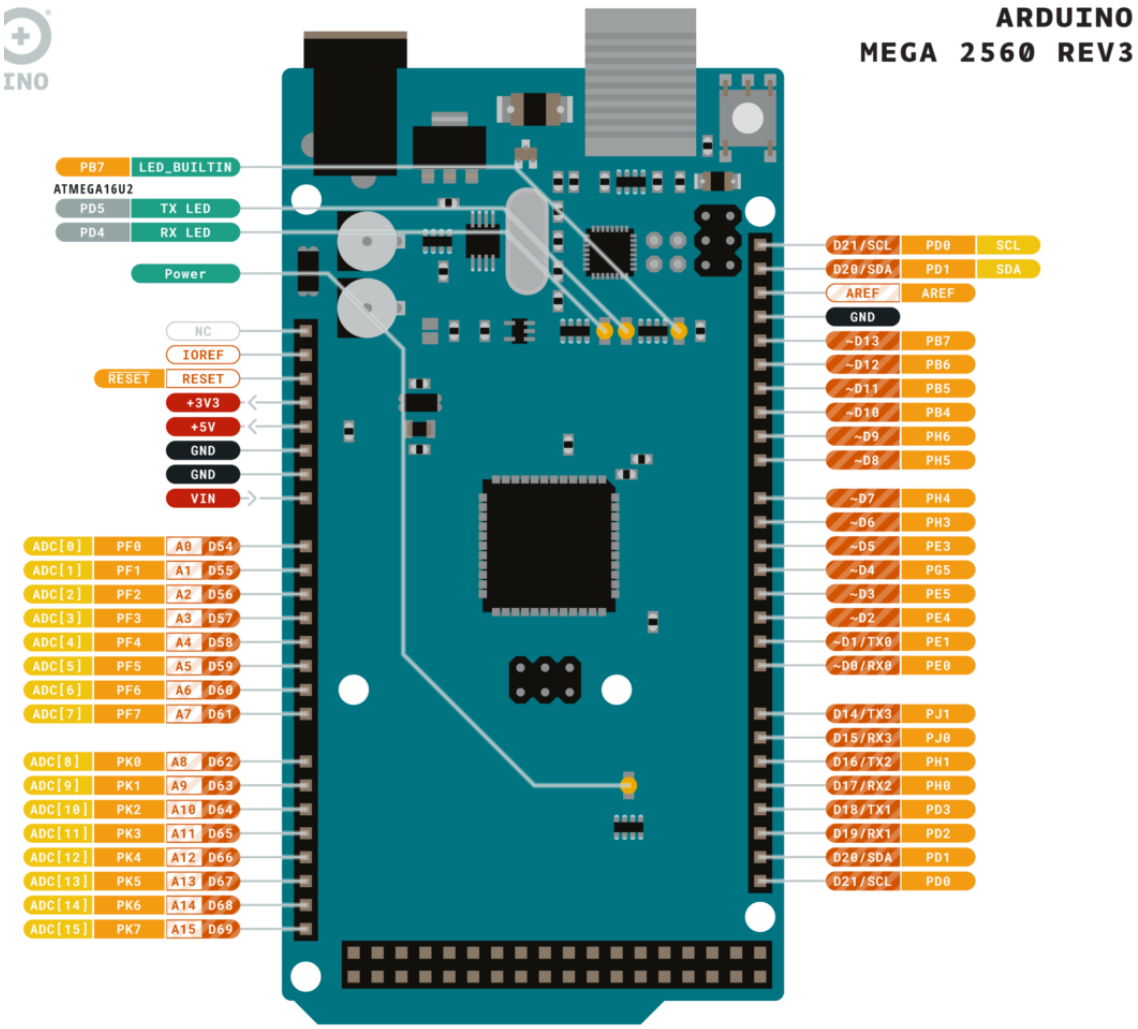
CARATERISTICHE SCHEDA:

	CARATERISTICA	VALORE
1	Microcontrollore	ATmega 2560
2	Tensione Operativa	5V
3	Input Voltage(consigliata)	7-12V
4	Input Voltage(limiti)	6-20V
5	Pin I/O digitali	54 (14 sono di uscita segnali PWM)
6	Pin imput Analogici	16
7	DC current per I/O	40mA
8	DC current per pin allimentati a 3.3V	50mA
9	Flash Memory	256KB
10	SRAM	8KB
11	EEPROM	4KB
12	Frequenza Clock	16MHz

DESCRIZIONE:

La Arduino Mega 2560 rev3 è una scheda basata sul microcontrollore ATmega2560, dotata di 54 pin di input/output digitali (14 di questi possono essere usati come segnali PWM), 16 input analogici. Fornisce tutto ciò che è necessario per supportare il funzionamento del microcontrollore, permettendo di controllare dispositivi elettronici, leggere sensori e utomatizzare processi tramite un codice programmato dall'utente.

INOUT Arduino Mega 2560:



Phototransistor

Il BPX 43 è un fototransistor al silicio NPN ad alta sensibilità progettato per rilevare la luce nell'intervallo spettrale del visibile e del vicino infrarosso, con una risposta spettrale che va da circa 400 nm a 1100 nm. Ciò lo rende ideale per applicazioni quali rilevatori ottici, interruttori fotoelettrici e sistemi di controllo a infrarossi. Il suo contenitore in plastica nera aiuta a filtrare la luce visibile e migliora la risposta agli infrarossi.

Grazie all'elevata corrente di collettore (fino a 50 mA in condizioni di illuminazione ottimali) e alla tensione operativa fino a 70 V, il BPX 43 funziona in modo affidabile in una varietà di ambienti, come telecomando, sensore di prossimità e tachimetro. La sua struttura consente tempi di risposta rapidi, il che è fondamentale per le applicazioni che richiedono un rilevamento della luce rapido e accurato.

L'angolo di rilevamento del BPX 43 lo rende adatto a specifiche direzioni della luce ed è particolarmente utile nei sistemi di misurazione e controllo, dove combina elevata sensibilità, velocità di risposta e facile integrazione nei circuiti elettronici.

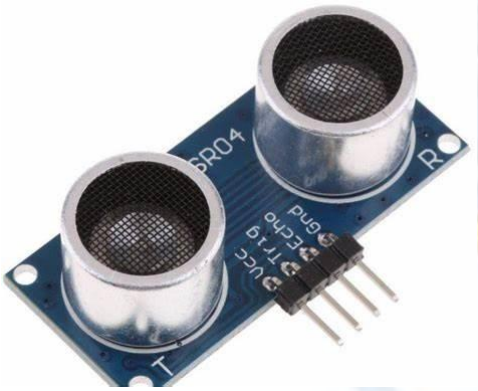
Caratteristiche:

Caratteristiche
Particolarmente adatto per applicazioni da Da 450 nm a 1100 nm
Elevata linearità
Confezione metallica ermeticamente sigillata (TO-18) con collegamento alla base adatto fino a 125 °C
Disponibile in gruppi
Applicazioni
Foto-interruttori
Elettronica industriale
Per circuiti di controllo e azionamento

Altri Modelli:

TIPO	CODICE D'ORDINE
BPX 43	Q62702-P16
BPX 43-2/3	Q62702-P3580
BPX 43-3	Q62702-P16-S3
BPX 43-3/4	Q62702-P3581
BPX 43-4	Q62702-P16-S4
BPX 43-4/5	Q62702-P3582
BPX 43-5	Q62702-P16-S5

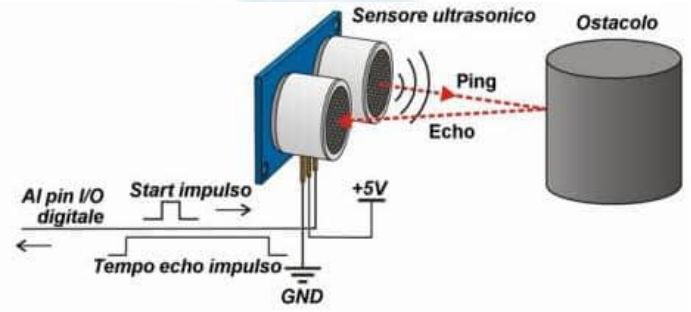
Sensore ad Ultra Suoni



Il sensore ad ultrasuoni (HC-SR04) è un dispositivo che serve a rilevare la presenza di oggetti/ostacoli nelle sue immediate vicinanze senza bisogno di contatto fisico, in particolare usa il principio del sonar.

Il HC-SR04 è dotato di un trasmettitore e di un ricevitore.

Il sensore emette impulsi sonori ultrasonici che, se riflessi da un oggetto/ostacolo, generano un burst ultrasonico (40KHz). In seguito, utilizzando il tempo impiegato dal burst per tornare, il dispositivo può determinare la presenza e la distanza di un oggetto entro il raggio di azione.

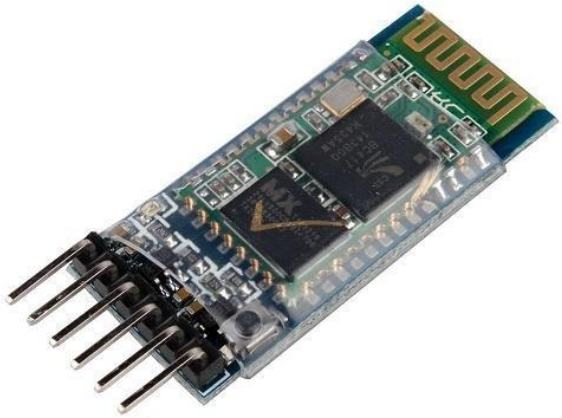


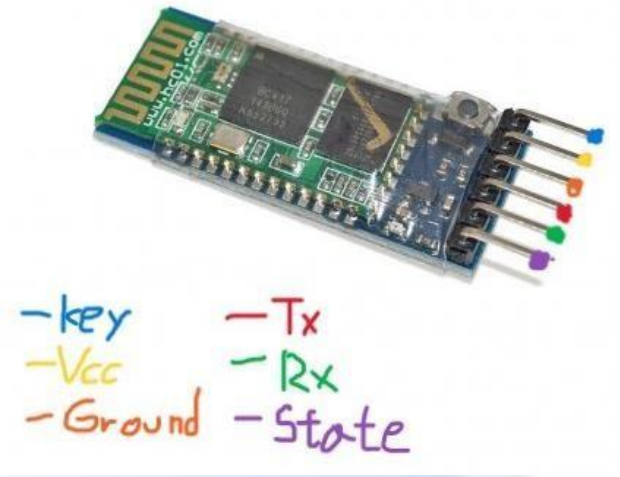
LIMITI:

- Distanza dell'oggetto superiore a quella massima ammessa dal sensore
- Angolo di incidenza del tono ultrasonico con l'oggetto troppo piccolo per permettere una sufficiente riflessione
- Oggetto troppo piccolo
- Insufficiente riflessione del suono da parte delle superfici fonoassorbenti e irregolari

