Costruire un robot mobile con Arduino UNO rev

Autori: classe IV BEA A.S. 2013/14

doc. G.Drei

3. Controllo della velocità e della direzione

3.1 Il controllo dei motori

3.1.1 La tecnica PWM

3.1.2 Il ponte H

3.1.3 Come controllare la velocità e la direzione dei motori

3.2 Le MOTOR-shield

3.2.1 Ardumoto la motor shield della Sparkfun

3.2.2 Arduino motor shield

3.3 Pilotaggio dei motori

3.4 Le funzioni per il movimento del robot

4. Gestione dei sensori

4.1 Il sensore di Luce: la fotoresistenza

4.2 sensore di luminosità Analog line sensor CNY70

4.2.1 Inseguimento di una linea

4.3 sensore di distanza ad infrarossi SHARP 2d120x f 2y

4.4 sensore di temperatura MLX90614

4.5 Il sensore ad Ultrasuoni SFR05

4.5.1 Il controllo della distanza

5. Collegamento all'I2C bus dei sensori

5.1 Il protocollo I2C

5.2

1. Introduzione

“ARDUINO IS AN OPEN-SOURCE ELECTRONICS PROTOTYPING PLATFORM BASED ON FLEXIBLE, EASY-TO-USE HARDWARE AND SOFTWARE. IT'S INTENDED FOR ARTISTS, DESIGNERS, HOBBYISTS AND ANYONE INTERESTED IN CREATING INTERACTIVE OBJECTS OR ENVIRONMENTS.”

www.Arduino.cc

Arduino è un sistema di prototipazione elettronica open-source prodotto ad opera di un team di sviluppatori composto da: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, e David Mellis.

In termini più semplici, Arduino è un insieme hardware e software comprendente diversi modelli di scheda hardware, un ambiente di sviluppo software completamente Open-Source e una comunità di utilizzatori pari all' enorme successo che questo sistema ha avuto in tutto il mondo (10000 schede vendute solo nel primo anno).

1.1 HARDWARE

Il sistema Arduino è costituito per parte Hardware da diverse schede a microcontrollore prodotte per la maggior parte in Italia.

Tra i vari modelli trattiamo l' Arduino UNO rev , il più comune tra le diverse *release*.

1.1.1 MICROCONTROLLORE

Arduino UNO monta un microcontrollore ATMega328P (datasheet: http://www.atmel.com/Images/doc8161.pd) dotato di una memoria flash da 32KB (corrispondenti a 32.768 Bytes) disponibile per immagazzinare il programma e le eventuali librerie. É altresì dotato di una RAM seriale da 64 Bytes. Il clock del processore è fornito da un oscillatore XTAL al quarzo con frequenza di 16MHz.

1.1.2 PIN di I/O

La scheda arduino UNO rev è dotata di:

- 14 PIN digitali
  - di cui 6 dotati di controllo PWM
- 6 PIN analogici

Tutte le connessioni sono effettuate mediante connettori DIN-SIL femmina da 0.1", questo permette l'applicazione di schede integrative Plug&Play dette *Shield*. (aggiungere foto di shield)

1.1.2.1 PIN Digitali

I PIN sono numerati da 0 a 13, con un simbolo (~) per i PIN con disponibilità per il controllo PWM.

1.1.2.2 PIN Analogici

I 6 PIN analogici, numerati da A0 a A6, sono campionati mediante un ADC a 6 canali, 10 bit di risoluzione; implementato direttamente nel microcontrollore.

1.2 Dialogo Arduino <-> PC

Il microcontrollore ATMel 328P è programmabile via comunicazione seriale con protocollo RS232. I dati vengono convertiti da un chip ATMega 8U2 in comunicazione TTL per USB.

1.3 SOFTWARE

1.3.1 Arduino IDE

Arduino adotta un software di programmazione Open-Source chiamato Arduino IDE (*Integrated Development Environment*), scritto in linguaggio Java; l'ultima *release* prodotta è la versione 1.0.5 ed è scaricabile gratuitamente all'indirizzo: http://arduino.cc/en/Main/Software

1.3.2 Linguaggio di programmazione

Arduino viene programmato prevalentemente in C/C++ dedicato……..; essendo l'intero sistema pensato per neofiti e dilettanti dell'elettronica e della programmazione, si richiede all'utente di definire solo le due funzioni:

- `setup()`

– funzione chiamata una sola volta all'inizio di un programma che può essere utilizzata per i settaggi iniziali;

- `loop()`

– funzione chiamata ripetutamente, la cui esecuzione si interrompe solo con lo spegnimento della scheda.

Potendo essere programmato in C/C++ dedicato, Arduino IDE permette una programmazione relativamente semplice e veloce; oltre a riconoscere le strutture normali del C/C++, Arduino IDE possiede alcune funzioni proprie tra le quali si evidenziano:

- pinMode(); http://arduino.cc/en/Reference/pinMode
- digitalWrite(); http://arduino.cc/en/Reference/digitalWrite
- digitalRead(); http://arduino.cc/en/Reference/digitalRead
- analogWrite(); http://arduino.cc/en/Reference/analogWrite
- analogRead(); http://arduino.cc/en/Reference/analogRead
- delay(); http://arduino.cc/en/Reference/delay

- map(); http://arduino.cc/en/Reference/map

1.3.2.1 Funzione pinMode();

```
pinMode(pin, INPUT);
```

Definisce la direzione dei dati Binari in un dato pin; accetta variabili int per il numero del pin e i *define* "INPUT" e "OUTPUT" per la direzione dei dati.

1.3.2.2 Funzione digitalWrite();

```
digitalWrite(pin, HIGH);
digitalWrite(pin, data);
```

Modifica il valore di un pin digitale sia mediante i *define* "HIGH" e "LOW" , sia mediante il passaggio diretto di una variabile.

1.3.2.3 Funzione digitalRead();

```
digitalRead(pin);
int value = digitalRead(pin);
```

Legge il livello logico presente sul pin digitale e ne ritorna il valore binario.

1.3.2.4 Funzione analogWrite();

```
analogWrite(pin, data);
```

Modifica il valore di un pin analogico utilizzando una variabile con valore compreso tra 0 e 1023. (risoluzione 10bit)

1.3.2.5 Funzione analogRead();

```
int value = analogWrite(pin);
```

Legge il valore di un pin analogico ritornando in una variabile un valore compreso tra 0 e 1023. (risoluzione 10bit)

1.3.2.6 Funzione delay();

```
delay(2600);
delay(value);
```

Sospende l'esecuzione del programma per il tempo in millisecondi passato alla funzione. Nel primo esempio il programma si fermerà per 2.6 Secondi.

1.3.2.7 Funzione map();

```
trueValue = map(value, 12, 97, 0, 1023);
```

Converte un valore passato (value) entro due termini passati (12 e 97) ad un altro valore proporzionale compreso tra gli altri due termini (0 e 1023).

Come nel C/C++ per commentare una linea di codice in modo che non sia eseguita dal processore si utilizza il segno '//'; allo stesso modo le istruzioni del pre-processore sono precedute dal carattere '#' e presentano la stessa sintassi.

La programmazione viene assistita dalle *utility* dell' IDE:

- Indentazione automatica
- Controllo parentesi
- *Syntax Highlighting* (evidenzia automaticamente le parole riconosciute dal compilatore)

Arduino è inoltre programmabile da diversi IDE tra i quali:

- Minibloq (http://blog.minibloq.org/p/download.html)
- Android Accessory Development Kit (http://accessories.android.com/)

1.3.3 Compilazione e Upload

Sia la compilazione che l'upload del programma alla Arduino Board vengono effettuate dall' IDE. Il compilatore inoltre aggiunge le seguenti righe:

```
int main(void)
{
  init();

  setup();

  for (;;)
```

```
    loop();

  return 0;
}
```

che permettono al Microcontrollore di riconoscere le funzioni `setup();` e `loop();`

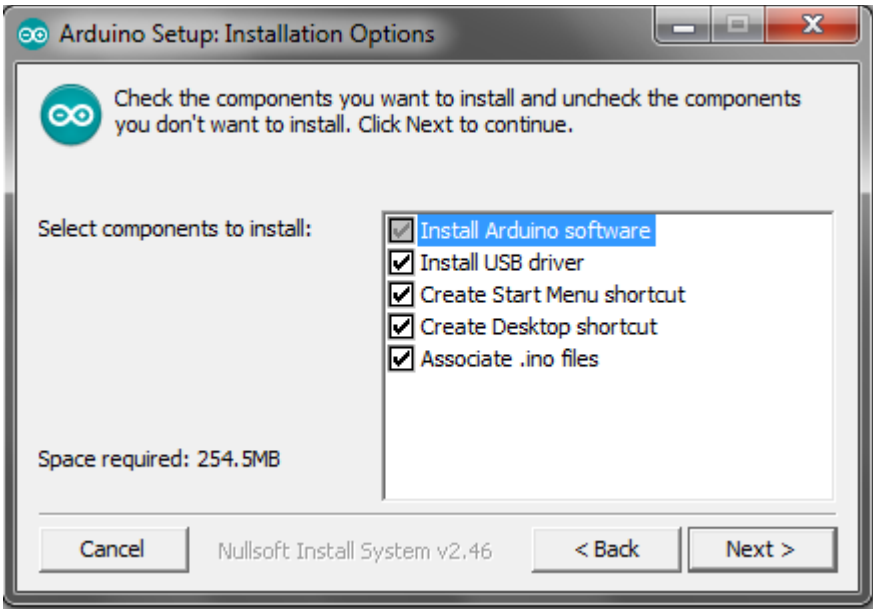
1.4 Installazione di Arduino IDE

1.4.1 Download

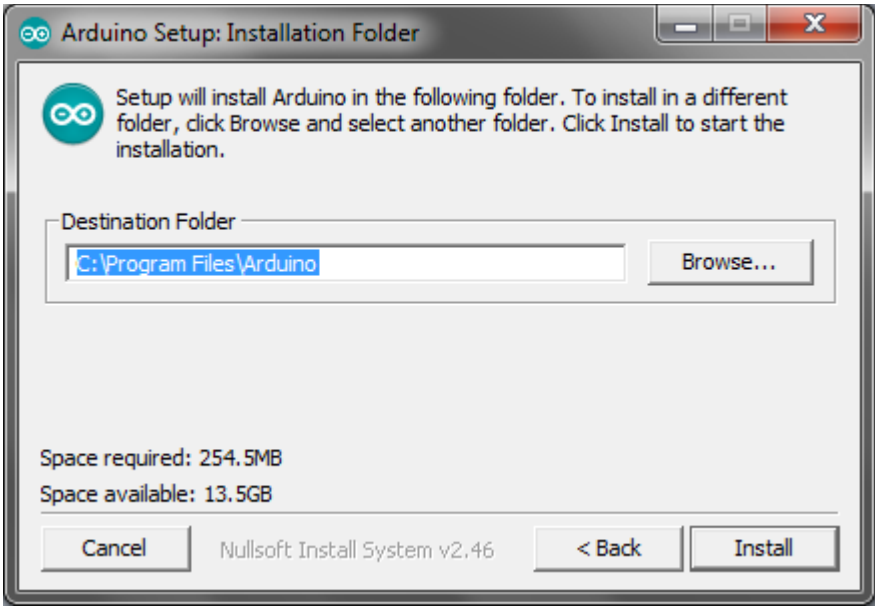
Arduino IDE è disponibile gratuitamente all'indirizzo http://arduino.cc/en/Main/Software
la versione 1.0.5 è disponibile sia per Windows, Mac OS che per Linux.