CARATERISTICHE	
ALIMENTAZIONE	+5 V DC
ANGOLO DI MISURA	< 30°
DISTANZA DI RILEVAMNETO	2 cm A 40 cm
RISOLUZIONE	1 cm
FREQUENZA	40 kHz
4 PIN	
VCC	+5 V DC
TRIGGER	Genera l'impulso ultrasonico
ECHO	Rileva il segnale ultrasonico di ritorno
GND	0V Ground

HC-05 (Dispositivo Bluetooth)



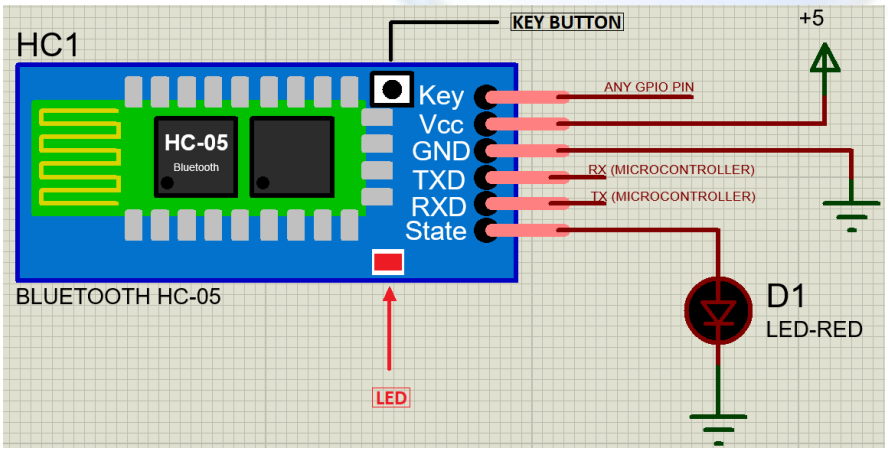


Nome Bluethooth	HC-05
Sensibilità	a -80dBm
Trasmissione	fino a +4dBm RF
Antenna	integrata
DATA bit	8
Stop bit	1
Bit parità	NO
Controllo dei dati	Sì
Dimensioni	37 x 17 x 7mm
Peso	4,5g

Il modulo HC-05 è un modulo in gradi di convertire una porta seriale UART in una porta Bluetooth; questo dispositivo può essere utilizzato per consentire ad un microprocessore (MCU) installato su una scheda Arduino di comunicare con un dispositivo dotato di Bluetooth (come un PC, smartphone o tablet).

Il modulo si basa sul chip CSR Bluecore 04 External, utilizza la tecnologia CMOS e AFH (adptive frequency hopping function), ed è compatibile con il protocolloBluetooth V2.0 3Mbps + EDR (Enhanced Data Rate).

La caratteristica principale del modulo HC-05 è che può essere impostato in due modalità tramite comandi AT a seconda delle esigenze dell'utente: come dispositivo master o come dispositivoslave; questo permetterà di utilizzare lo stesso tipo di modulo per creare una rete di dispositivi.



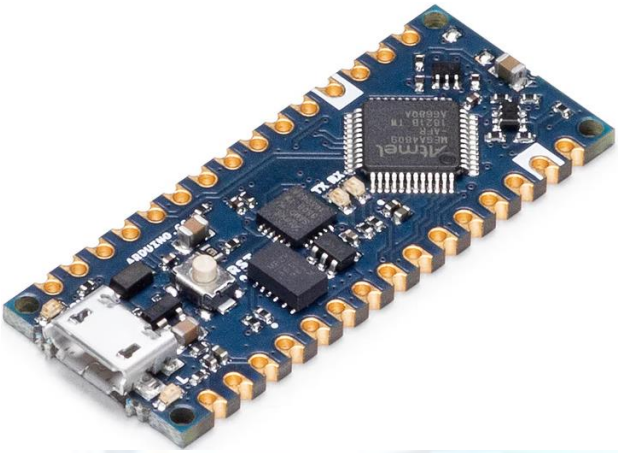
(immagine da: Micricontroller Lab immagine da: Micricontroller Lab)

Comandi

Comandi AT	Funzione
AT	Testa la connessione all'interfaccia AT
AT + reset	Resetta il nostro componente
AT + version	Comunica la versione del firmware
AT + orgl	Riporta le impostazioni correnti alle predefinite
AT + addr	Il dispositivo ci comunica il proprio indirizzo
AT + name	Domandiamo o impostiamo il nome del dispositivo
AT + rname	Domandiamo il nome di un dispositivo bluetooth a cui siamo collegati (remoto)
AT + role	Domandiamo o impostiamo il ruolo del dispositivo (1=Master / 0=Slave)
AT + class	Domandiamo o impostiamo la classe del dispositivo (Class of Device CoD)
AT + iac	Domandiamo o impostiamo le richieste del codice di accesso
AT + inqm	Domandiamo o impostiamo le richieste della modalità di accesso
AT + pswd	Domandiamo la password o ne impostiamo l'associazione
AT + uart	Domandiamo i parametri UART o li reimpostiamo
AT + cmode	Domandiamo la modalità di connessione o la impostiamo
AT + bind	Domandiamo o impostiamo l'associazione all'indirizzo del bluetooth
AT + polar	Domandiamo o impostiamo la polarità di output del LED
AT + pio	Impostiamo o resettiamo un pin di Input/Output (I/O) utente
AT + mpio	Impostiamo o resettiamo più pin di I/O utente
AT + mpio?	Domandiamo l'I/O di un pin utente
AT + ipscan	Domandiamo o impostiamo i parametri di scansione
AT + sniff	Domandiamo o impostiamo i parametri di risparmio energetico SNIFF
AT + senm	Domandiamo o impostiamo i modi di sicurezza e crittografia
AT + rmsad	Elimina un dispositivo autenticato dalla lista
AT + fsad	Trova un dispositivo dalla lista dei dispositivi autenticati
AT + adcn	Domandiamo il numero totale dei dispositivi dalla lista dei dispositivi autenticati
AT + mrad	Domandiamo quale è il dispositivo autenticato utilizzato più di recente
AT + state	Domandiamo lo stato corrente del dispositivo
AT + init	Inizializziamo il profilo SPP (ricerca)
AT + inq	Domandiamo quale sia il dispositivo individuabile più vicino
AT + inqc	Cancelliamo la ricerca dei dispositivi individuabili
AT + pair	Associazione al dispositivo
AT + link	Connessione ad un dispositivo remoto
AT + disc	Disconnessione da un dispositivo remoto
AT + ensniff	Entriamo in modalità risparmio energetico
AT + exsniff	Usciamo dalla modalità di risparmio energetico

Arduino nano every

(Usato solo nei test e non nel progettato finale)



La scheda Arduino Nano Every è compatibile con i 5V nella forma più piccola: 45*18 mm; è una scheda usata per molti progetti che richiedono un microcontrollore piccolo e facile da usare, adatto per invenzioni indossabili, robotica a basso costo, strumenti musicali elettronici e per controllare le parti piccole di progetti grandi.

La scheda presenta 14 pin digitali, di cui 5 supportano il PWM, e 8 ingressi analogici, più un'uscita analogica con DAC a 10 bit. Supporta una tensione di funzionamento a 5V e può essere alimentato tramite USB o attraverso il pin VIN, accettando tensioni da 5V fino a 21V, rendendola molto flessibile per progetti alimentati esternamente. Include inoltre interfacce di comunicazione come UART, SPI e I²C, ampliando la gamma di dispositivi con cui può interagire.

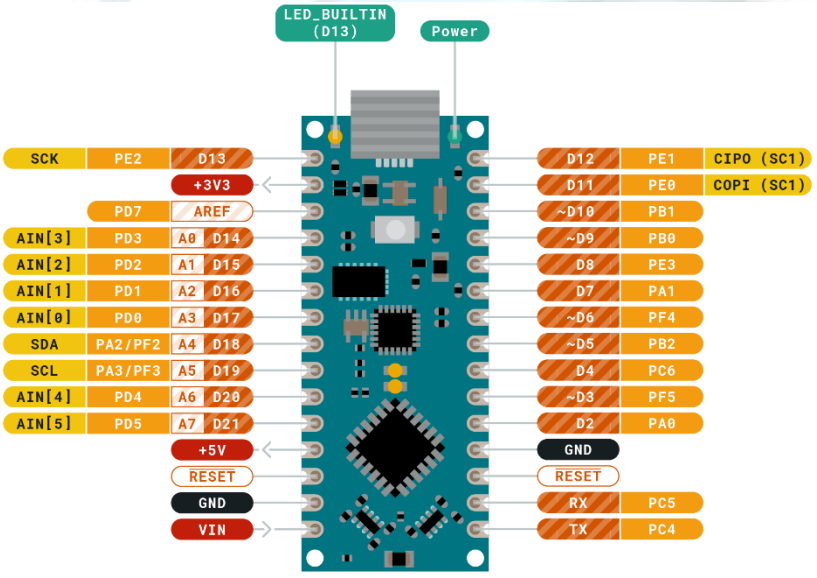
Il microcontrollore ATmega4809 introduce anche supporto per Core Independent Peripherals (CIPs), che consentono alle periferiche interne di operare in modo autonomo, alleggerendo il carico di lavoro della CPU. Inoltre, la scheda supporta modalità di risparmio energetico avanzate, risultando particolarmente utile per progetti che devono essere alimentati a batteria o che richiedono bassi consumi energetici.