1.4.2 Installazione

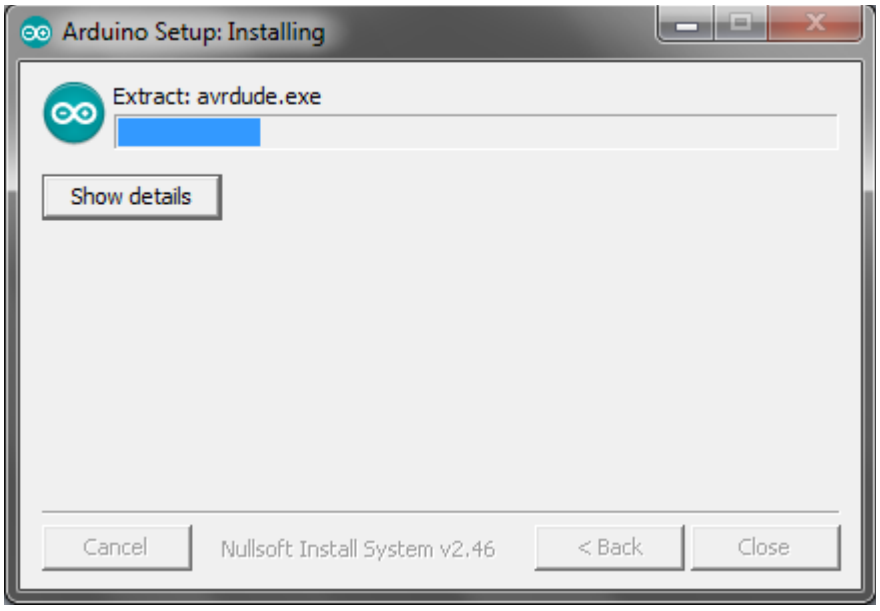
Una volta eseguito il download lanciando il programma installer si otterà la seguente schermata:



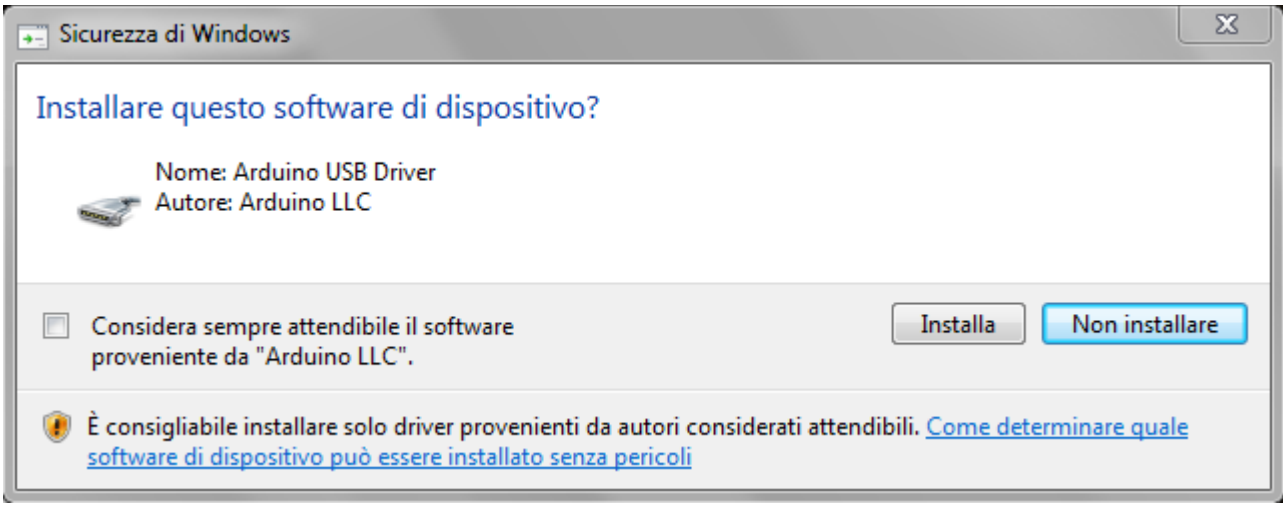
è possibile scegliere quali componenti o caratteristiche installare.
Per un corretto funzionamento dell' IDE è consigliato installare almeno i primi due elementi.



In questa finestra è possibile scegliere la *directory* dove verrà installato il programma.



Il processo di installazione durerà alcuni minuti.



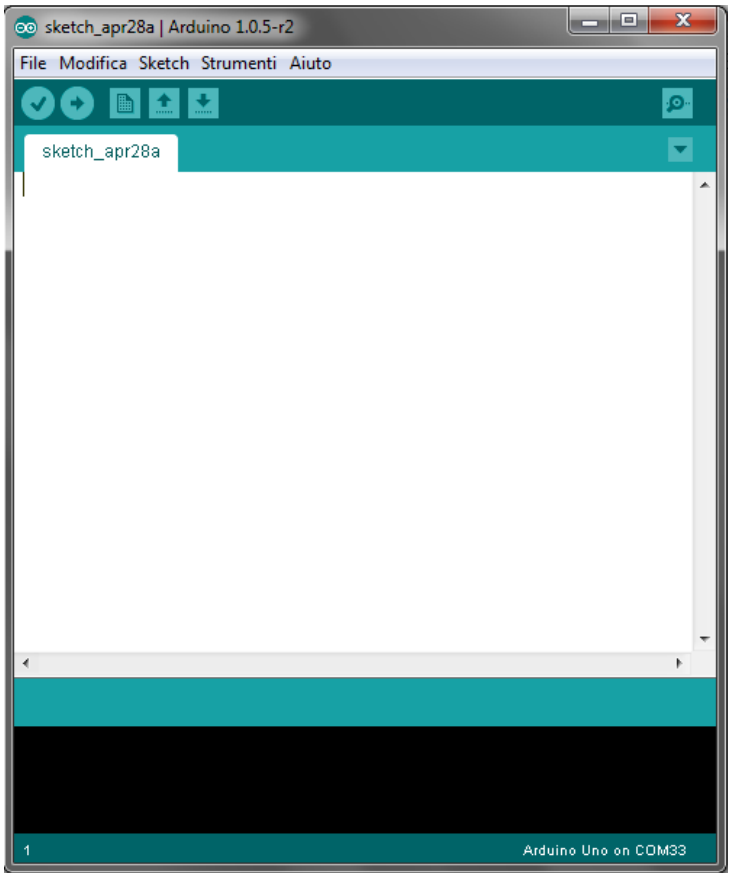
Al termine dell'installazione Windows chiederà se si vuole installare il Driver USB di Arduino.

1.4.3 Collegamento al PC

Il collegamento di arduino al PC viene effettuato mediante cavo USB-A <-> USB-B, non incluso nelle confezioni di Arduino tranne che negli Starter Kit

1.5 Introduzione all'uso di Arduino IDE

1.5.1 Presentazione



L' IDE all' avviamento si presenta come un editor di testo con i caratteristici colori di Arduino.

1.5.1 Pulsanti

I pulsanti dell'IDE, da sinistra a destra:

- **Compila** compila il programma e individua eventuali errori.
- **Compila e scarica** compila il programma e lo scarica su arduino.
- **Nuovo** nuovo file .ino
- **Apri** apre un file .ino già esistente
- **Salva** salva il programma in un file .ino
- **Monitor Seriale** apre il monitor seriale, per controllare le comunicazioni tra Arduino e PC.

1.5.2 Menu a tendina

- **File**
  - **Nuovo** crea un nuovo file .ino
  - **Apri…** apre un file .ino esistente
  - **Cartella degli Sketch** mostra la cartella dei programmi (C:\Utenti\Utente\Documenti\Arduino)
  - **Esempi** mostra i programmi di esempio di Arduino
  - **Chiudi** chiude il file aperto
  - **Salva**
  - **Salva con Nome…**
  - **Carica** scarica il file su Arduino
  - **Carica con un Programmatore** scarica il file su arduino utilizzando un compilatore differente
  - **Imposta pagina**
  - **Stampa**
  - **Preferenze**
  - **Esci** chiude l'IDE
- **Modifica**
  - **Annulla**
  - **Ripeti**
  - **Taglia**
  - **Copia**
  - **Copia per il Forum** copia il testo selezionato come quote per il forum
  - **Copia come HTML** copia il testo selezionato come codice HTML
  - **Incolla**
  - **Seleziona Tutto**
  - **Commenta / Togli commento** Aggiunge o toglie il simbolo '//' che introduce ad un commento nel programma
  - **Aumenta indentazione**
  - **Diminuisci Indentazione**
  - **Trova…**
- **Sketch**
  - **Verifica/compila** compila il programma e segnala gli errori
  - **Mostra cartella dello sketch** mostra la cartella del programma, contenente tutti i file e le librerie
  - **Aggiungi file** aggiunge un file di codice (.c/.cpp/.h/.lib) al programma
  - **Importa libreria** Importa una libreria dalle cartelle dell'IDE
- **Strumenti**

- **Formattazione Automatica** converte il testo nel formato standard
- **Archivia Sketch** permette di salvare il file con estensione a piacere
- **Tipo di Arduino** permette di scegliere il tipo di arduino collegato
- **Processore** permette di scegliere il processore
- **Porta Seriale** consente di scegliere a quale porta seriale è collegato Arduino
- **Programmatore** Consente di scegliere con quale tra i programmatori installati compilare il programma
- **Scrivi il Bootloader** Apre l'editor per la scrittura delle procedure di carico durante il Boot di Arduino.

1.5.3 File inclusi

Con l'installazione standard vengono copiati nella cartella del programma (C:\Programmi\Arduino\Arduino IDE) i file di alcune librerie e degli esempi.

Librerie incluse:

- EEPROM Permette l'uso di arduino come programmatore per EEPROM
- Ethernet Abilita arduino a funzioni di rete mediante la Shield Ethernet
- Firmata Permette l'uso delle funzioni del protocollo Firmata
- GSM Inserisce funzioni per l'utilizzo di comunicazioni GSM
- LiquidCrystal Permette l'uso di Display LCD
- SD Permette l'uso della SD Shiedl
- Servo Controlla facilmente Servomotori
- SPI Abilita il protocollo SPI su Arduino
- Stepper Controlla facilmente Motori Passo-passo
- WiFi Permette l'uso della WiFi-Shield o di moduli ZigBee o WiBee

1.5.4 Esempio di sketch di Arduino completo:

```
//OBIETTIVO: muovere un Servomotore di 180°C in base al movimento di una manopola


#include <servo.h>                    //Include nel programma la libreria servo.h che permette
                                      //il controllo dei servi
servo myservo;                        //Creazione dell'oggetto "myservo" per la libreria

void setup()                          //Funzione Setup -- Chiamata solo all'inizio del programma
{
   Serial.begin(9600);                //Inizializzazione della comunicazione seriale tra Arduino e PC
```

```
                                        //con una velocità di 9600 [baud] (bits per seconds)
    Serial.println("Setup");            //Stampa su monitor seriale la stringa "Setup"
     pinMode(9, OUTPUT);                        //Definisce il Pin 9 (PWM) come uscita per il servomotore
    pinMode(13, OUTPUT);                //Definisce il pin 13 come uscita per il LED (incluso sulla scheda)
    myservo.attach (9);                 //Associa il Pin 9 all'oggetto "myservo"
}

void loop()                             //Funzione Loop -- Esegue le istruzioni in ciclo perpetuo
{
    digitalWrite(13, HIGH);             //accende il led

    val = analogRead(potpin);           //legge il valore analogico dal potenziometro
    val = map(val, 0, 1023, 0, 179);    //forza il valore da 0 a 179

    myservo.write(val);                 //Comunica il valore al servomotore

    delay(1000);                        //lampeggia il led
    digitalWrite(13,LOW);
    delay(1000);
}
```

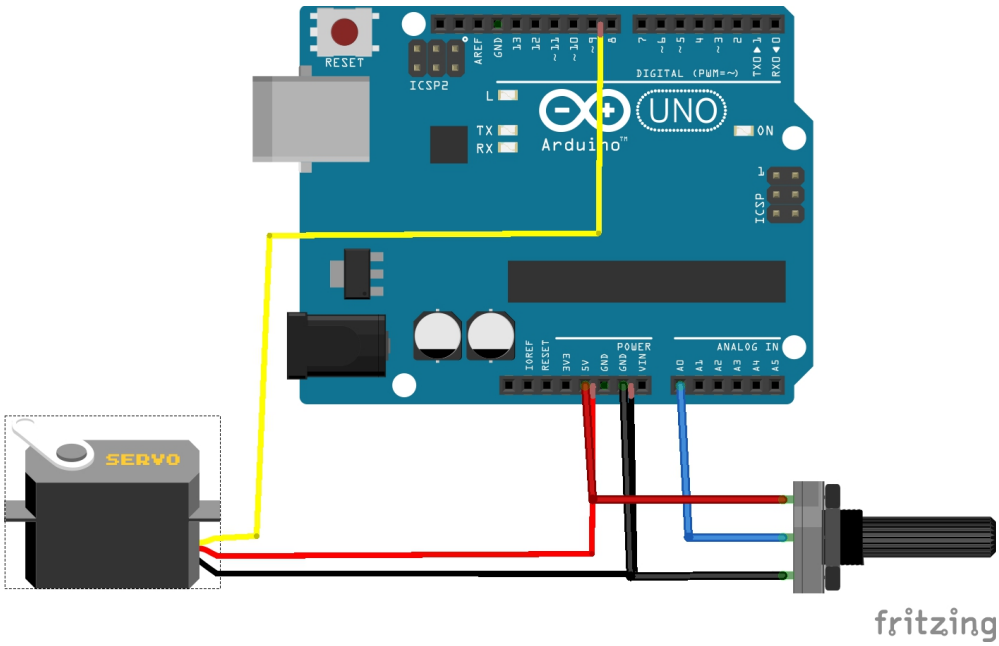
2. **Costruire la piattaforma mobile a due ruote (Autore: PASTORINO)**

2.1 **HW utilizzato**

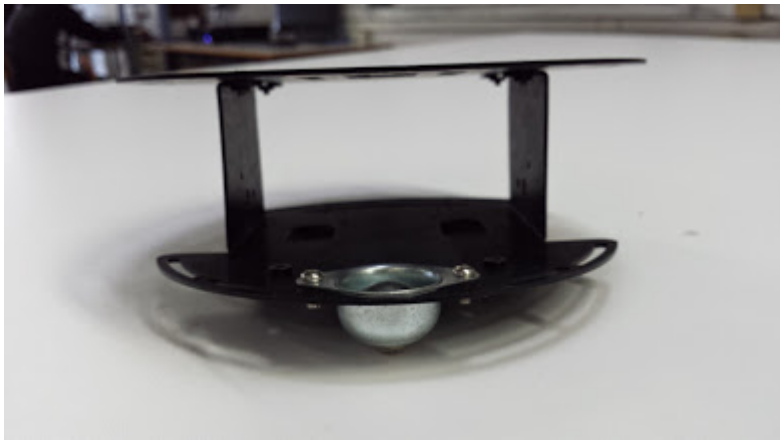
shield Arduino uno

shield Ardumoto (https://www.sparkfun.com/products/9815)