Specifiche Tecniche

Microcontrollore	ATMega4809
Tensione di esercizio	5V
Vin min-Max	7-21V
Corrente CC per pin I/O	20mA
Corrente CC per pin da 3,3 V	50mA
Velocità di clock	20MHz
Memoria Flash CPU	48KB
SRAM	6KB
Memoria EEPROM	256byte
Perni PWM	5 (D3, D5, D6, D9, D10)
UART	1
SPI	1

I2C	1
Pin di ingresso analogici	8 (ADC 10 bit)
Pin di uscita analogici	Solo tramite PWM
Interruzioni esterne	tutti i pin digitali
LED_INCORPORATO	13
USB	Utilizza ATSAMD11D14A
Lunghezza	45 millimetri
Larghezza	18 millimetri
Peso	5 gr (con intestazioni)

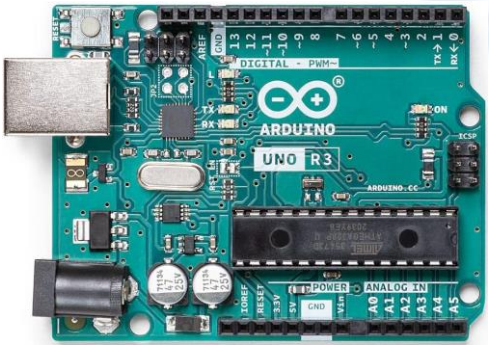
Porte PIN



- Ground
- Power
- LED
- Internal Pin
- SWD Pin
- Other Pin (Reset)
- Digital Pin (D2, D3...)
- Analog Pin (A0, A1…)
- Default
- Porta microcontrollore

Arduino Uno

(Usato solo nei test del progetto e non nel progettato finale)

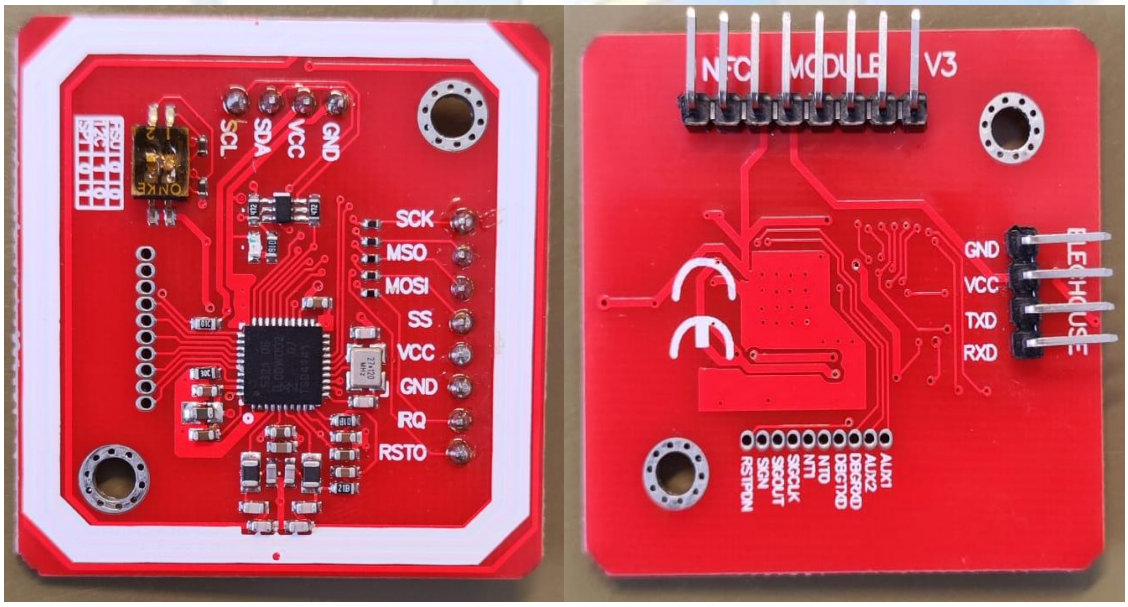


L'**Arduino Uno** è una scheda microcontrollore che utilizza il chip **ATmega328P**. Dispone di 14 pin digitali per input/output, 6 dei quali supportano l'output PWM, oltre a 6 ingressi analogici. La scheda include un risonatore ceramico da 16 MHz (modello CSTCE16M0V53-R0), una porta USB per la connessione al computer, un jack per l'alimentazione esterna, un header ICSP e un pulsante di reset. Fornisce tutto il necessario per gestire il microcontrollore: è sufficiente collegarla al PC tramite USB o alimentarla con una batteria o un adattatore AC-DC per iniziare ad utilizzarla.

Specifiche Tecniche

Microcontrollore	ATmega328p
Voltaggio Operativo	5V
Voltaggio Imput	7-12V
Voltaggio Imput (limite)	6-20V
Pin Digitale I/O	14
Pin PWM Digitale I/O	6
Pin Input Analogici	6
Corrente DC per I/O	20 mA
Corrente DC per 3.3V	50 Ma
Memoria Flash	32 KB
SRAM	2 KB
EEPROM	1 KB
Velocita Clock	16 MHz
LED_BUILTIN	13
Lunghezza	68.6 mm
Larghezza	53.4 mm
Peso	25 g

Modulo NFC V3



Il Modulo NFC v3 è un dispositivo hardware che consente la comunicazione tramite tecnologia NFC (Near Field Communication), un tipo di connessione wireless che funziona su brevi distanze, solitamente entro 4 cm. È ampiamente utilizzato per progetti elettronici che coinvolgono microcontrollori come Arduino, Raspberry Pi e altre piattaforme simili.

CARATERISTICHE

- **Chip:** L'NFC V3 è spesso basato su chip NXP PN532, uno dei più diffusi
- **Interfaccia di Comunicazione:**
    1. **I2C** (modalità di default)
    2. **SPI**
    3. **UART**
- **Distanza Lettura:** Solitamente intorno a 2-4 cm, ma può variare in base al tag utilizzato
- **Velocità di comunicazione:**
    1. **I2C:** Velocita standard (100 kHz) e veloce (400kHz)
    2. **SPI:** Supporto fino a 5 Mbps
    3. **UART:** Baud rate configurabile
- **Tensione Operativa:** 3,3 V e 5 V, adatto sia per progetti Arduino e Raspberry
- **Protocolli NFC supportati:**
    1. **ISO/IEC 14443-A/B**
    2. **ISO/IEC 18092**
- **Dimensioni:** Solitamente attorno ai 40 mm x 40mm

Modalità di Funzionamento

- **Reader/writer Mode:** In questa modalità, il modulo può leggere e scrivere su tag NFC.
- **Peer-to-Peer Mode:** Consente la comunicazione diretta tra due dispositivi NFC, ad esempio tra un telefono NFC e il modulo stesso.
- **Card Emulation Mode:** Il modulo può simulare una smart card NFC, permettendo ad altri lettori di leggere i dati del modulo come se fosse un tag NFC.

Esempio Codice

```
#include <Wire.h>            // Libreria per la comunicazione I2C
#include <Adafruit_PN532.h>      // Libreria Adafruit per PN532

// Definisci i pin I2C
#define SDA_PIN A4
#define SCL_PIN A5

// Crea un'istanza dell'oggetto PN532 usando l'I2C
Adafruit_PN532 nfc(SDA_PIN, SCL_PIN);

void setup(void) {
  Serial.begin(115200);        // Inizializza la comunicazione seriale
```

```
  Serial.println("Inizializzazione del modulo NFC...");

  nfc.begin();              // Avvia la comunicazione con il modulo NFC

  uint32_t versiondata = nfc.getFirmwareVersion();
  if (!versiondata) {
    Serial.println("Modulo NFC non trovato!");
    while (1);              // Se non trova il modulo NFC, blocca l'esecuzione
  }

  // Mostra la versione del firmware del modulo NFC
  Serial.print("Firmware trovato: 0x");
  Serial.println(versiondata, HEX);

  // Configura il modulo NFC per la lettura di tag
  nfc.SAMConfig();
  Serial.println("Modulo NFC pronto per leggere tag.");
}

void loop(void) {
  Serial.println("Posiziona un tag NFC vicino al lettore...");

  uint8_t success;
  uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };  // Array per memorizzare l'UID
  uint8_t uidLength;                        // Variabile per la lunghezza dell'UID

  // Prova a leggere un tag NFC (protocollo ISO14443A come MIFARE)
  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength);

  if (success) {
```

```
    // Se viene rilevato un tag, stampa l'UID del tag
    Serial.println("Tag NFC trovato!");
    Serial.print("UID del tag: ");

    for (uint8_t i = 0; i < uidLength; i++) {
      Serial.print(" 0x");
      Serial.print(uid[i], HEX);  // Stampa l'UID in formato esadecimale
    }
    Serial.println();
    delay(1000);  // Aspetta 1 secondo prima di cercare di nuovo
  }
  else {
    Serial.println("Nessun tag trovato.");
  }

  delay(1000);  // Aspetta un po' prima di riprovare
}
```

Vantaggi e Limitazioni

- **Vantaggi:**
  a) Facile da usare con piattaforme come Arduino e Raspberry Pi.
  b) Compatibile con una vasta gamma di protocolli NFC.
  c) Supporto per lettura, scrittura ed emulazione di tag NFC.
- **Limitazioni:**
  a) Portata limitata a pochi centimetri.
  b) Richiede la libreria e una configurazione corretta per l'interfaccia desiderata.
  c) Prestazioni variabili a seconda del tipo di tag utilizzato.

RFId

RFId (Radio Frequency Identification) viene utilizzata per l'identificazione automatica di oggetti animali e persone;

basato sulla lettura di informazioni nel Tag.

Componenti:

- Tag o transponder
- Reader o tranceiver

- Antenna
- Sistema elaborazione

TAG (dipende dal tipo di alimentazione):

- **trasponder passivo:** utilizzano l'onda a radio frequenza generato dal lettore sai come fonte di energia per alimentazione che per ricevere dati, Basso costo tempi di vita più lunghi, distanza limitati;
- **trasponder semi passivo:** batteria incorporata, grande distanza di comunicazione, costosi, impossibile stabilire stato batteria;
- **trasponder attivo:** completamente autonomi, autoalimentati da batteria e possiedono un trasmettitore attivo bordo, grande distanza di comunicazione, costosi, impossibile stabilire stato batteria;

Distanza(tag): da pochi centimetri a qualche metro;

Contenuto(tag): antenna, chip che svolge semplici elaborazioni, piccola memoria;

Assemblaggio(tag): substrato, antenna, chip, overlay;

Campi che utilizzano RFId:

- Industriale
- medico
- automotivo

PN532 --> basato su corrente

Distanza: 5-6 cm

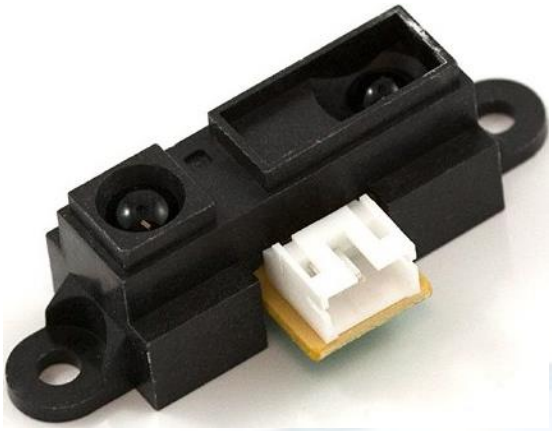
reader che writer

alimentazione: 3,3V a 5V

SDA --> serial data

SCL --> serial connector

Sensore IR Sharp (SOLO NELLE PROVE)



Il sensore IR sharp è un tipo di sensore a infrarossi usato per il rilevamento di distanze di oggetti senza andare in contatto.

Il sensore emette un fascio di luce infrarossi attraverso un LED IR. La luce viene riflessa dall'oggetto e rilevata da un fotodiodo all'interno del sensore e basandosi sull'angolo di riflessione e l'intensità della luca riflessa, il sensore calcola la distanza dall'oggetto.

Specifiche Tecniche

Nome	Sensore IR Sharp
Portata di misura	10 cm - 80 cm
Tensione di Alimentazione	4.5 V - 5.5 V
Corrente Operativa	Circa 30 mA
Tempo di risposta	Circa 38 ms
Segnale di uscita	0.4 V a 3.1 V
Uscita	Analogica
Angolo di rilevamento	10 °
Dimensioni	29.5mm x 13mm x 21.6mm
Peso	Circa 3.5 g

Sharp GP2Y0D805Z0F Sensore IR

Il Sharp GP2Y0D805Z0F è un sensore IR (a infrarossi) di prossimità progettato per rilevare oggetti in un intervallo molto ravvicinato. Si distingue per la sua velocità di risposta elevata e per la capacità di rilevare la presenza o l'assenza di oggetti a una distanza specifica, generalmente prefissata a circa 5 cm. Ecco una panoramica delle sue caratteristiche principali:

Caratteristiche principali:

1. **Tecnologia**: Utilizza la riflessione della luce infrarossa per rilevare la presenza di oggetti. Un LED IR emette luce, e un sensore riceve il segnale riflesso per determinare la presenza di un oggetto.
2. **Distanza di rilevamento**: Ottimizzato per rilevare oggetti a circa 5 cm.
3. **Uscita digitale**: Fornisce un segnale digitale (HIGH/LOW) in base alla rilevazione. Se un oggetto si trova entro il range, l'uscita è attivata (LOW o HIGH, a seconda della configurazione).
4. **Velocità di risposta**: Molto rapida, adatta per applicazioni in cui il tempo di reazione è critico.
5. **Tensione operativa**: Solitamente alimentato a 5V DC.
6. **Compattezza**: Il modulo è molto piccolo e leggero, ideale per applicazioni con vincoli di spazio.

Applicazioni tipiche

- **Robotica**: Per evitare ostacoli a distanza ravvicinata o rilevare la presenza di oggetti.
- **Automazione**: In sistemi di conteggio o di rilevamento rapido.
- **Elettronica di consumo**: Per attivare funzioni di prossimità in dispositivi come distributori automatici.
- **Sicurezza**: Come sensore di prossimità in sistemi di protezione.
- Vantaggi
- **Affidabilità**: Funziona bene con oggetti di vari materiali e colori.
- **Bassa complessità**: Non richiede circuiti complessi per elaborare il segnale.
- **Consumo energetico ridotto**: Adatto per dispositivi a bassa potenza.

Limitazioni

- **Range fisso**: Il sensore non misura la distanza, ma rileva la presenza entro una soglia preimpostata.
- **Sensibilità alla luce ambientale**: Può essere influenzato da fonti di luce intensa, anche se ben protetto contro interferenze.

Specifiche:

- **Intervallo**: 0,5 cm-5 cm
- **Tasso di campionamento**: 390Hz
- **Tensione operativa**: 2.7V-6.2V
- **Consumo attuale**: 5Ma (tipico)
- **Dimensione del modulo**: 21,6 x 8,9 x 10,4 mm
- **Peso senza perni di intestazione**: 1.3g

SCHEMA A BLOCCO

Schema

1. Sensori e Rilevamento

- **Sensore di Luce**
  - Fototransistor o LDR per rilevare sorgenti luminose.
  - Uscita: Segnale analogico o digitale per indicare la presenza di luce.
- **Sensori di Prossimità**
  - IR o Ultrasuoni per rilevare ostacoli.
  - Uscita: Segnale di distanza per l'evitamento degli ostacoli.
- **Sensore NFC**
  - Lettore per identificare i TAG NFC.
  - Uscita: UID del TAG letto.

2. Unità di Elaborazione Centrale (Microcontrollore)

- **Arduino UNO**
  - Riceve input dai sensori.
  - Esegue algoritmi per la navigazione autonoma e il rilevamento di luci.
  - Controlla i motori e il modulo Bluetooth.
  - Comunica con il sistema di controllo remoto.
- **Algoritmi di Navigazione**
  - Evitamento ostacoli.
  - Ricerca di sorgenti luminose.
  - Lettura e validazione TAG NFC.

3. Attuatori

- **Motori DC con Driver di Controllo**
  - Movimento del robot.
  - Input: Comandi di direzione e velocità dall’Arduino.
- **Eventuale Servo**
  - Movimenti di precisione per allineamento al TAG NFC o direzione sensore luce.

4. Comunicazione

- **Modulo Bluetooth (HC-05/HC-06)**
  - Trasmissione UID al sistema remoto.
  - Ricezione del comando per spegnere la sorgente luminosa.
- **Display LCD (Facoltativo)**
  - Mostra informazioni sui TAG rilevati e l'esito del confronto.

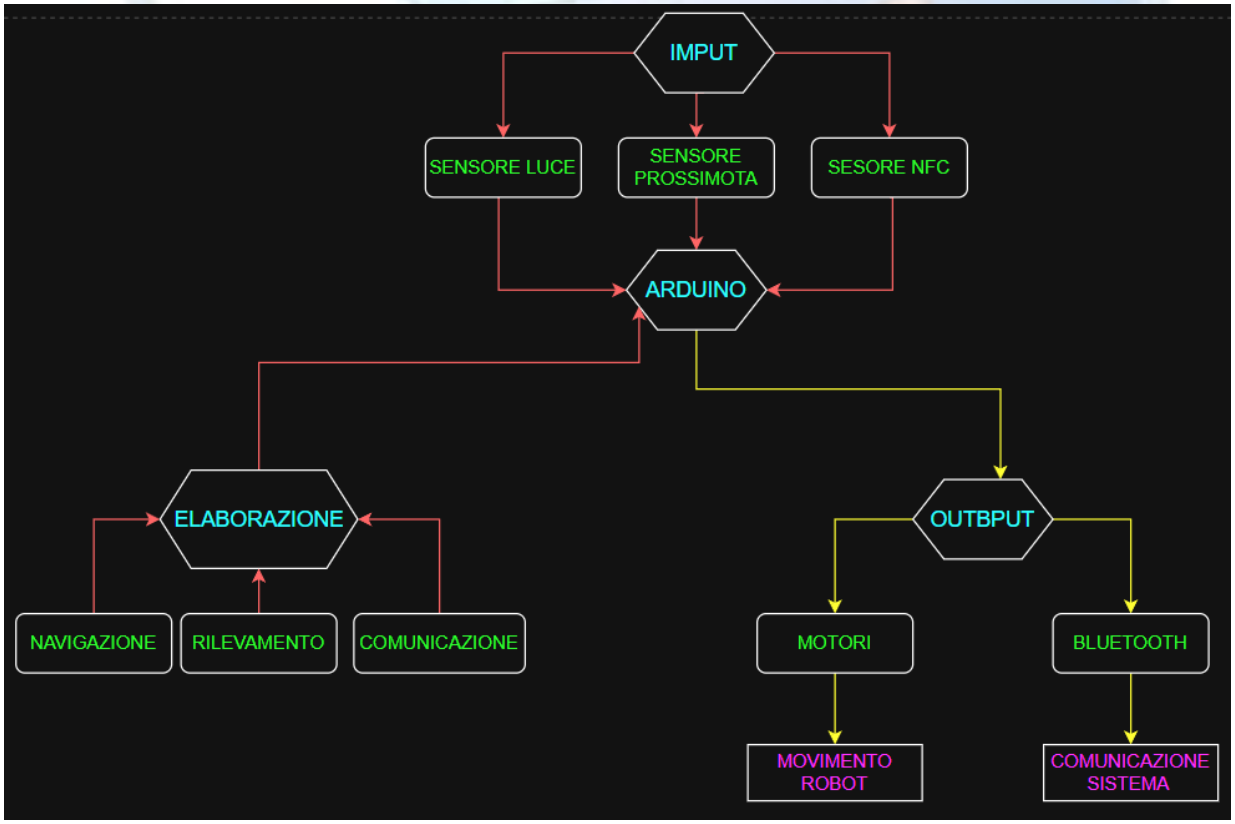
5. Sistema di Controllo Remoto

- **Arduino UNO del Sistema di Controllo**
  - Riceve gli UID inviati dal robot.
  - Confronta i codici con un elenco predefinito.
  - Spegne la sorgente luminosa corrispondente.
- **Modulo Bluetooth**
  - Comunica con il robot.

6. Alimentazione

- **Batteria Ricaricabile**
  - Alimenta il microcontrollore, i sensori, i motori e i moduli di comunicazione.