Turtle -2WD Mobile Platform Kit comprensivo di 2 motori DC a Magneti Permanenti



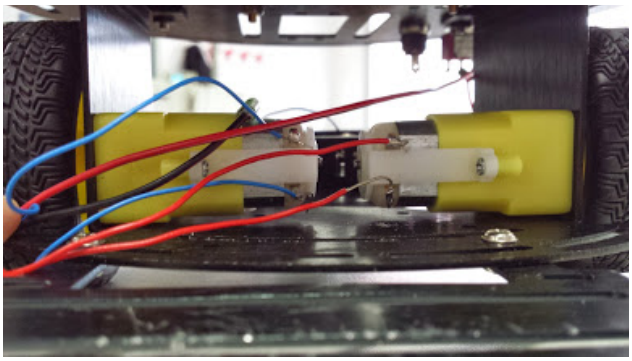
2.2 **Assemblaggio meccanico**

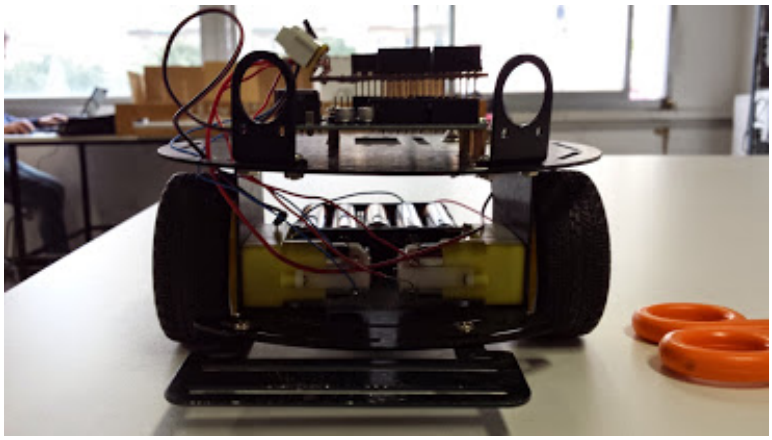


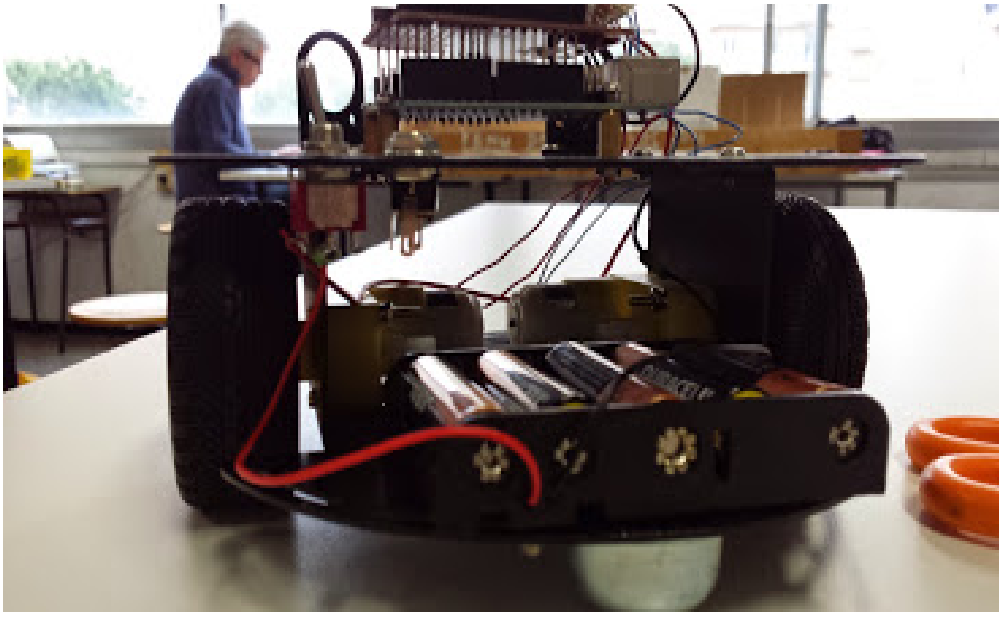
Per prima cosa si montano i 2 motori DC , forniti insieme al kit Turtle -2WD Mobile Platform, per

le due ruote, successivamente si monta il contenitore di plastica delle 5 pile AA. Questi tre elementi vanno inseriti nella parte inferiore della struttura in alluminio del robot . Sulla parte superiore si monta la scheda Arduino Uno e sopra ad essa va posta la shield Ardumoto , sopra a quest' ultima va posta una proto-shield per la gestione degli IN/OUT . Viene inserito un pulsante (ON/OFF) per l' alimentazione delle shield per evitare inutili sprechi di corrente e tensione.

Per ogni motore sono necessari due cavi (fasi) che sono collegati alla ardumoto shield.



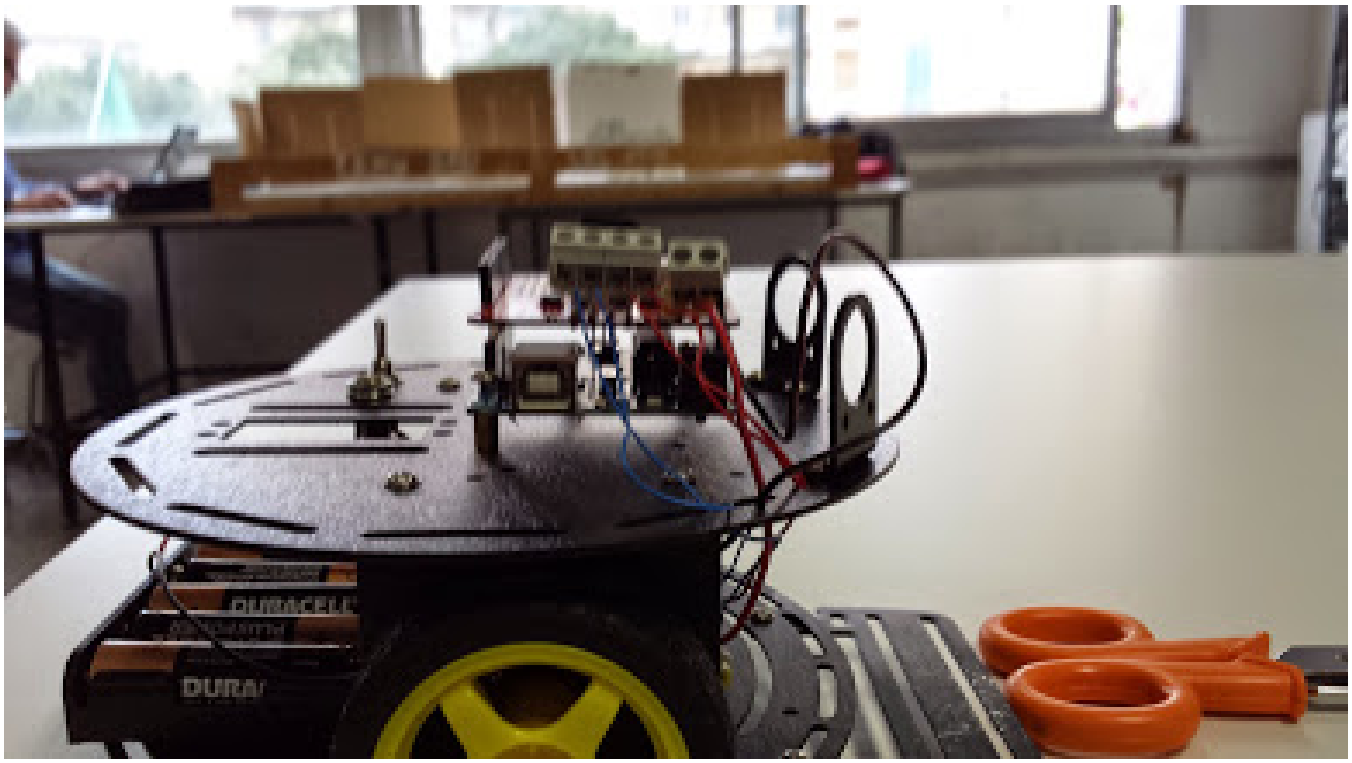




2.3 **Alimentazione Arduino + Ardumoto + protoshield (in/out)**

Per il collaudo del robot:

- Arduino è stato alimentato via USB- USB lato PC ;
- Ardumoto è stata alimentata via 5 pile da 1.5 V (7.5V) , tensione questa necessaria per far funzionare i due motori che assorbono 0,5 A come corrente di spunto e poi ne consumano a regime circa 0,3-0,4 A; per un solo motore occorrono 0,3 A di spunto e 0,1-0,2 a regime;
- la protoshield (IN/OUT) viene alimentata da Arduino



2.4 Montaggio dei sensori

E' possibile utilizzare una proto-shield di Input/Output per il collegamento dei sensori agli input di Arduino per rendere più semplici i collegamenti tra la scheda Arduino e i sensori , vedi fig .

2.4.1 Sensori di LUCE

devono essere montati a pochi millimetri da terra per poterli far funzionare al meglio. Questi sensori hanno 3 piedini : uno per la massa, uno per Vcc e uno per il segnale analogico. Il piedino della massa va collegato al ground della shield, nella quale sono presenti 2 piedini differenti per le masse, il segnale analogico va inserito in uno dei 6 ingressi analogici, la Vcc è da 5 V e per far funzionare entrambi i sensori si è dovuto creare un ponte dall' uscita dei 5 V per far in modo che entrambi fossero alimentati.

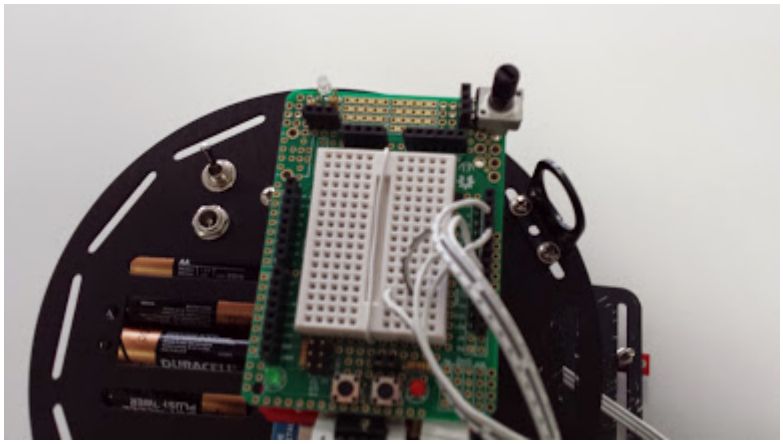


Fig.

faRE SCHEMA DI COLLEGAMENTO CON FRITZING

3. Controllo della velocità e della direzione

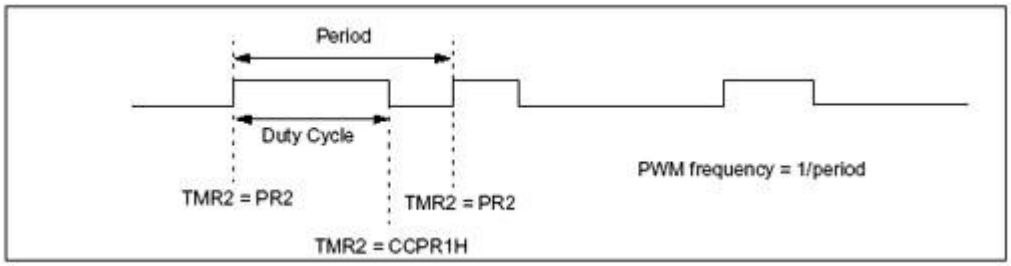
3.1 il controllo dei motori

I motori utilizzati sono a corrente continua a magneti permanenti. Sono dotati di due terminali collegati alle spazzole (contatti). La direzione del motore (Avanti/Indietro) può essere invertita invertendo la polarità dell'alimentazione. Per controllare la velocità del motore si adotta la tecnica PWM e per controllare la direzione di due motori è necessario l'uso di un ponte H .

3.1.1 la tecnica PWM (AUTORI: Ameglio-Vallarino-Albani)

Cos'é un segnale pwm? (akizukidenshi.com/download/**PIC**16F877A.pdf)

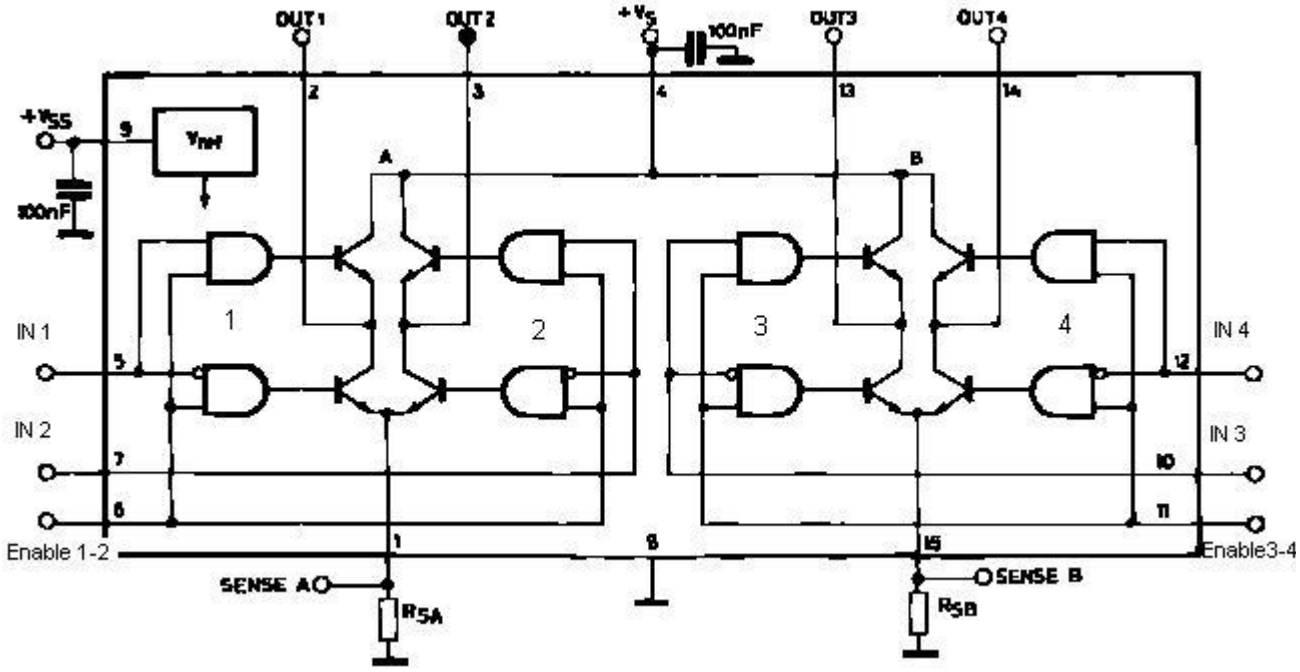
Un segnale PWM (Pulse Width Modulation ovvero modulazione a variazione della larghezza d'impulso) è un' onda quadra di duty cycle variabile che permette di controllare l'assorbimento (la potenza assorbita) di un carico elettrico(nel nostro caso il motore DC), variando (modulando) il duty cycle.



Un segnale PWM è caratterizzato dalla frequenza (fissa) e dal duty cycle (variabile); come si deduce dalla Figura , il duty cycle è il rapporto tra il tempo in cui l'onda assume valore alto e il periodo T (l'inverso della frequenza: T=1/f) ne segue che un duty cycle del 50% corrisponde ad un'onda quadra che assume valore alto per il 50% del periodo, un duty cycle dell'80% corrisponde ad un'onda quadra che assume valore alto per l'80% del periodo e basso per il restante 20%, un duty cycle del 100% corrisponde ad un segnale sempre alto e un duty cycle dello 0% ad un segnale sempre basso (come vedremo anche questi ultimi due casi non sono del tutto inutili).

3.1.2 Il ponte H (http://www.tmasi.com/robotica/pwmtut/pwmtut_1.htm)

Per pilotare due motori mediante un segnale PWM utilizziamo un integrato a ponte H: l' L298. Con l'introduzione del Ponte H abbiamo risolto il problema della rotazione del motore in entrambi i versi, mentre con il segnale PWM riusciamo a regolare la velocità di rotazione.

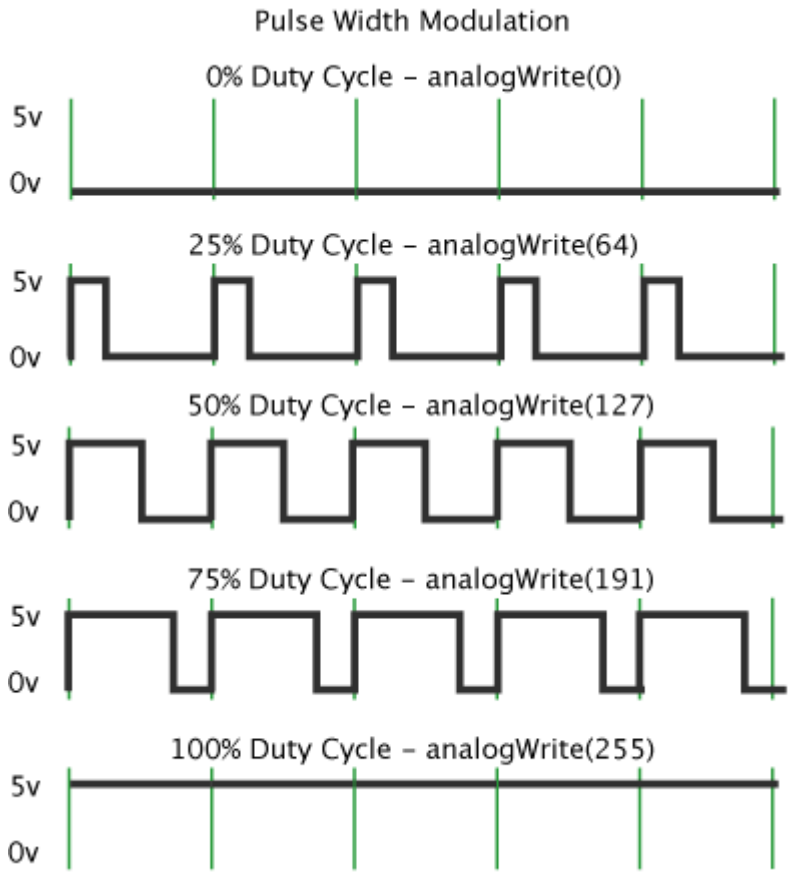


schema interno dell' IC L298

Tale integrato è costituito da quattro mezzi ponti H (numerati in figura come 1-2-3-4) ognuno dei quali è costituito da due transistor e da una logica combinatoria che li comanda in modo da poter accende solo uno alla volta: quando il transistor superiore di un mezzo ponte è in conduzione quello inferiore sarà necessariamente spento e viceversa. E' inoltre presente un comando di ENABLE che permette di inibire il funzionamento di una coppia di mezzi ponti.
Si può inoltre notare la presenza di due pin dedicati alla connessione di una resistenza di Sense per monitorare la corrente che scorre nel motore (utile per controllare l'assorbimento di corrente ed evitare rischiosi stalli del motore) ma tale caratteristica non è presente in tutti i ponti H integrati tra cui il SN754410 che verrà utilizzato nella scheda controllo motori presentata in seguito.
Ricapitolando, per ognuno dei due ponti presenti nell' integrato abbiamo a disposizione due ingressi di controllo per permettere il passaggio di corrente in un verso e un ingresso di ENABLE per accendere e spegnere il ponte: come vedremo a breve ci sono più modi di pilotare il ponte H e per comprenderli è necessario aver capito bene a cosa servono questi tre ingressi.

3.1.3 Come possiamo controllare la velocità dei motori

Come detto in precedenza il segnale PWM è un'onda quadra a cui è applicato un duty cycle variabile.
Un'onda quadra con un duty cycle è un'onda rettangolare cioè un'onda quadra con il livello alto differente da quello basso infatti il duty cycle o ciclo di lavoro è il rapporto tra lo stato alto e il periodo..
Avendo il livello alto differente da quello basso si possono ottenere effetti molto utili come ad esempio si può controllare la luminosità di un led, aumentando il valore del duty cycle la luminosità del led aumenta questo perchè il led rimane nello stato di alimentazione alto per un tempo maggiore. Un altro grande vantaggio è il consumo minore rispetto all'impiego di un segnale DC perchè fornendo in entrata un segnale con un duty cycle del 85% otterremo un' illuminazione pressoché identica o comunque con una differenza impercettibile per l'occhio umano ma dovremo mantenere il segnale per 15% del tempo in meno. (vedi fig.) Un altro importante vantaggio del segnale PWM è che è un modo per ottenere un segnale analogico tramite risorse digitali.



3.2 Le MOTOR-shield (AUTORI: Ameglio-Vallarino-Albani)

3.2.1 Ardumoto la motor shield della Sparkfun

(Fonti:https://learn.sparkfun.com/tutorials/ardumoto-shield-hookup-guide)

Questa shield consente un'alimentazione variabile da 3.3V a 5V e una connessione VIN separata. Ardumoto è una scheda in cui il componente principale è l'IC L298 con due H-bridge, usati principalmente per azionare piccoli motori DC. I motori tendono a consumare tanta corrente risulta quindi difficile pilotare un motore direttamente dai pin di uscita di Arduino. Ardumoto consente di controllare un'elevata quantità di corrente (necessaria al pieno funzionamento dei motori) con un segnale molto piccolo con un conseguente minor dispendio di energia per Arduino.Questa shield per Arduino può controllare due motori in DC. Sulla base del L298 H-bridge, l' Ardumoto può pilotare fino a 2 ampere per canale. La scheda è alimentata dalla stessa linea 5V che alimenta la scheda Arduino, include i LED blu e giallo per indicare la direzione attiva, e tutte le linee del driver sono protette da diodi EMF indietro. Per l'alimentazione dei motori si può utilizzare la Vin fornita

dall'Arduino o fornire l'alimentazione di un generatore esterno collegando i fili più e meno negli ingressi sull'Ardumoto per la Vin accanto a quelli per il controllo A1-2 e B-3-4 dei motori presenti sulla Ardumoto shield.

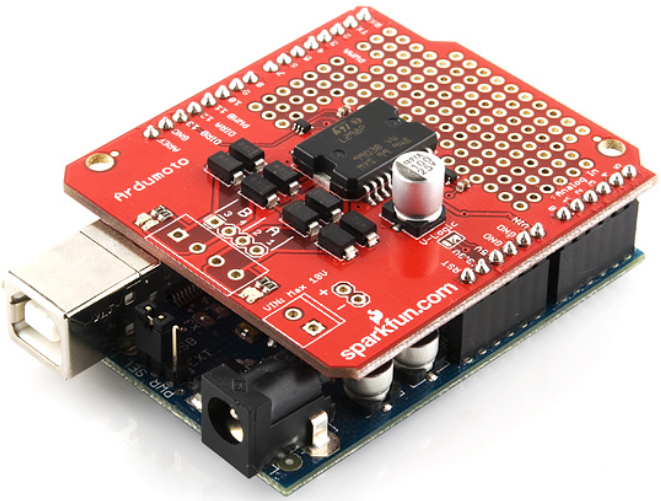


Fig. Ardumoto

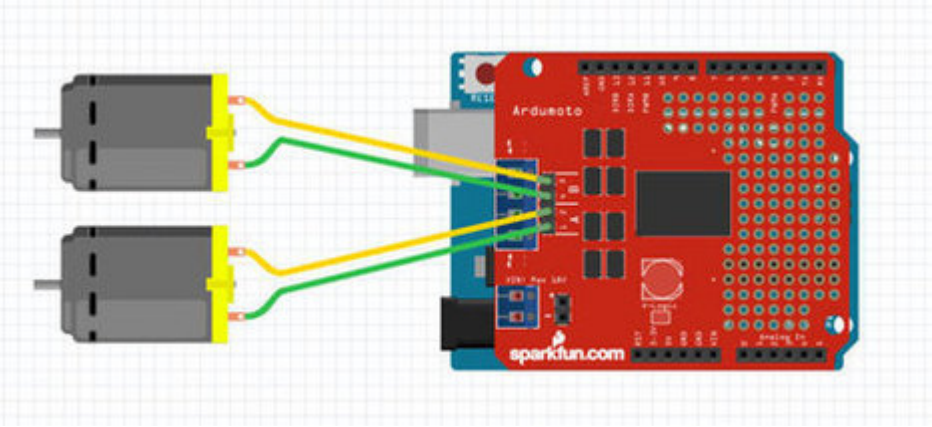


Fig. schema di collegamento motori+ Ardumoto+ Arduino uno

Il controllo per motore collegato a OUT1 / 2 è collegato alla linea digitale 12 (DIR A) e alla linea digitale 3 (PWM A).
Il controllo per il motore collegato a OUT3 / 4 è collegato alla linea digitale 13 (DIR B) e alla linea digitale 11

(PWM B).

3.2.2 Arduino motor shield

(http://www.google.it/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&ved=0CFIQFjAG&url=http%3A%2F%2Fwww.instructables.com%2Fid%2FArduino-Motor-Shield-Tutorial%2F&ei=N3FnU-fxJpD70gXh4IDAAw&usg=AFQjCNH_mAESLWfDho8jqHa50AyX7xY-Ug&sig2=KXsJyqmMywHyF04x76z8VQ&bvm=bv.65788261,d.d2k&cad=rja)

Questa motor shield per Arduino è basato sull'integrato L298: un doppio ponte H progettato per il pilotaggio di carichi induttivi come relè, solenoidi e motori. La shield permette di controllare la velocità e direzione di 2 motori tramite Arduino, in maniera indipendente l'uno dall'altro. E' anche possibile misurare l'assorbimento in corrente di ogni motore.