DIAGRAMMA



CODICI

Libreria Robottino

```
// Per far funzionare questa libreria bisogna spostarla nella cartella librerie Arduino IDE.

#ifndef G1LIBMINICARROBOT02_H

#define G1LIBMINICARROBOT02_H

// dichiarazione librerie necessarie

#include <Arduino.h>

#include <Wire.h>

#include <SoftwareSerial.h> // per il modulo bluetooth HC-05

#include <Adafruit_PN532.h> // per sensore NFC PN532

// --- programmazione oggetti per robottino ---

class Motori_cc {

  private:

    int pwm_a;  //PWM control for motor outputs

    int pwm_b;  //PWM control for motor outputs

    int dir_a;  //direction control for motor outputs

    int dir_b;  //direction control for motor outputs


  public:

    // Costruttore per i motori

    Motori_cc(int pwma, int pwmb, int dira, int dirb) : pwm_a(pwma), pwm_b(pwmb), dir_a(dira), dir_b(dirb) {


    }


    // --- setup pins motori ---

    void MotoriSetup() {

      pinMode(pwm_a, OUTPUT);  //Set control pins to be outputs

      pinMode(pwm_b, OUTPUT);
```

```
    pinMode(dir_a, OUTPUT);
    pinMode(dir_b, OUTPUT);
    digitalWrite(dir_a, LOW);
    digitalWrite(dir_b, LOW);
    analogWrite(pwm_a, 0);
    analogWrite(pwm_b, 0);
}

// --- movimenti ---
void Indietro(int d) {
    digitalWrite(dir_a, LOW);
    digitalWrite(dir_b, LOW);

    analogWrite(pwm_a, 100);
    analogWrite(pwm_b, 100);

    delay(d);
}

void Avanti() {
    digitalWrite(dir_a, HIGH);
    digitalWrite(dir_b, HIGH);

    analogWrite(pwm_a, 80);
    analogWrite(pwm_b, 80);
}

void Destra() {
    digitalWrite(dir_a, HIGH);
    digitalWrite(dir_b, HIGH);
```

```
    analogWrite(pwm_a, 0);
    analogWrite(pwm_b, 70);
}

void Sinistra() {
    digitalWrite(dir_a, HIGH);
    digitalWrite(dir_b, HIGH);

    analogWrite(pwm_a, 90);
    analogWrite(pwm_b, 0);
}

void GiroSuSeStesso(int d) {
    digitalWrite(dir_a, HIGH);
    digitalWrite(dir_b, LOW);

    analogWrite(pwm_a, 100);
    analogWrite(pwm_b, 100);

    delay(d);
}

void Tre_Giri() {
    digitalWrite(dir_a, HIGH);
    digitalWrite(dir_b, LOW);

    analogWrite(pwm_a, 100);
    analogWrite(pwm_b, 100);
```

```
    delay(7600);
}

void Giro_180() {
    digitalWrite(dir_a, HIGH);
    digitalWrite(dir_b, LOW);

    analogWrite(pwm_a, 100);
    analogWrite(pwm_b, 100);

    delay(1400);
}

void Giro_90_Dx() {
    digitalWrite(dir_a, LOW);
    digitalWrite(dir_b, HIGH);

    analogWrite(pwm_a, 100);
    analogWrite(pwm_b, 100);

    delay(700);
}

void Giro_90_Sx() {
    digitalWrite(dir_a, HIGH);
    digitalWrite(dir_b, LOW);

    analogWrite(pwm_a, 100);
    analogWrite(pwm_b, 100);
```

```
        delay(700);
    }

    void Fermo (int d) {
        digitalWrite(dir_a, HIGH);
        digitalWrite(dir_b, HIGH);
        analogWrite(pwm_a, 0);
        analogWrite(pwm_b, 0);

        delay(d);
    }

    void StopLoop() {
        while(true) {
            digitalWrite(dir_a, HIGH);
            digitalWrite(dir_b, HIGH);
            analogWrite(pwm_a, 0);
            analogWrite(pwm_b, 0);
        }
    }
};

// Sensore ultrasuoni
class Ultrasuoni {
  private:
    int TrigPin;
    int EchoPin;

  public:
    // Costruttore ultrasuoni
```

```
    Ultrasuoni(int trigPin, int echoPin) : TrigPin(trigPin), EchoPin(echoPin) {

    }

    // --- setup ultrasuoni pins ---
    void HCSR04_Setup() {
        // HCSR04 trig echo pins set
        pinMode(TrigPin, OUTPUT);
        pinMode(EchoPin, INPUT);
    }

    float DistanzaOstacoli() {
        // Pulire TrigPin sensore
        digitalWrite(TrigPin, LOW);
        delayMicroseconds(2);

        //impulso di 10 microsecondi
        digitalWrite(TrigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(TrigPin, LOW);

        // Leggi il tempo di ritorno dell'eco
        long durata = pulseIn(EchoPin, HIGH);

        return (durata * 0.0343) / 2;
    }
};

// Phototransistors
class Phototransistor {
```

```
    private:
        int phototransistorPin;

    public:
        // Costruttore phototransistor
        Phototransistor(int PhototransistorPin) : phototransistorPin(PhototransistorPin) {

        }

        // --- Setup phototransistor ---
        void Phototransistor_Setup() {
            pinMode(phototransistorPin, INPUT);
        }

        // --- actions phototransistor ---
        int Lettura() {
            int reading = analogRead(phototransistorPin);
            float voltage = reading * 5.0 / 1023.0;

            if (voltage >= 1) {
                return 0;
            }

            return 1;
        }
};

class NFC {
private:
    uint8_t success;
```

```
    uint8_t uid[7]; // Maximum UID size (7 bytes for MIFARE)
    uint8_t uidLength;

    String uidString;

    int SDA_pin;
    int SCL_pin;
    Adafruit_PN532 nfc; // Create an Adafruit_PN532 object

public:
    // Constructor to initialize SDA and SCL pins
    NFC(int SDA, int SCL) : SDA_pin(SDA), SCL_pin(SCL), nfc(SDA, SCL) {
        // Initialization
    }

    // Setup method to initialize NFC
    void NFC_Setup() {
        nfc.begin();
        nfc.SAMConfig(); // Configure the PN532 to read RFID tags
    }

    // Check if an NFC card is available
    String Check_NFC() {
        uidString = "";
        uidLength = 0;

        // Read NFC card
        success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength);

        if (success) {
```

```
            for (uint8_t i = 0; i < uidLength; i++) {
                uidString += "0x"; // Prefix "0x"
                uidString += String(uid[i], HEX); // Value in hexadecimal format
                if (i < uidLength - 1) {
                    uidString += " "; // Space between values, if not the last
                }
            }
        } else {
            uidString = "Nothing"; // No card detected
        }

        return uidString; // Return the UID string or "Nothing"
    }
};

// Sensore IR analog
class IR_sensor {
    private:
        int IR_Pin;
        const float Soglia;
        float LetturaIR;
        int State;

    public:
        IR_sensor(int IRpin, const float S) : IR_Pin(IRpin), Soglia(S) {

        }

        void IR_Setup() {
            pinMode(IR_Pin, INPUT);
```

```
    }

    int Read_IR() {
        LetturaIR = analogRead(IR_Pin);

        if (LetturaIR <= Soglia) {
            return 0;
        }

        return 1;
    }
};

// Bluetooth HC-05
class HC05 {
private:
    int RxPin;
    int TxPin;
    SoftwareSerial bt; // Create a SoftwareSerial object for Bluetooth communication

public:
    // Constructor to initialize the Rx and Tx pins
    HC05(int Rx, int Tx) : RxPin(Rx), TxPin(Tx), bt(Rx, Tx) {
        // Initialize the SoftwareSerial object
    }

    // Setup method to begin Bluetooth communication
    void bt_Setup() {
        bt.begin(9600); // Ensure the baud rate is correct
    }
```

```
    // Method to send a string over Bluetooth
    void SendString(String message) {
        bt.println(message); // Send the string over Bluetooth
    }

    // Method to read a string from Bluetooth (optional)
    String ReadString() {
        String received = "";
        while (bt.available()) {
            char c = bt.read(); // Read a character
            received += c; // Append the character to the string
        }
        return received; // Return the received string
    }
};
#endif
```