Caratteristiche

- Tensione operativa: 5 - 12V
- Massima corrente: 2A per canale
- Rilevamento di corrente ing: 1.65V/A

La Arduino Motor Shield consente di controllare facilmente direzione e velocità del motore utilizzando Arduino. Consentendo di utilizzare i pin Arduino, rende molto semplice incorporare un motore nel progetto. Esso permette anche di alimentare un motore con un alimentatore separato fino a 12V.

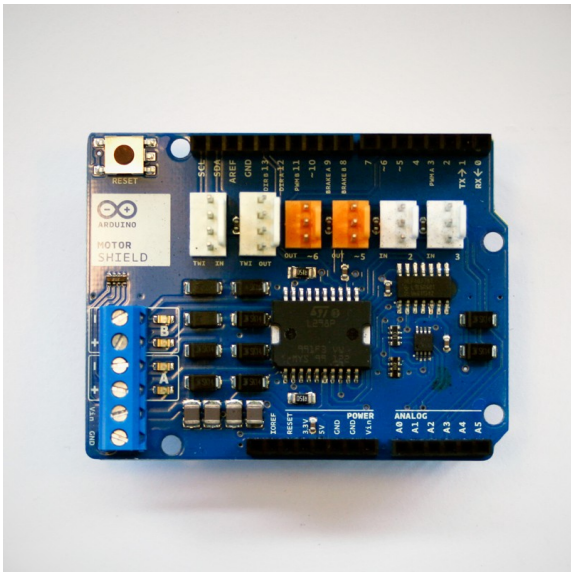


Fig. Arduino motor shield

NOTA I pin della Arduino motor shield sono compatibili solo con Arduino Uno Rev.
Per farlo funzionare con le versioni precedenti di Arduino, sarà necessario tagliare alcuni pin della shield.

Caratteristiche

La motor shield ha due canali, che permettono il controllo di due motori DC, o 1 motore passo-passo. Dispone inoltre di 6 ingressi Tinkerkit per il fissaggio di uscite e linee di comunicazione.

Con un alimentatore esterno, la motor shield può tranquillamente fornire fino a 12V e 2A per canale motore (o 4A a un singolo canale).

Utilizzando gli appositi pin per l'interfacciamento dei motori è possibile selezionare un canale per avviare il motore, specificare la direzione del motore (polarità), impostare la velocità del motore (PWM), arrestare e avviare il motore, e monitorare l'attuale assorbimento di ogni canale.

La ripartizione pin è la seguente:

-Funzione Canale A Canale B
-Direzione Digital 12 Digital 13
-Velocità (PWM) Digital 3 Digital 11
-Freno Digital 9 Digital
-Rilevamento corrente analogica 0 analogico 1

Per controllare un motore utilizzando l' Arduino Motor Shield, prima occorre inserire il positivo (filo rosso) del motore nel terminale + del canale A e la massa (filo nero) del motore nel canale A del terminale – sulla motor shield .
Un alimentatore esterno non è sempre necessario, ma migliora drasticamente le prestazioni del motore. Si consiglia di utilizzarne sempre uno.
Per collegare l'alimentatore esterno, collegare il positivo (filo rosso) dalla rete di alimentazione al terminale "Vin", e la terra (filo nero) al terminale "GND".

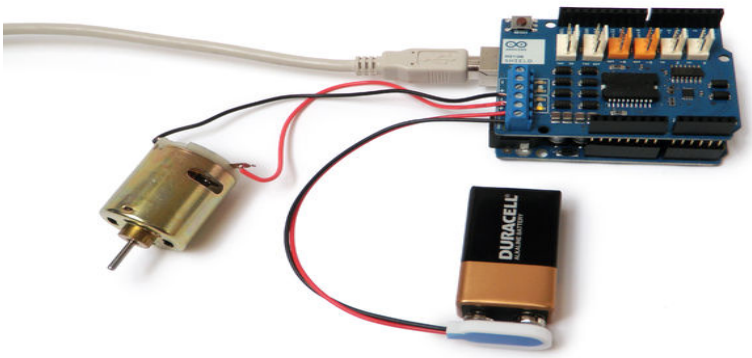


Fig.

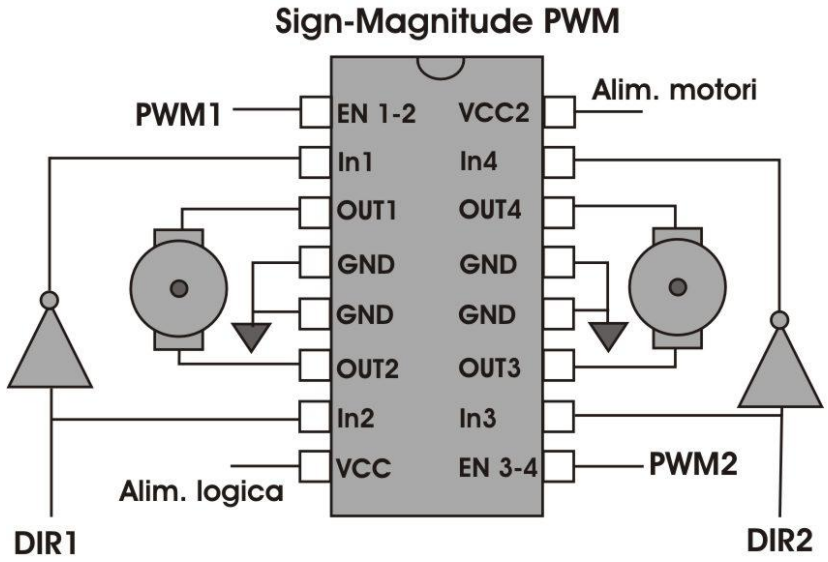
3.3 Pilotaggio dei motori (AUTORI: Ameglio-Vallarino-Albani)

(http://www.tmasi.com/robotica/pwmtut/pwmtut_1.htm)

Le configurazioni per pilotare uno o più motori mediante un ponte H e un segnale PWM, sono due:

- La configurazione **Sign-Magnitude PWM**:

Arduino mediante la linea digitale PWM (A o B)(IN 1-2) invia un segnale con periodo (e quindi anche frequenza) costanti ma con un duty cycle che varia in base alla velocità alla quale vogliamo far girare il motore e attraverso la linea digitale DIR comandiamo la direzione. Quindi inviando il segnale PWM all'ingresso enable del ponte, possiamo comandare la direzione di rotazione del motore tramite i due ingressi di controllo. Tali due ingressi devono essere comandati da segnali invertiti; utilizzando un inverter come nello schema di figura si riduce il numero di pin del microcontrollore necessari per il controllo.



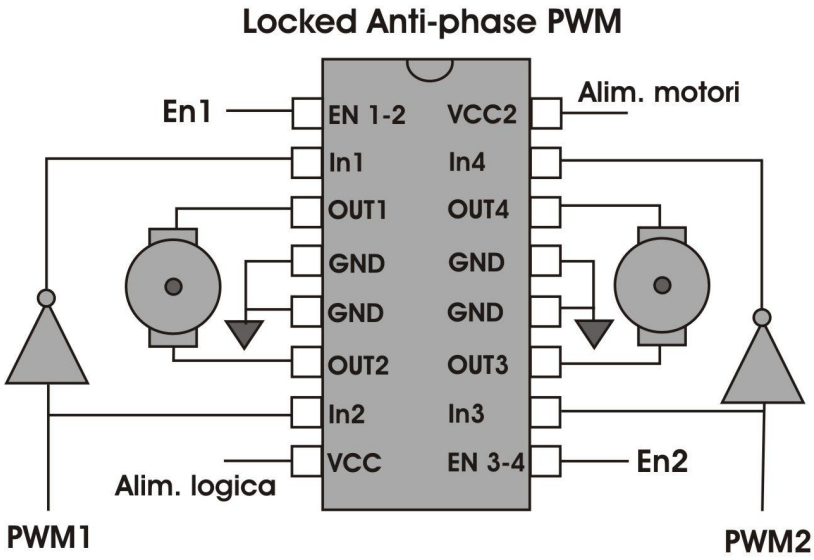
In figura DIRA corrisponde a DIR1 e DIRB corrisponde a DIR2.

- La configurazione **Locked Anti-phase PWM**:

Arduino mediante la linea digitale PWM A o B (IN 1-2) invia un segnale con periodo (e quindi anche frequenza) costanti ma con un duty cycle che varia in base alla velocità alla quale vogliamo far girare il motore. In questo caso però la direzione del motore viene sempre decisa dal segnale digitale che viene messo in ingresso all'invertitore in modo da avere ai due lati opposti del ponte due segnali invertiti tra loro. Il verso di rotazione, viene dedotto dal duty cycle dell'onda nel seguente modo:

1.Duty cycle a 0% : rotazione alla massima velocità in un verso
2.Duty cycle al 50%: motore fermo
3.Duty cycle al 100%: rotazione alla massima velocità nell'altro verso

La linea digitale DIR (Enable) è fissa a 5V quindi agendo sull'enable è possibile spegnere il rispettivo ponte.



In figura PWMA corrisponde a PWM1 e PWMB corrisponde a PWM2.

3.4 Il codice per i movimenti del robot

Questo programma realizzato con l'ambiente di sviluppo Arduino consente al robot di muoversi indietro,avanti, a sinistra e a destra. Per motivi di ordine e di miglior comprensione abbiamo diviso le vari parti del software in funzioni ognuna delle quali ha un compito preciso e differente.

```
#define mot_sinistra 0
#define mot_destra 1

const byte PWMA = 3; // PWM control (speed) for motor A
const byte PWMB = 11; // PWM control (speed) for motor B
const byte DIRA = 12; // Direction control for motor A
const byte DIRB = 13; // Direction control for motor B

void setup()
```

```
{
	setupArdumoto(); // funzione di inizializzazione
}
void loop()
{
	int velocita=125; //imposto la velocità a metà di quella massima fornita dai motori
	avanti(mot_sinistra,velocita); //faccio avanzare il motore di sinistra alla velocità .
	delay(150); //applico un ritardo di 150 ms dato che uno dei motori parte leggermente
prima dell'altro
	avanti(mot_destra,velocita);
	delay(1000); //mantengo il moto in avanti per 1 secondo
	indietro(sinistra,velocita);//faccio indietreggiare il motore di sx
	delay(150);
	indietro(destra,velocita);
	delay(1000);
	destra(velocita); //faccio girare il robot passando solo la velocità
	delay(5000);
	sinistra(velocita);
	delay(5000);
}
void avanti(byte mot, byte spd)
{
	if (mot == 1) // regola quale motore deve essere attivo
	{
		digitalWrite(DIRA, 0); //attivo l'enable
		analogWrite(PWMA, spd);// imposto la velocità
	}
		else if (mot == 0)
			{
				digitalWrite(DIRB, 0);
				analogWrite(PWMB, spd);
			}
}
```

```
void indietro(byte mot, byte spd)
{
	if (mot == 1)
	{
		digitalWrite(DIRA, 1);
		analogWrite(PWMA, spd);
	}
		else if (mot == 0)
			{
				digitalWrite(DIRB, 1);
				analogWrite(PWMB, spd);
			}
}
void destra (byte spd)
{
	spd= spd/2; //dimezzo la velocità in modo da ottenere una precisione maggiore
	digitalWrite(DIRA, 1);
	analogWrite(PWMA, spd);// attivo contemporaneamente entrambi i motori facendoli
andare in senso alternati
	digitalWrite(DIRB, 0);
	analogWrite(PWMB, spd);
	spd= spd*2;
}
void sinistra (byte spd)
{
	spd= spd/2;
	digitalWrite(DIRA, 0);
	analogWrite(PWMA, spd);
	digitalWrite(DIRB, 1);
	analogWrite(PWMB, spd);
	spd= spd*2;
}
void setupArdumoto()
```

```
{
// All pins should be setup as outputs:
        pinMode(PWMA, OUTPUT);
        pinMode(PWMB, OUTPUT);
        pinMode(DIRA, OUTPUT);
        pinMode(DIRB, OUTPUT);
// Initialize all pins as low:
        digitalWrite(PWMA, LOW);
        digitalWrite(PWMB, LOW);
        digitalWrite(DIRA, LOW);
        digitalWrite(DIRB, LOW);
}
```

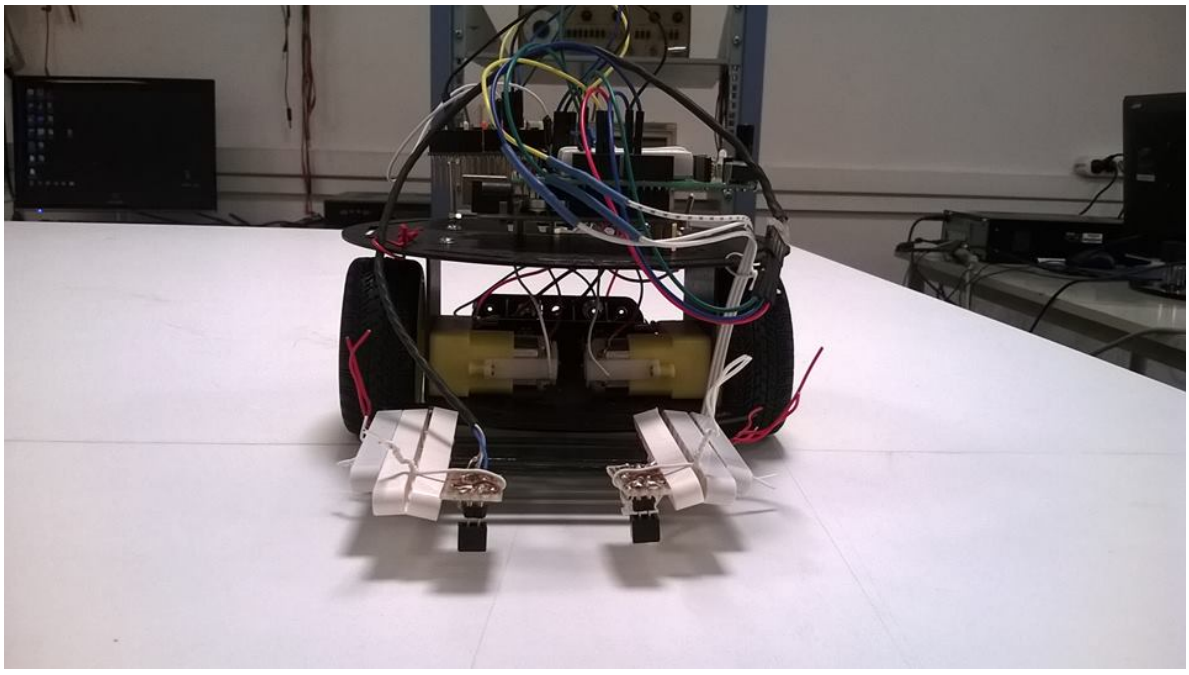


Fig.

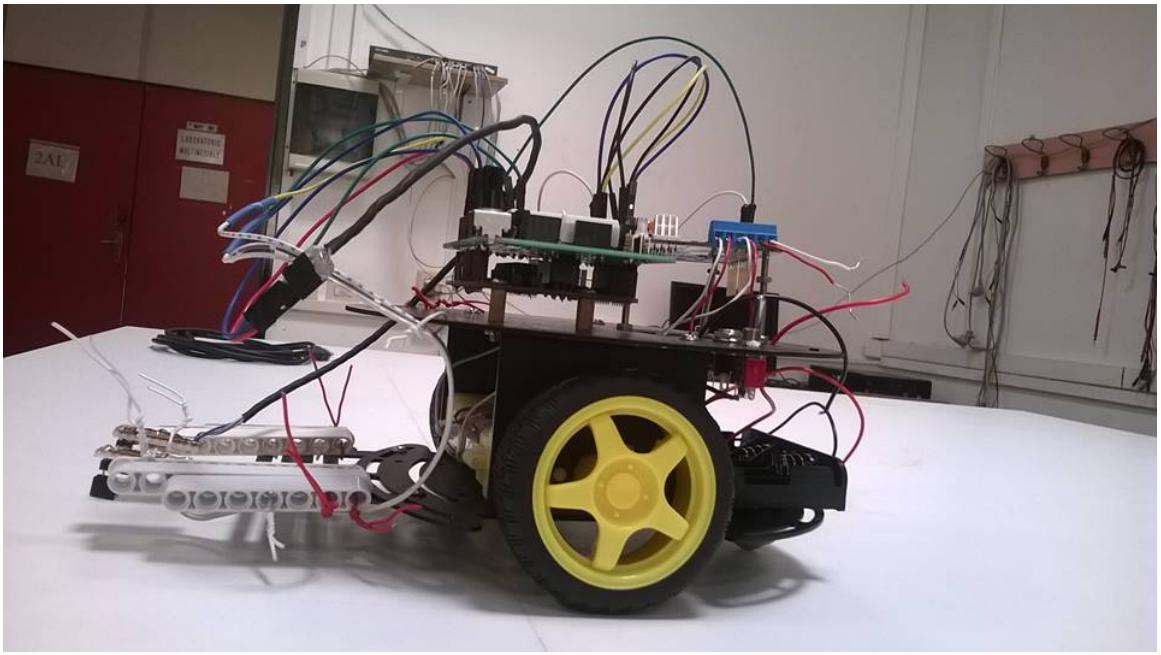


Fig.

4. Introduzione ai sensori utilizzati www.wikipedia.it

4.1 la fotoresistenza (Autori: Novelli-Magro)

La fotoresistenza è un componente elettronico la cui resistenza è inversamente proporzionale alla quantità di luce che lo colpisce. Si comporta come un tradizionale resistore, ma il suo valore in Ohm diminuisce mano a mano che aumenta l'intensità della luce che lo colpisce. Ciò comporta che la corrente elettrica, che transita attraverso tale componente, è proporzionale all'intensità di una sorgente luminosa. In tale maniera si realizza una sorta di potenziometro attuabile tramite la luce anziché tramite forze meccaniche o segnali elettrici.

4.2 sensore di luminosità **Analog line sensor CNY70**

Il CNY70 (fig...) e' un sensore molto diffuso, si interfaccia facilmente con qualsiasi microcontrollore, produce un'uscita analogica quando individua una superficie.

Fig.5

E' composto da un LED ad infrarossi ed un fototransistor. Il fototransistor è un transistor a giunzione bipolare che

viene inscatolato in un contenitore trasparente in modo che la luce possa raggiungere la giunzione del collettore di base: si può paragonare praticamente ad una foto resistenza in cui la cui resistenza elettrica è inversamente proporzionale alla quantità di luce che la colpisce. Si comporta come un tradizionale resistore, ma il suo valore in Ohm diminuisce mano a mano che aumenta l"intensità della luce che la colpisce.
Ciò comporta che la corrente elettrica che transita attraverso tale componente è proporzionale all'intensità di una sorgente luminosa. In tale maniera si realizza una sorta di potenziometro attuabile tramite la luce anziché tramite forze meccaniche o segnali elettrici.
Sfrutteremo questa caratteristica del sensore nel far passare o meno corrente a seconda della superficie che gli si presenta, e con delle prove determineremo la tensione che si misura con una superficie nera e parallelamente con una superficie bianca in modo tale da trovare un range di tensioni dove poter lavorare durante la programmazione.
Tre di questi sensori sono stati montati in linea a capo del robot ad una distanza dal suolo non superiore ai 3 millimetri. Tali dispositivi hanno dei package di dimensione molto ridotta (inferiore al centimetro di lato) e hanno 4 piedini per il loro utilizzo (fig. 6).

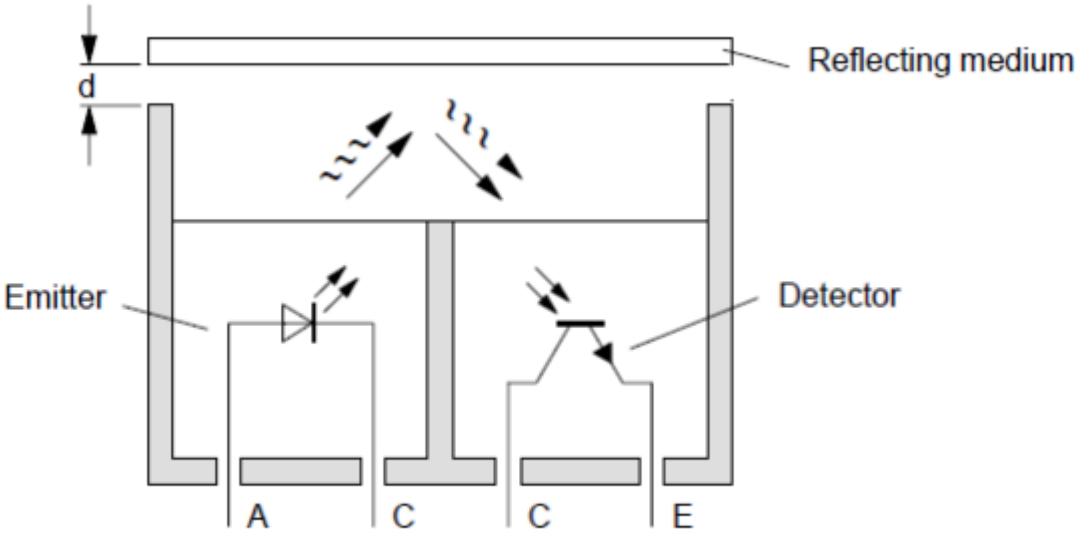
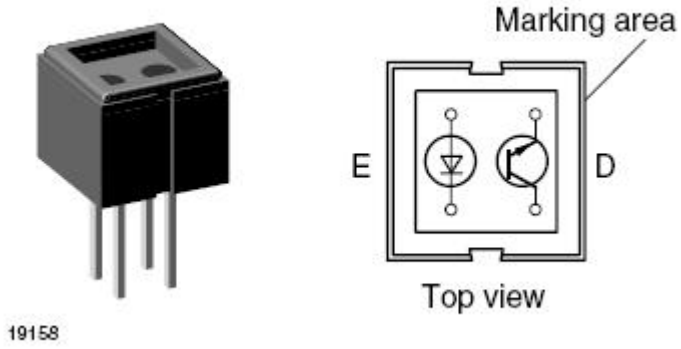


Fig.6

Da sinistra verso destra evidenziamo:
1) A – Anodo
Questo va collegato alla batteria esterna in modo tale da fornire al diodo emettitore una corrente sufficiente (il valore effettivo si può regolare con una opportuna resistenza).
2) C – Catodo
Collegato a massa.
3) C – Collettore
Quando il transistor BJT fotorilevatore viene illuminato con luce infrarossa riflessa da un oggetto ad alto fattore di riflessione, entra in conduzione (interruttore chiuso) e il collettore viene portato alla tensione dell"emettitore. Al

collettore collegheremo pertanto una resistenza di pull-up e all"emettitore collegheremo la massa.12
4) E – Emettitore
Collegato a massa.
I suoi terminali

PIN 1 PIN 2 PIN 3 PIN 4
ANODO CATODO COLLETTORE TRASMETTITORE



Lo schema di montaggio (fig.7) che si è usato per ognuno dei 3 sensori è allora il seguente.

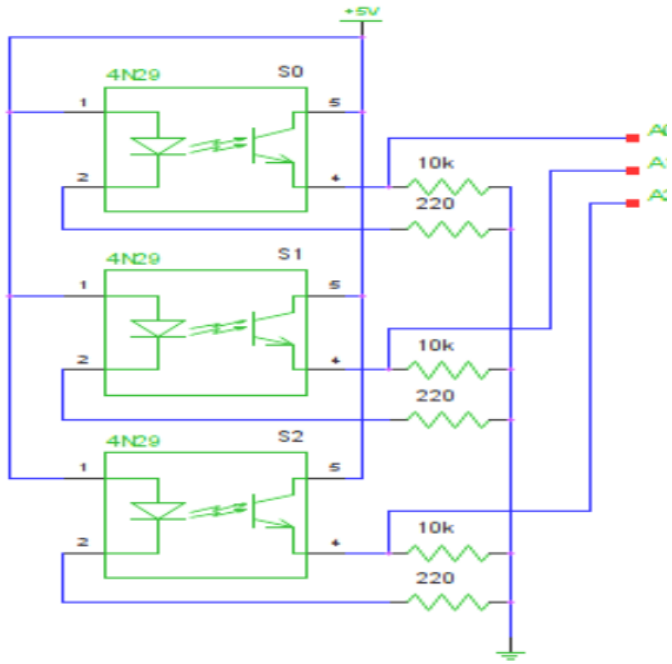


Fig.7

La resistenza da 220 Ω per il pull-up dell' anodo del diodo garantisce una corrente di circa 22mA che forza il diodo ad emettere una luce IR sufficiente ad essere rilevata dal transistor. La resistenza da 10 kΩ per il pull-up sul collettore del fototransistor garantisce invece una corrente molto contenuta all"interno del transistor: questo

comporta anche consumi di potenza molto ridotti per la parte relativa al fototransistor.13
In generale, la tensione misurata sul piedino del microcontrollore sarà un valore intermedio tra 0 e 4.99 V a seconda della corrente che il transistor di rilevazione lascia scorrere (la quale influenza la caduta di potenziale sulla resistenza da 10 kΩ).