Codice Robottino

```
// Light follower, codice sorgente robottino, Vidotti Zhou Grecu G1 5TELA
#include <G1LIBMiniCarRobot02.h> // Libreria classi per robottino Gruppo 1

// Area globale
int LoopState = 0;
const int DIM = 9;
String Vet_Superamenti[DIM]; // Memoria 10 superamenti, 0---9
String SuperamentoAttuale;

// Motori cc
int pin_a = 3;
int pin_dir_a = 2;
```

```
int pin_b = 9;
int pin_dir_b = 8;

// HCSR04
const float SogliaHCSR04 = 7;
float d_HCSR04 = 0;
int pin_Trig = 52;
int pin_Echo = 53;
int LetturaHCSR04 = 0;

// IR sensors
int pin_IR_Sx = 51;
int pin_IR_Dx = 50;
int LetturaIR_Sx = 0;
int LetturaIR_Dx = 0;

// Phototransistors
int pin_BPX45_Sx = A13;
int pin_BPX45_Ce = A11;
int pin_BPX45_Dx = A8;
int LightVar_Sx = 0;
int LightVar_Ce = 0;
int LightVar_Dx = 0;

// NFC
int pin_SDA = 20;
int pin_SCL = 21;

// Bluetooth pin incrociati
int pin_bt_Tx = 48;
```

```
int pin_bt_Rx = 49;ù
String Lettura_bt = "";

// --- Istanziazione Oggetti ---
Motori_cc Motori(pin_a, pin_b, pin_dir_a, pin_dir_b);
Ultrasuoni HCSR04(pin_Trig, pin_Echo, SogliaHCSR04);
IR_sensor IR_sensor_Sx(pin_IR_Sx);
IR_sensor IR_sensor_Dx(pin_IR_Dx);
Phototransistor BPX45_Sx(pin_BPX45_Sx);
Phototransistor BPX45_Ce(pin_BPX45_Ce);
Phototransistor BPX45_Dx(pin_BPX45_Dx);
NFC nfc_G1(pin_SDA, pin_SCL);
HC05 bt_HC05(pin_bt_Rx, pin_bt_Tx);

// --- Funzioni ---
void LettureOstacolo() {
  // Lettura Ultrasuoni e IR
  LetturaHCSR04 = HCSR04.LetturaOstacolo();
  LetturaIR_Sx = IR_sensor_Sx.Read_IR();
  LetturaIR_Dx = IR_sensor_Dx.Read_IR();
}

void LettureLuce() {
  LightVar_Sx = BPX45_Sx.Lettura();
  LightVar_Ce = BPX45_Ce.Lettura();
  LightVar_Dx = BPX45_Dx.Lettura();
}

void Distanza_HCSR04() {
  d_HCSR04 =  HCSR04.LetturaDistanza();
```

```
}

void AnalizzaSuperamenti() {
  // Controllo ripetizioni di "Destra" e "Sinistra"
  int destraCount = 0;
  int sinistraCount = 0;
  int consecutiveCount = 1; // Contatore per movimenti consecutivi
  String lastMovement = ""; // Ultimo movimento registrato

  for (int i = 0; i < DIM; i++) {
    if (Vet_Superamenti[i] == "Destra") {
      destraCount++;
      if (lastMovement == "Destra") {
        consecutiveCount++;
      } else {
        consecutiveCount = 1; // Reset del contatore
      }
      lastMovement = "Destra";
    } else if (Vet_Superamenti[i] == "Sinistra") {
      sinistraCount++;
      if (lastMovement == "Sinistra") {
        consecutiveCount++;
      } else {
        consecutiveCount = 1; // Reset del contatore
      }
      lastMovement = "Sinistra";
    }
  }

  // Controllo più di 3 movimenti consecutivi a destra o sinistra
```

```
  if (consecutiveCount > 3) {
    Motori.Indietro(); // Funzione per andare indietro
    delay(75);
  }

  // Controllo ripetizioni di movimenti alternati
  if (destraCount > 0 && sinistraCount > 0) {
    // Se ci sono movimenti alternati, controllo sequenza
    bool alternato = true;
    for (int i = 0; i < DIM - 1; i++) {
      if ((Vet_Superamenti[i] == "Destra" && Vet_Superamenti[i + 1] == "Destra") ||
          (Vet_Superamenti[i] == "Sinistra" && Vet_Superamenti[i + 1] == "Sinistra")) {
        alternato = false;
        break;
      }
    }

    if (alternato) {
      Motori.Giro_180(); // Gira di 180° se è bloccato
    }
  }
}

void setup() {
  Motori.MotoriSetup();
  HCSR04.HCSR04_Setup();
  IR_sensor_Sx.IR_Setup();
  IR_sensor_Dx.IR_Setup();
  BPX45_Sx.Phototransistor_Setup();
  BPX45_Ce.Phototransistor_Setup();
```

```
  BPX45_Dx.Phototransistor_Setup();
  nfc_G1.NFC_Setup();
  bt_HC05.bt_Setup();

  // Azzeramento memoria superamenti
  for (int i = 0; i < DIM; i++) {
    Vet_Superamenti[i] = "";
  }

  // Tre giri su se stesso
  Motori.Tre_Giri();
}

void loop() {
  Motori.ZigZag(LoopState);

  // Letture ostacolo
  LettureOstacolo();

  Distanza_HCSR04();

  LettureLuce();

  if (LightVar_Sx == 1 || LightVar_Ce == 1 || LightVar_Dx == 1) {
    Motori.Avanti();

    Distanza_HCSR04();

    if (d_HCSR04 < 4) {
      Motori.Fermo(10);
```

```
for (int i=0; i < 5; i++) {
  String LetturaNFC = nfc_G1.Check_NFC();

  if (LetturaNFC != "Nothing") {
    bt_HC05.SendString(LetturaNFC);
    delay(100);
    Lettura_bt = bt_HC05.ReadString();
  }

  Motori.Destra();
  delay(25);

  LetturaNFC = nfc_G1.Check_NFC();

  if (LetturaNFC != "Nothing") {
    bt_HC05.SendString(LetturaNFC);
    delay(100);
    Lettura_bt = bt_HC05.ReadString();

  }

  Motori.Fermo(10);
  Motori.Sinistra();
  delay(25);

  LetturaNFC = nfc_G1.Check_NFC();

  if (LetturaNFC != "Nothing") {
    bt_HC05.SendString(LetturaNFC);
```

```
      delay(100);
      Lettura_bt = bt_HC05.ReadString();

    }

    Motori.Fermo(1);
  }

  if (Lettura_bt == "end") {
    Motori.Indietro();
    delay(1000);
    Motori.Tre_Giri();
    Motori.StopLoop();
  }
}

LettureLuce();

if (LightVar_Sx == 0 && LightVar_Ce == 0 && LightVar_Dx == 1) { // case 001
  Motori.Destra();
  delay(100);
  LettureLuce();
}

if (LightVar_Sx == 1 && LightVar_Ce == 0 && LightVar_Dx == 0) { // case 100
  Motori.Sinistra();
  delay(100);
  LettureLuce();
}
```

```
  if (LightVar_Sx == 0 && LightVar_Ce == 1 && LightVar_Dx == 1) { // case 011
    Motori.Destra();
    delay(100);
    LettureLuce();
  }

  if (LightVar_Sx == 1 && LightVar_Ce == 1 && LightVar_Dx == 0) { // case 110
    Motori.Sinistra();
    delay(100);
    LettureLuce();
  }

  if (LightVar_Sx == 0 && LightVar_Ce == 1 && LightVar_Dx == 0) { // case 010
    Motori.Avanti();
    delay(100);
    LettureLuce();
  }

  if (LightVar_Sx == 1 && LightVar_Ce == 0 && LightVar_Dx == 1) { // case 010
    Motori.Avanti();
    delay(100);
    LettureLuce();
  }

  if (LightVar_Sx == 1 && LightVar_Ce == 1 && LightVar_Dx == 1) { // case 111
    Motori.Avanti();
    delay(100);
    LettureLuce();
  }
}
```

```
// --- Superamento Ostacoli ---
if (LetturaIR_Sx == 0 && LetturaHCSR04 == 0 && LetturaIR_Dx == 0) {
  SuperamentoAttuale = "/";
}

if (LetturaIR_Sx == 0 && LetturaHCSR04 == 0 && LetturaIR_Dx == 1) {
  Motori.Sinistra();
  delay(100);
  SuperamentoAttuale = "Sinistra";
}

if (LetturaIR_Sx == 1 && LetturaHCSR04 == 0 && LetturaIR_Dx == 0) { // case 100
  Motori.Destra();
  delay(100);
  SuperamentoAttuale = "Destra";
}

if (LetturaIR_Sx == 0 && LetturaHCSR04 == 1 && LetturaIR_Dx == 1) { // case 011
  Motori.Sinistra();
  delay(200);
  SuperamentoAttuale = "Sinistra+";
}

 if (LetturaIR_Sx == 1 && LetturaHCSR04 == 1 && LetturaIR_Dx == 0) { // case 110
  Motori.Destra();
  delay(200);
  SuperamentoAttuale = "Destra+";
}
```

```
if (LetturaIR_Sx == 0 && LetturaHCSR04 == 1 && LetturaIR_Dx == 0) { // case 010
  Motori.Giro_180();
  SuperamentoAttuale = "180";
}

if (LetturaIR_Sx == 1 && LetturaHCSR04 == 0 && LetturaIR_Dx == 1) { // case 101
  Motori.Giro_180();
  SuperamentoAttuale = "180";
}

if (LetturaIR_Sx == 1 && LetturaHCSR04 == 1 && LetturaIR_Dx == 1) { // case 111
  Motori.Giro_180();
  SuperamentoAttuale = "180";
}

while (d_HCSR04 < 4) {
   Motori.Fermo(1);

   for (int i=0; i < 5; i++) {
     String LetturaNFC = nfc_G1.Check_NFC();

     if (LetturaNFC != "Nothing") {
       bt_HC05.SendString(LetturaNFC);
       delay(100);
       Lettura_bt = bt_HC05.ReadString();
     }

     Motori.Destra();
     delay(25);
```

```
    LetturaNFC = nfc_G1.Check_NFC();

    if (LetturaNFC != "Nothing") {
      bt_HC05.SendString(LetturaNFC);
      delay(100);
      Lettura_bt = bt_HC05.ReadString();
    }

    Motori.Fermo(100);
    Motori.Sinistra();
    delay(25);

    LetturaNFC = nfc_G1.Check_NFC();

    if (LetturaNFC != "Nothing") {
      bt_HC05.SendString(LetturaNFC);
      delay(100);
      Lettura_bt = bt_HC05.ReadString();
    }

    Motori.Fermo(10);
  }

  if (Lettura_bt == "end") {
    Motori.Indietro();
    delay(1000);
    Motori.Tre_Giri();
    Motori.StopLoop();
  }
```

```
    Distanza_HCSR04();
  }

  // Memorizza il superamento attuale
  Vet_Superamenti[LoopState] = SuperamentoAttuale;

  // Analizza i superamenti passati
  AnalizzaSuperamenti();

  // Incrementa LoopState
  LoopState++;
  if (LoopState >= DIM) {
    LoopState = 0;
  }
}
```

Codice Piattaforma(creata)

```
// codice piattaforma G1
#include <G1LIBMiniCarRobot02.h>
#include <LiquidCrystal_I2C.h>
#include <ctime>

// Dichiarazione Badge da trovare
String Badge_str = "79:56:b0:f";

// Leds
int LedGiallo_pin = 8;
int LedVerde_pin = 7;

// Bluetooth pin incrociati
```

```
int pin_bt_Tx = 4;
int pin_bt_Rx = 2;

HC05 bt_HC05(pin_bt_Rx, pin_bt_Tx);

LiquidCrystal_I2C lcd(0x27,16,2);

Led LedGiallo(LedGiallo_pin);
Led LedVerde(LedVerde_pin);

void setup(void) {
  bt_HC05.bt_Setup();
  lcd.init();
  lcd.backlight();
  LedGiallo.AccendiLed();
  LedVerde.SpegniLed();
}

void loop(void) {
  lcd.setCursor(0,0);
  lcd.print("Waiting...");

  String Lettura_bt = bt_HC05.ReadString();

  bool cls = true;

  while (Lettura_bt == Badge_str) {
    bt_HC05.SendString("end");

    LedGiallo.SpegniLed();
```

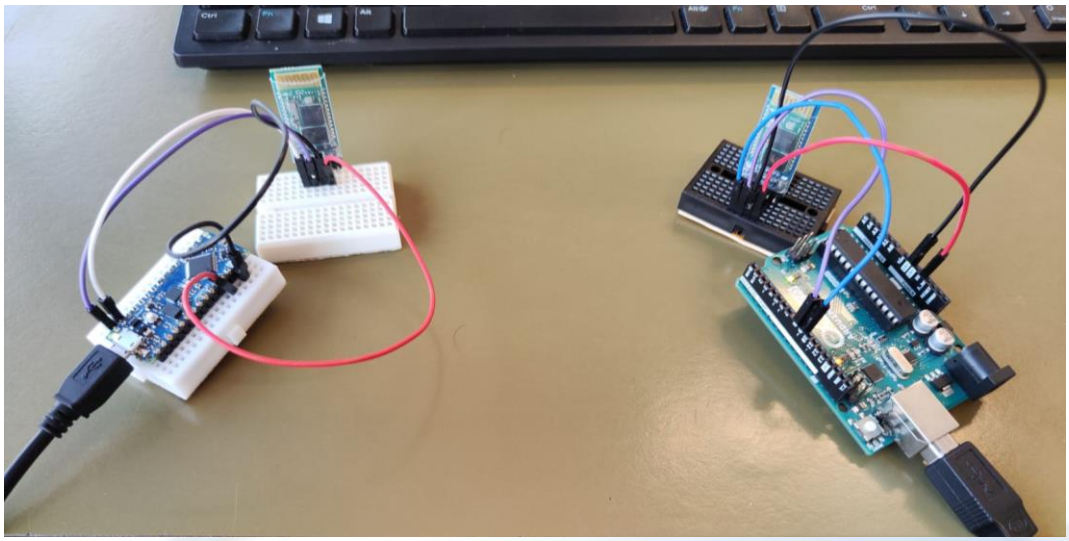
```
    LedVerde.AccendiLed();


    if (cls) {
      lcd.clear();
      cls = false;
    }


    lcd.setCursor(0,0);
    lcd.print("Trovato!");
    lcd.setCursor(0,1);
    lcd.print(Lettura_bt);
    Serial.print(Lettura_bt);
  }
}
```

PROVE FATTE DURANTE IL PROGETTO

Durante il periodo di progettazione del robottino, abbiamo eseguito diverse prove ogni volta che veniva completata una specifica parte del suo funzionamento, al fine di testarne l'efficacia e individuare eventuali errori.

Tra queste, abbiamo svolto test sulla comunicazione tra i due moduli Bluetooth, sul rilevamento e superamento degli ostacoli, nonché sulla funzionalità generale del robottino finale.

BLUETOOTH



Problemi scontrati all’inizio

- **Collegamento:** All’inizio dell’assemblaggio per la prova del bluetooth ci fu un errore nel collegamento che non permetteva il coretto funzionamento. Risolto poi in seguito.
- **Comunicazione:** Durante la fase delle prove del bluetooth è emerso il problema della comunicazione; infatti, fra i due bluetooth non avveniva un corretto “dialogo”. Risolto in seguito, si trattava di un problema di PIN scambiati uno per l’altro.

CODICE usato per la Prova

CODICE MASTER:

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(8, 9); // RX, TX

void setup() {
  Serial.begin(9600); // Monitor seriale
  BTSerial.begin(9600); // Assicurati che il baud rate sia corretto
  delay(1000); // Attendi la connessione
  Serial.println("Master pronto e in attesa di inviare messaggi...");
}

void loop() {
  // Invia la stringa al dispositivo slave
  BTSerial.println("Hello from Master!");
```

```
  Serial.println("Stringa inviata: Hello from Master!");
  delay(2000); // Aspetta 2 secondi prima di inviare di nuovo

  // Controlla se ci sono messaggi in arrivo
  if (BTSerial.available()) {
    String response = BTSerial.readStringUntil('\n'); // Legge la risposta fino a newline
    Serial.print("Risposta dallo slave: ");
    Serial.println(response);
  }
}
```

CODICE SLAVE:

```
#include <SoftwareSerial.h>

#define RxPin 12
#define TxPin 11
SoftwareSerial bt(RxPin, TxPin);

void setup() {
  Serial.begin(9600);
  bt.begin(9600); // Assicurati che il baud rate sia corretto
}

void loop() {
  // Controlla se ci sono messaggi in arrivo
  if (bt.available()) {
    String receivedMessage = bt.readStringUntil('\n');
    Serial.println("Received: " + receivedMessage);

    // Invia una risposta
```

```
    String responseMessage = "Hello from Slave!";
    bt.println(responseMessage);
    Serial.println("Sent: " + responseMessage);
  }