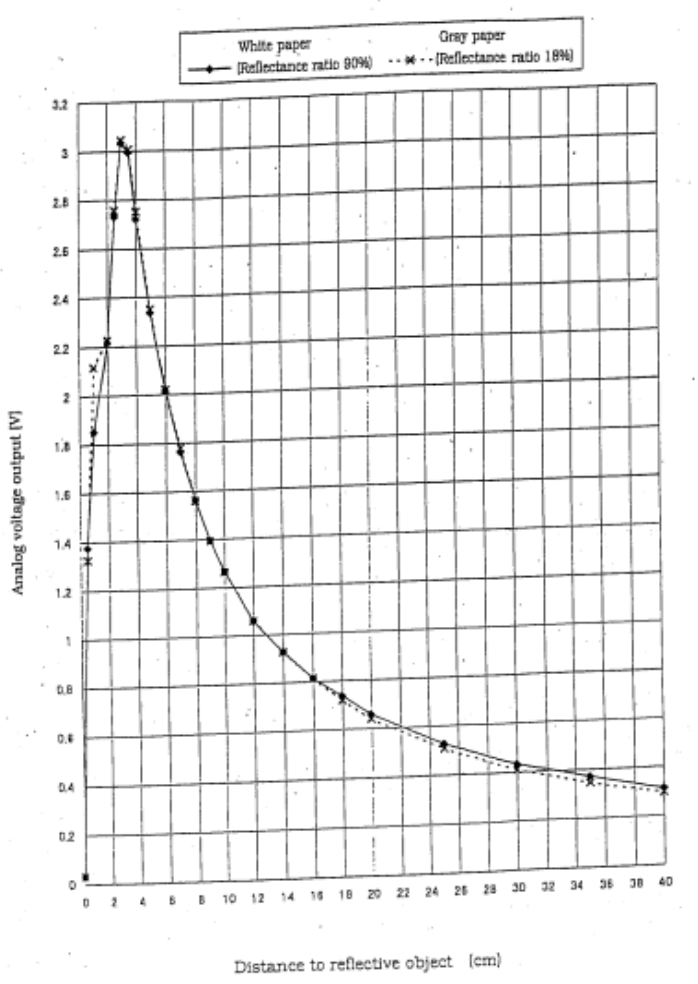
4.2.1 Inseguimento di una linea

Manca circuito HW

Esempio di sketch (incompleto…) per inseguire la linea nera in campo bianco utilizzando solo due sensori di luminosità

```
#define mot_sinistra 0
#define mot_destra 1
#define Bianco 286  //imposto la soglia di riconoscimento del bianco
#define Bianco2 40  //imposto la soglia di riconoscimento del bianco

int value1=0;  //dichiaro la variabile per leggere il valore del 1°sensore
int value2=0;  //dichiaro la variabile per leggere il valore del 2°sensore
int difference=0;  //dichiaro la variabile differenza per i  due valori
int velocita=60; //imposto la velocità come variabile globale a metà della potenza
int SensoreS = 1; //setto il sensore sul piedino 1
int SensoreD =2; //setto il secondo sensore sul piedino 2

const byte PWMA = 3;  // PWM control (speed) for motor A
const byte PWMB = 11; // PWM control (speed) for motor B
const byte DIRA = 12; // Direction control for motor A
const byte DIRB = 13; // Direction control for motor B

void setup()
	{
	Serial.begin(9600);
	setupArdumoto();  // utilizzo la funzione inizializzatrice per il movimento dei motori
	}
```

```
void loop()
 {

      if(analogRead(SensoreS)<=Bianco && analogRead(SensoreD)<=Bianco)
            {
                  avanti(velocita);
                 delay(1000);
           }
       else if(analogRead(SensoreS)<Bianco && analogRead(SensoreD)>Bianco)
            {
                 sinistra(velocita);
                 delay(500);
            }

      else if(analogRead(SensoreS)>Bianco && analogRead(SensoreD)<Bianco)
            {
                destra(velocita);
                delay(500);
          }
      if(analogRead(SensoreS)>Bianco && analogRead(SensoreD)>Bianco)
         {
                  indietro(velocita);
                  delay(1000);
        }
 }
```



4.3 sensore di distanza ad infrarossi SHARP 2d120x f 2y

Il sensore SHARP 2D120X ha un costo contenuto, sotto i 20 euro, ed è utilissimo per realizzare semplici applicazioni .

Questo sensore è molto popolare nelle applicazioni di robotica amatoriale, è costruito dalla Sharp e produce un' uscita analogica che varia da 3.1V a 4 cm fino a 0.3V a 30cm.

Non è un sensore di precisione ed il suo funzionamento non è lineare come mostra il grafico di risposta estratto dal datasheet ufficiale.
La formula per tradurre il valore del sensore (SensorValue) nella corrispondente distanza (Distance) è la seguente (la formula è valida solo per un SensorValue compreso tra 80 e 530):

Distance (cm) = 2076\(sensorvalue - 11)

Nella tabella seguente sono riportate le caratteristiche principali del sensore in esame:

Characteristic	Value
Operating Supply Voltage	4.5V to 5.5V
Minimum Measuring Distance	4cm
Maximum Measuring Distance	30cm
Average Supply Current - Typical	33mA
Response Time	38 ± 10ms
Weight	3.5g

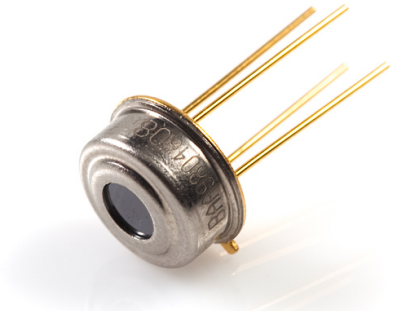
http://www.robot-italy.com/it/3520-sharp-distance-sensor-2d120x-4-30cm.html#sthash.5x1rSGt0.dpuf

4.4 sensore di temperatura MLX90614 (Autore: Solazzo)

http://bildr.org/2011/02/mlx90614-arduino/

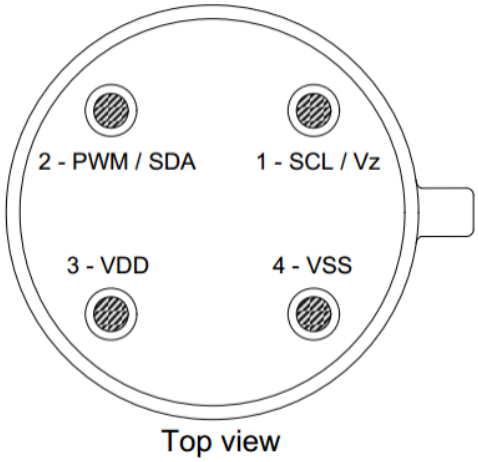
Caratteristiche:

- Dimensioni e costo ridotti.
- Facile da integrare.
- Calibrato tra ampi range di temperatura:



-40 to +85°C per misurazioni ambientali.
-70 to +380°C per misurazione di oggetti.

- Interfaccia digitale SMBus.
- Uscita PWM adattabile per letture continue.
- Sensibilità di 0,02°C.
- Adattamento semplice da 8 a 16 V
- Disponibile nella versione da 3V e da 5V
- Possibilità di utilizzo del power saving mode



Top view

Descrizione generale

Il termometro MLX90614 è un termometro ad infrarossi sviluppato per la misura di temperature a distanza. Esso contiene al suo interno un ADC (Analogic-Digital Converter) a 17 bit ed un DSP (Digital Signal Processing) estremamente performante costituito da un processore dedicato ed ottimizzato all' esecuzione di sequenze di istruzioni ricorrenti. Tale componente ha una ampia gamma di applicazioni tra cui la misura della temperatura corporea e la rilevazione di movimenti.

L'MLX90614 fornisce due metodi differenti di lavoro, il PWM (10 bit) e l' SMBus (i.e, TWI, I^2C), il sistema PWM consente di ottenere una sensibilità di 0,14°C, mentre l'interfaccia TWI riesce a ridurre tale sensibilità fino a 0,02°C.

L' MLX90614 viene calibrato in fabbrica con ampi range di temperatura, ovvero da -40°C a 85°C per le misurazioni ambientali e

E' importante ricordare che i valori di uscita del componente risultano come la media dei valori rilevati nell'area di visibilità del sensore.

Questo sensore viene prodotto secondo lo standard TO-39 package, e ne è stata creata anche una versione funzionante a 3V

Testing del sensore

Il sensore necessita di un semplice circuito di controllo per assicurarne il perfetto funzionamento come illustrato in figura.

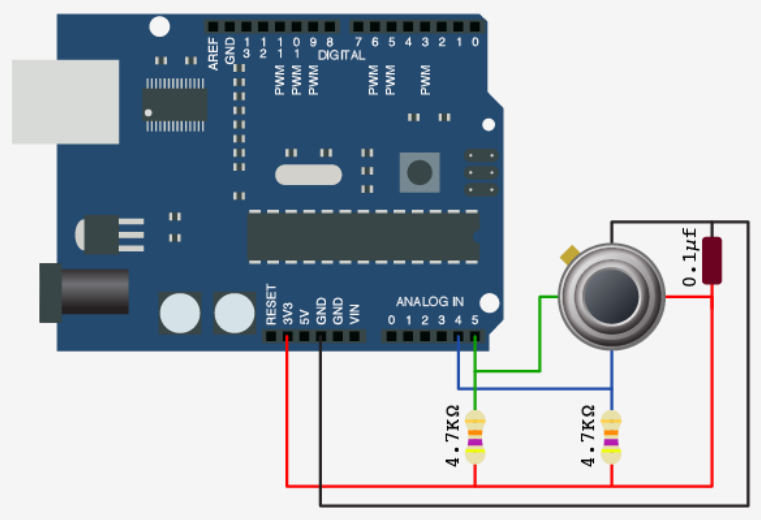


Fig.

Pin Name	Funzione
VSS	Ground.
SCL / Vz	Ingresso seriale del clock per il protocollo di comunicazione a 2 fili. Su questo pin è disponibile uno zener da 5.7V per il la connessione di un transistor bipolare esterno al MLX90614 per supportare il dispositivo con un alimentazione esterna tra gli 8 ed i 16V.
PWM / SDA	Input/Output digitale. Pin per le misure di temperatura in modalità PWM. In SMBus compatible mode automatically configured as open drain NMOS.
VDD	Alimentazione esterna

Codice per il testing del sensore : _____

4.5 Il sensore ad Ultrasuoni SFR05 (Autori: Novelli-Magro)

In generale i sensori di prossimità ad ultrasuoni funzionano sul principio del sonar: emettono impulsi sonori ultrasonici, e rilevano un' eventuale eco di ritorno generato dalla presenza di un oggetto all'interno della portata nominale.
Questi sensori sono complessi e dispongono spesso di funzioni evolute:

1. settaggio della distanza di commutazione;
2. uscita analogica per la trasduzione della distanza dell'oggetto rilevato;
3. settaggio del campo sensibile;
4. programmazione software dei settaggi dello strumento.

La velocità di commutazione di questi sensori di prossimità è bassa, ma in compenso presentano dei significativi vantaggi:

1. possono avere portate nominali molto elevate (fino a 10 m);
2. sono immuni ai disturbi elettromagnetici;
3. possono rilevare oggetti di qualsiasi materiale (eccetto materiali fonoassorbenti);
4. possono rilevare oggetti senza che questi siano stati preventivamente preparati.

Una certa attenzione va però posta nella dimensione e nell'orientamento della superficie dell'oggetto che si rivolge al sensore, infatti una superficie troppo piccola o orientata malamente (non ortogonale alla direzione di lettura del sensore) può non assicurare la generazione di un'eco rilevabile.

Il sensore SFR05 è COSTITUITO da un emettitore di ultrasuoni ed un ricevitore orientati nella stessa direzione. Per eseguire una misura è necessario inviare un segnale (Trigger) e leggere l'eco di quel segnale che dopo aver rimbalzato sull'oggetto torna indietro (Echo).
L'ultrasuono emesso dal trasmettitore, non udibile dall'orecchio umano in quanto oltre la soglia che puoi

percepire, rimbalza sull'oggetto e torna nella stessa direzione in modo che l'Echo, questo è il nome del segnale di ritorno, possa essere captato dal ricevitore.
Questo sensore è inoltre dotato di un microcontrollore che assolve tutte le funzioni di calcolo ed elaborazione.
Per convertire l'intervallo di tempo misurato in una lunghezza, bisogna ricordare che la velocità del suono NELL'ARIA è di 343,4 m/s a 20 °C ed in generale varia secondo la relazione v = 331,4 + 0,62 T dove la temperatura T è misurata in °C.
Per la realizzazione del nostro metro elettronico presumiamo di lavorare ad una temperatura ambiente di 20 °C e quindi la velocità del suono sarà di 343 m/s (approssimiamo) che vuol dire anche 0,0343 cm/microsecondi.Quindi, ricordando che v = s/t, lo spazio percorso sarà:
s = v*t da cui s = 0,0343 *t
però, per calcolare lo spazio percorso, bisogna tener conto che il suono percorre due volte la distanza da misurare (giunge sull'oggetto e ritorna indietro al sensore) quindi il valore di t ottenuto deve essere diviso per 2.
La formula corretta per la misura dello spazio percorso è:
s = 0,0343 * t/2
eseguendo la divisione di 0,0343/2 possiamo scrivere:
s = 0,01715 * t

oppure:
s = t/58,31
approssimando
s = t/58
formula più semplice da ricordare.
Per calcolare la distanza dell'oggetto dal sensore sarà sufficiente dividere il tempo t (tempo impiegato dal segnale per giungere sull'oggetto e tornare al sensore) per 58.
Da notare che il campo visivo di questo sensore e' abbastanza ampio, sebbene questo si possa ottimizzare regolando la soglia;

le specifiche tecniche:

Frequenza operativa: 40 kHz
Portata: da 3 cm a 400 cm
Angolo di rilevamento ottimale: 30°
Tensione operativa: 5 V DC
Corrente a riposo: <2 mA
Segnale in uscita: PWM 0-5V
Sensibilità: fino a 3 mm
Dimensioni: 45x21x18 mm