  delay(1000); // Aspetta 1 secondo prima di controllare di nuovo
}
```

Link del Video sulla Prova dei moduli Bluetooth

Clicca QUI

RILEVAMENTO OSTACOLO

Nel processo di sviluppo del sistema di rilevamento ostacoli, abbiamo effettuato diversi cambiamenti ai sensori utilizzati a causa dei problemi riscontrati durante le prove:

1. **Tre sensori a ultrasuoni → Sensore IR Sharp**

   Questo cambiamento è stato necessario perché, utilizzando tre sensori a ultrasuoni, si verificava un'interferenza: un sensore riceveva il segnale emesso da un altro sensore, causando errori nel rilevamento degli ostacoli. Per questo motivo, abbiamo deciso di sostituirli con un sensore IR Sharp.

2. **Sensore IR Sharp → Sensore IR Sharp GP2Y0D805Z0F**

   Successivamente, abbiamo sostituito il primo sensore IR Sharp con il modello Sharp GP2Y0D805Z0F. La scelta è stata motivata dal fatto che il sensore originale aveva un raggio di rilevamento troppo lungo rispetto alle nostre esigenze, mentre il modello GP2Y0D805Z0F, con il suo raggio di rilevamento minore, si adattava meglio alle specifiche del sistema.

Link dei Video sulla Prova del superamento ostacoli

LINK 1 SUPERAMENTO OSTACOLO

LINK 2 SUPERAMENTO OSTACOLO

Rilevamento BADGE con sensore NFC PN 532

Non si sono creati problemi nelle prove sul rilevamento del badge con il sensore NFC

Segue il Link sulla prova fatta:

LINK SUL RILEVAMENTO

PIANO ATTIVITA

DATA		n° ore lezione	Argomenti	Allievo/i
23/10/2024	mercoledì	1	a) Relazione(phototransistor) b) Ideazione circuito c) Prova bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
25/10/2024	venerdì	2	a) creazione piano attività b) risoluzione alimentazione c) finalizzazione prove bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
30/10/2024	mercoledì	1	a) ricerca componenti progetto b) Continuazione robottino c) Studio piattaforma bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
06/11/2024	mercoledì	1	a) continuazione relazione(componenti) b) Continuazione robottino c) Studio funzionamento bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
08/11/2024	venerdì	2	a) analisi teorica relazione b) test movimenti robottino c) test slave bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
13/11/2024	mercoledì	1	a) analisi componenti b) revisione del codice c) analisi master bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
15/11/2024	venerdì	2	a) analisi componenti b) test phototransistors c) test master bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
20/11/2024	mercoledì	1	a) revisione relazione b) verifica della sorgente c) verifica bluetooth sorgente	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
22/11/2024	venerdì	2	a) ampliamento relazione b) test RFid c) bluetooth bidirezionale	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
27/11/2024	mercoledì	1	a) verifica della relazione b) revisione codice RFid c) analisi comunicazione bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
29/11/2024	venerdì	2	a) miglioramento relazione b) test 1 autonomia robottino c) test 1 autonomia bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo

04/12/2024	mercoledì	1	a) continuazione relazione(componenti) b) revisione codice test 1 c) revisione test 1 autonomia bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
06/12/2024	venerdì	2	Controllo generale del progetto e consegna completa remota su GitHub	Zhou Luigi; Simone Vidotti; Grecu Lorenzo
11/12/2024	mercoledì	1	a) revisione relazione completa b) revisione codice sorgente c) revisione codice bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
13/12/2024	venerdì	2	Ulteriore verifica generale e rispettive risoluzioni	Zhou Luigi; Simone Vidotti; Grecu Lorenzo
18/12/2024	mercoledì	1	a) revisione relazione completa b) revisione codice sorgente c) revisione codice bluetooth	a) Zhou Luigi b) Simone Vidotti c) Grecu Lorenzo
20/12/2024	venerdì	2	Collaudo generale del progetto e consegna del progetto	Zhou Luigi; Simone Vidotti; Grecu Lorenzo

GITHUB

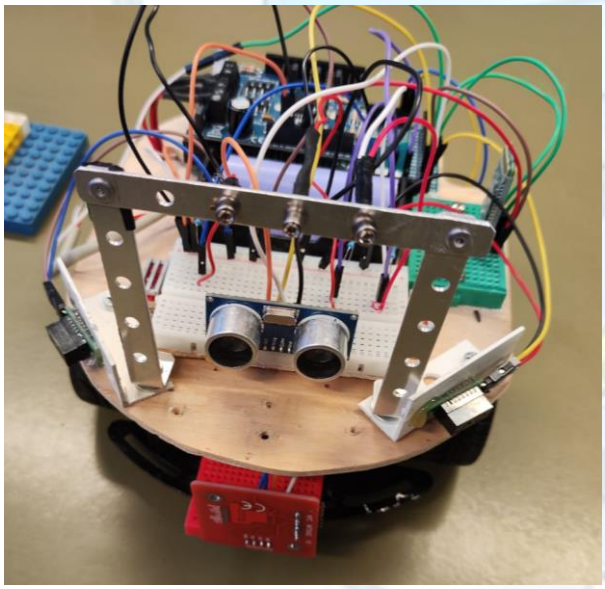
Nei seguenti link di GITHUB è possibile accedere a tutto il materiale utilizzato e sviluppato durante questa fase di progettazione e sperimentazione. All'interno del repository troverete i codici sorgente utilizzati per il funzionamento del robottino, la documentazione sulle prove effettuate e i file di configurazione necessari per replicare o approfondire il progetto.

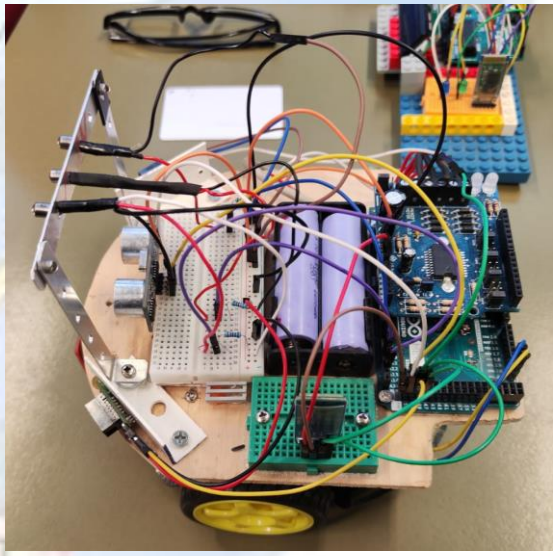
LINK GITHUB

LINK GITHUB G1 PROGETTO 1 2024/25

FOTO Robottino e Piattaforma di Controllo

FOTO del ROBOTTNO:





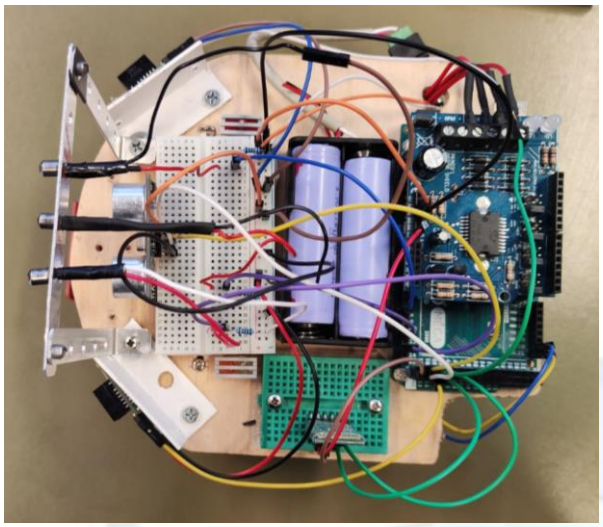
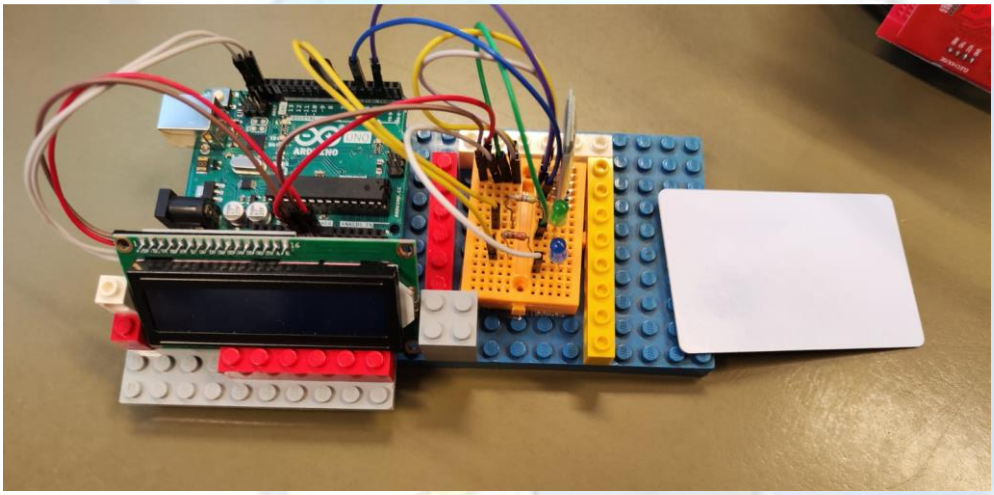


FOTO della PIATTAFORMA CREATA:



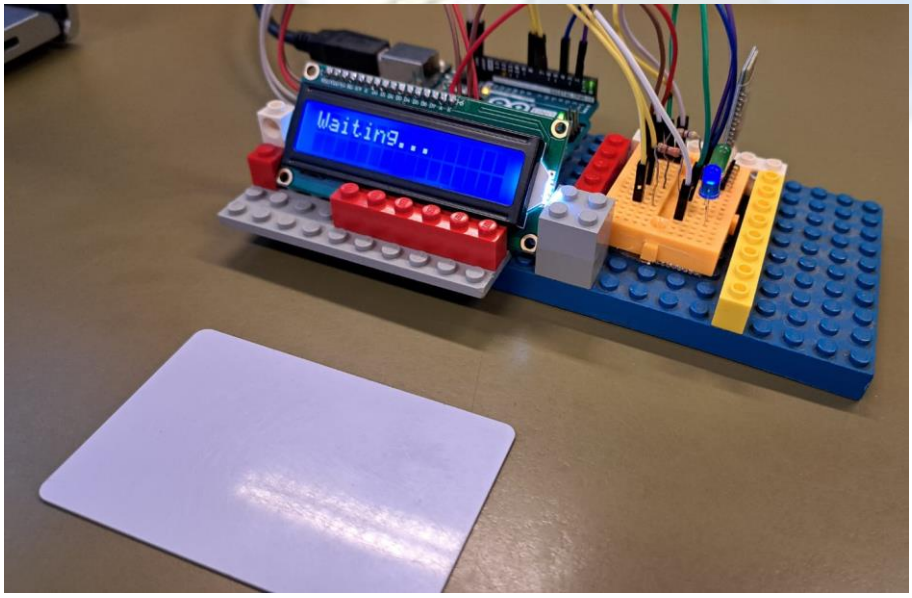
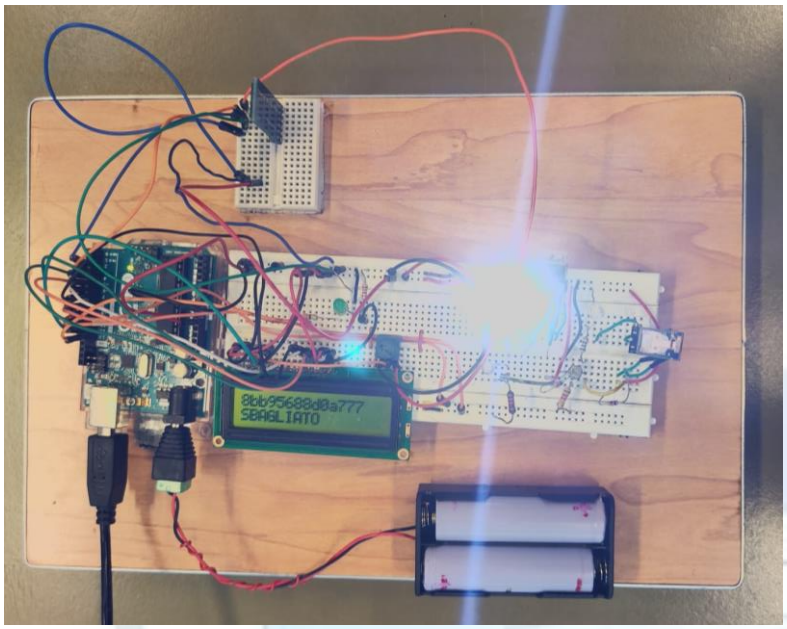
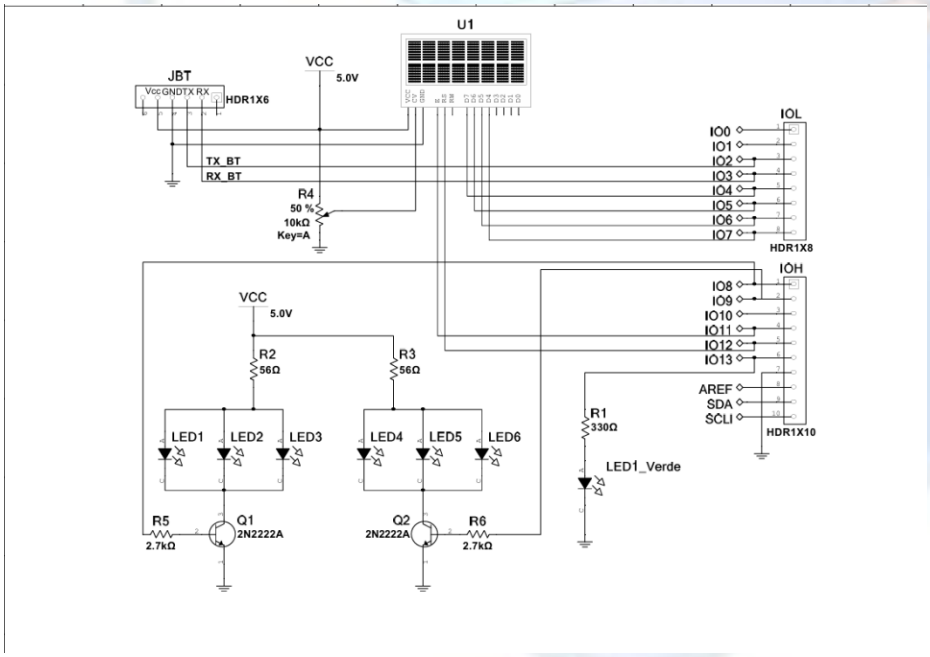


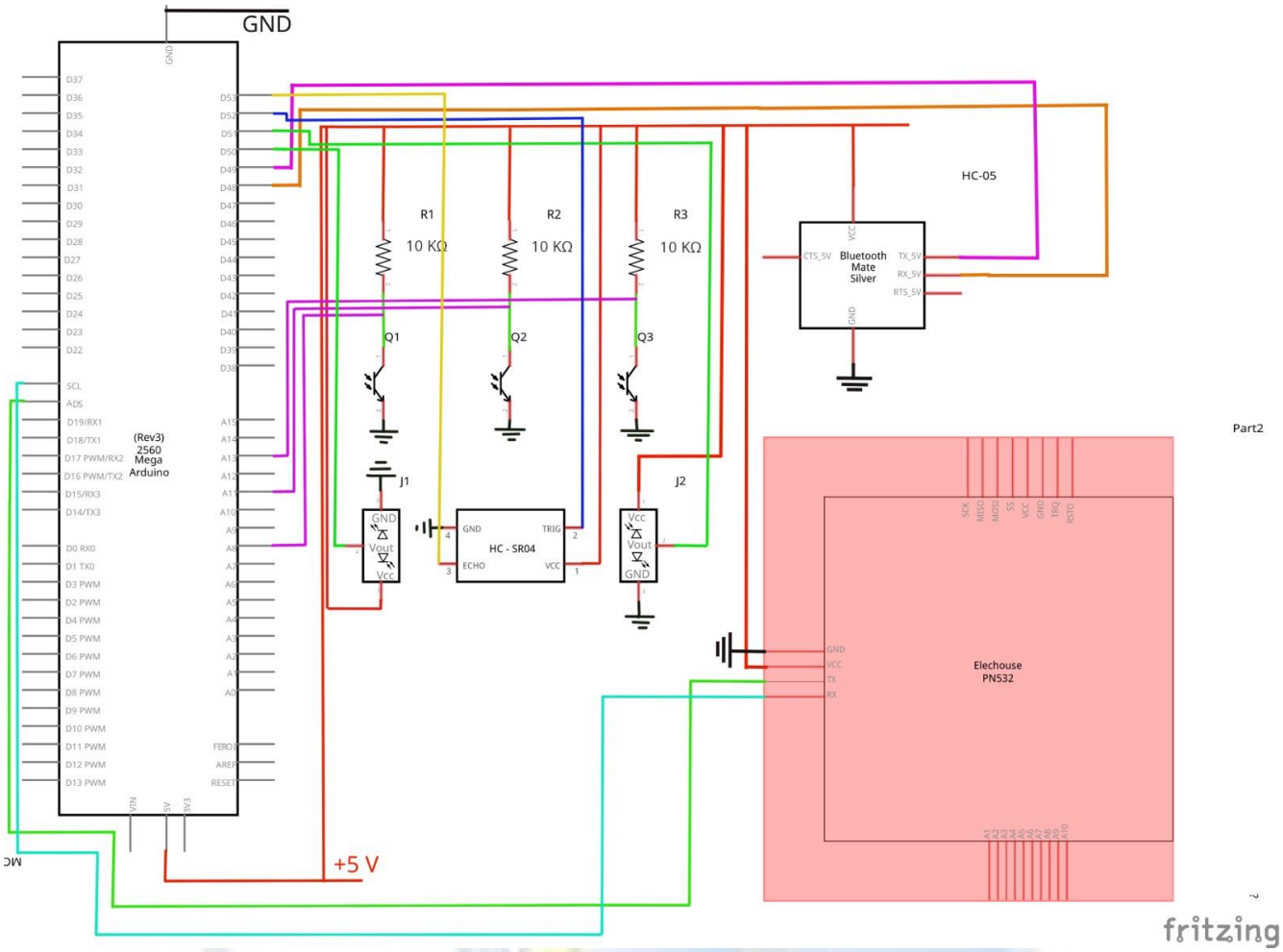
FOTO della PIATTAFORMA SCOLASTICA:



SCHEMA ELETTRICO DELLA PIATAFORMA SCOLASTICA:



Schema Elettrico:



CONCLUSIONE

In conclusione, il progetto ha rappresentato una sfida complessa ma altamente formativa, permettendoci di sviluppare competenze nella progettazione, nella programmazione e nel lavoro di squadra. Attraverso l'integrazione di sensori, attuatori e moduli di comunicazione, il robot ha dimostrato una capacità operativa parzialmente autonoma, raggiungendo buona parte degli obiettivi prefissati, come la ricerca delle fonti luminose, l'evitamento degli ostacoli e la comunicazione con il sistema di controllo remoto.

Le prove effettuate hanno evidenziato la necessità di migliorare l'affidabilità del sistema per gestire errori e malfunzionamenti, stimolando una continua ricerca di soluzioni creative. Nonostante le difficoltà incontrate, tra cui problemi nel rilevamento degli ostacoli e nella comunicazione Bluetooth, siamo riusciti a risolverle autonomamente, affinando le nostre capacità di problem solving e rafforzando lo spirito di collaborazione.

Questa esperienza ci ha permesso di acquisire una conoscenza pratica sulla progettazione e lo sviluppo robotico, fornendoci basi per affrontare progetti futuri in modo più consapevole e competente.