4.5.1 Il controllo della distanza

L’esempio di sketch sotto riportato permette al robot di acquisire la distanza in cm degli ostacoli da 3 cm a 4 metri circa.
In sostanza viene generato sul pin trigger un segnale HIGH per la durata di 10 millisecondi, necessario al sensore per generare l’ultrasuono da trasmettere e di conseguenza attendere sul pin echo il ritorno del segnale. Quando il pin echo riceve il segnale il microprocessore all'interno del sensore effettua il calcolo spiegato precedentemente e fornisce in uscita sul monitor seriale il valore in decimal

```
// sketch di esempio per usare il sensore ad ultrasuoni (SRF05)
#define SONAR_TRIGGER_PIN 6
# define SONAR_ECHO_PIN 7
#define LED 13
unsigned int measure_distance()
{
        digitalWrite(SONAR_TRIGGER_PIN,HIGH); // invia ultrasuoni
        delayMicroseconds(10); // compi l'azione precedente per 10 microsecondi
        digitalWrite(SONAR_TRIGGER_PIN,LOW); smetti di inviare ultrasuoni
        unsigned long pippo = pulseIn(SONAR_ECHO_PIN, HIGH); assegna a pippo il valore rilevato
        delay(50);
        return ((unsigned int)(pippo/58));
}
void setup()
{
        Serial.begin(9600); //utilizza il monitor seriale come uscita dei valori
        pinMode(SONAR_TRIGGER_PIN,OUTPUT);
        pinMode(SONAR_ECHO_PIN,INPUT);
        pinMode(LED,OUTPUT);
}

void loop()
{
        unsigned int ostacolo= measure_distance();
        Serial.println(ostacolo,DEC); //Stampa a schermo l'ostacolo in valore decimale
        if(ostacolo<13)
                {
                        digitalWrite(LED,HIGH); // led acceso
                }
        else
                {
                      digitalWrite(LED,LOW); // led spento
```

```
        }
}
```

Osservazioni: si è notato che il sensore ad ultrasuoni è uno strumento molto preciso ed utile. Può essere usato per molte funzioni anche perchè non è difficile da programmare e non occupa molto spazio visto che ha delle dimensioni ridotte.

5. COMUNICAZIONE SENSORI - ARDUINO

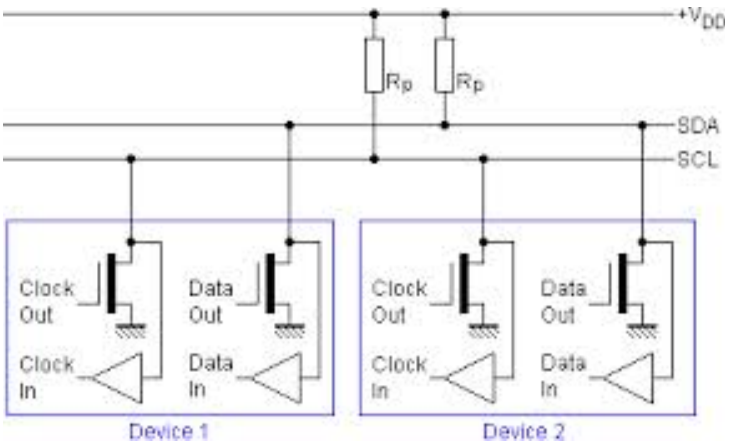
5.1 Il protocollo I^2C (Autrice: D. Staltari)

www.cis.ait.asia/course_offerings/92/handouts/386

La comunicazione I2C (Inter Integrated Circuit) è un sistema di comunicazione **seriale** bifilare utilizzato tra circuiti integrati.

L' I2C è costituito da due sole linee denominate rispettivamente **SDA** (Serial Data line) per i dati e il secondo chiamato **SCL** (Serial Clock Line), utilizzato come segnale di clock.

Il classico bus I2C è composto da almeno un dispositivo master ed uno slave (letteralmente "capo" e "sottoposto, schiavo").



Se si utilizza un solo dispositivo I2C non è necessario(normalmente) mettere delle resistenze di pull-up. Tuttavia, se si utilizzano diversi dispositivi, bisogna utilizzare due resistenze da 10 kOhm (,ma possono essere omesse)

L'I2C ha 7 bit di indirizzo (B1 è il bit più significativo, B7 il meno significativo) e quindi 128 possibili indirizzi diversi (detti nodi). Di questi però 16 sono riservati quindi i dispositivi che possono essere collegati sullo stesso bus sono al massimo 112. Le velocità di trasmissione nel modo standard sono di 100 kbit/s e 10 kbit/s (velocità del low-speed mode).

Nel progetto, la scheda Arduino è il Master e dispositivi sul bus I2C sono gli schiavi. Siamo in grado di scrivere/ leggere i dati da un dispositivo. Ognuno di esso ha un indirizzo univoco (indirizzo 7-bit). Usiamo

questo indirizzo nella funzione scrittura/lettura richiesta dal dispositivo.

-Protocollo di comunicazione I2C

I2C è un protocollo master / slave .

Il master avvia la comunicazione, a susseguirsi troviamo altri 5 step:

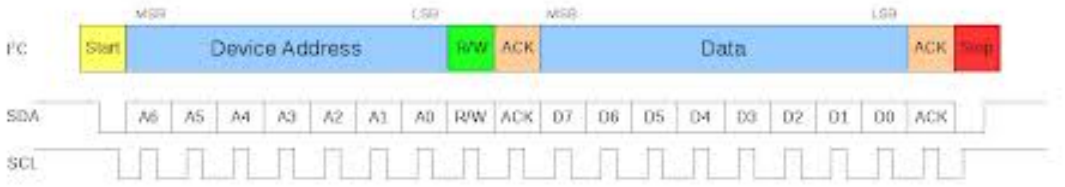
1 . Il dispositivo master emette una condizione di avvio . Questa informa tutti i dispositivi slave di ascoltare sulla linea dati seriale per le istruzioni .

2 . Dispositivo master invia l'indirizzo a 7 bit del dispositivo di destinazione slave ed un flag di lettura / scrittura

3.Dispositivo slave con l'indirizzo corrispondente risponde con un segnale di ACK .

4 . Il master e slave possono ricevere o trasmettere dati a seconda se la comunicazione è una lettura o scrittura. Dopodiche il trasmettitore invia 8 - bit di dati al ricevitore che risponde con un ACK (1 bit) .

5 . Quando la comunicazione è stata completata , il master invia una condizione di arresto.



come si interfacciano più sensori seriali ad Arduino UNO Rev.

Come detto precedentemente possiamo interfacciare più dispositivi tra loro, utilizzando I2C. Prima di tutto dovremmo dare ad ogni dispositivo un indirizzo diverso, così che non si crei conflitto l'uno con l'altro.

Per interfacciare i vari dispositivi dovremmo collegare via HW tutti i SDA, SCL massa e vcc insieme. Una volta fatto il collegamento bisogna passare alla parte SW.

Per prima cosa adopereremo la libreria I2c.master, dove andremo ad utilizzare delle funzioni al suo interno.

5.2 lettura della temperatura con il sensore MLX90614

Andremo a vedere come adoperare l'i2c con il sensore MLX90614:

Per prima cosa leggiamo l'indirizzo del sensore con il seguente esempio di sketch:

```
#include <i2cmaster.h>

int broadcastAddress= 0x00<<1;  //Indirizzo di broadcast

void setup()
```

```
{
     Serial.begin(9600);          //Inizia la comunicazione seriale col
                                                    PC a [9600 baud]

  Serial.println("Lettura dell'indirizzo del device collegato \n");

  i2c_init();                              //Inizializzazione I2C
  PORTC = (1 << PORTC4) | (1 << PORTC5);    //Abilitazione pullups
(pin A4 e A5)

  i2c_start_wait(broadcastAddress+I2C_WRITE); //Invia il bit di start
e il comando di scrittura all'indirizzo di broadcast
  i2c_write(0x2E);                  //COMANDO: accedi alla
lettura/scrittura dell'indirizzo

  i2c_rep_start(broadcastAddress+I2C_READ);   //Invio del bit di
repeated start e comando di lettura
  int addLow = i2c_readAck();       //Legge il byte Low e invia l'ACK
  int addHigh = i2c_readAck();      //Legge il byte High e invia l'ACK
  int pec = i2c_readNak();
  i2c_stop();                       //Chiude la comunicazione

  Serial.print("Indirizzo: 0x");    //Scrive sul monitor seriale
l'indirizzo letto
  Serial.println((((addHigh & 0x00FF) << 8) + addLow),HEX);
}

void loop()
{
}
```

Una volta letto l'indirizzo andremo a cancellarlo con il seguente codice:

```
/*
```

```
  Cancellazione dell'indirizzo del componente collegato (viene scritto
0x00)


*/
#include <i2cmaster.h>


void setup()
{
    Serial.begin(9600);
    Serial.println("Cancellazione dell'indirizzo \n");

    i2c_init(); //Inizializza
    PORTC = (1 << PORTC4) | (1 << PORTC5); //Abilita pullups (pin A4 e
A5)

    int dev = 0x00<<1;
    int data_low = 0;
    int data_high = 0;
    int pec = 0;

    i2c_start_wait(dev+I2C_WRITE);
    i2c_write(0x2E);
    i2c_write(0x00);
    i2c_write(0x00);
    i2c_write(0x6F);  //CRC relativo all'indirizzo 0x00 (non bisogna
mai cambiarlo, serve per capire se l'indirizzo è giusto, inoltre
dipende dal resto dell'indirizzo)
    i2c_stop();

    Serial.print("Indirizzo resettato!");
}
```

```
void loop()
{
}
```

A questo punto potremmo mettere l'inidirizzo a nostro piacimento.

```
/*
  Cambio dell'indirizzo di un sensore MLX90614

  */
#include <i2cmaster.h>
#define ADDRESS 0x60  //Indirizzo da assegnare, al posto di 0X60 si può mettere 0X1B ecc in esadecimale

int broadcastAddress= 0x00<<1;  //Indirizzo di broadcast
int newAddress= ADDRESS<<1;    //Indirizzo da assegnare shiftato a sinistra (lo spazio serve al bit di read/write)

void setup()
{
  Serial.begin(9600);

  Serial.println("Cambio dell'indirizzo del device collegato \n");

  i2c_init();                                 //Inizializzazione I2C
  PORTC = (1 << PORTC4) | (1 << PORTC5);    //Abilitazione pullups (pin A4 e A5)

  for (int a = 0; a != 256; a++)
  {
    i2c_start_wait(broadcastAddress+I2C_WRITE); //Invia il bit di start e il comando di scrittura all'indirizzo di broadcast
    i2c_write(0x2E);          //COMANDO: accedi alla
```

```
lettura/scrittura dell'indirizzo(accesso alla ram di indirizzo)
    i2c_write(ADDRESS);        //Scrive il byte Low dell'indirizzo e
invia l'ACK
    i2c_write(0x00);           //Scrive il byte High dell'indirizzo e
invia l'ACK
    if (i2c_write(a) == 0)     //Prova ad esclusione del CRC
    {
    i2c_stop();                // Chiude la comunicazione
    delay(100);                // Wait 10ms.
    Serial.print("CRC: 0x");
    Serial.println(a, HEX);
    }
  }

  Serial.print("Indirizzo cambiato (non verificato) in 0x");
  Serial.println(ADDRESS,HEX);
}

void loop()
{
}
```

Infine andiamo a misurare la temperatura di un solo sensore:

```
/*
  Misura della temperatura

  DESCRIVI FUNZIONAMENTO
*/

#include <i2cmaster.h>
#define ADDRESS   0x60

void setup(){
```

```
    Serial.begin(9600);
    Serial.println("Setup...");

    i2c_init(); //Initialise the i2c bus
    PORTC = (1 << PORTC4) | (1 << PORTC5);//enable pullups
}

void loop(){
    int dev = ADDRESS<<1;      //indirizzo del dispositivo (si può
mettere anche 0x00) traslato a sinistra di 1 per
                           //lasciare spazio al bit read/write
    int data_low = 0;
    int data_high = 0;
    int pec = 0;

    i2c_start_wait(dev+I2C_WRITE);     //bit di start + slave address
+ bit di read
    i2c_write(0x07);                   //indirizzo della RAM del
dispositivo in cui memorizzata la temperatura rilevata

    // read
    i2c_rep_start(dev+I2C_READ);       //invio del bit di repeated
start
    data_low = i2c_readAck(); //Read 1 byte and then send ack
    data_high = i2c_readAck(); //Read 1 byte and then send ack
    pec = i2c_readNak();
    i2c_stop();

    //This converts high and low bytes together and processes
temperature, MSB is a error bit and is ignored for temps
    double tempFactor = 0.02; // 0.02 degrees per LSB (measurement
resolution of the MLX90614)
    double tempData = 0x0000; // zero out the data
```

```
    int frac; // data past the decimal point

// This masks off the error bit of the high byte, then moves it left 8
bits and adds the low byte.
    tempData = (double)(((data_high & 0x007F) << 8) + data_low);
    tempData = (tempData * tempFactor)-0.01;

    float celcius = tempData - 273.15;
    float fahrenheit = (celcius*1.8) + 32;

    Serial.print("Celcius: ");
    Serial.println(celcius);


    delay(1000); // wait a second before printing again
}
```

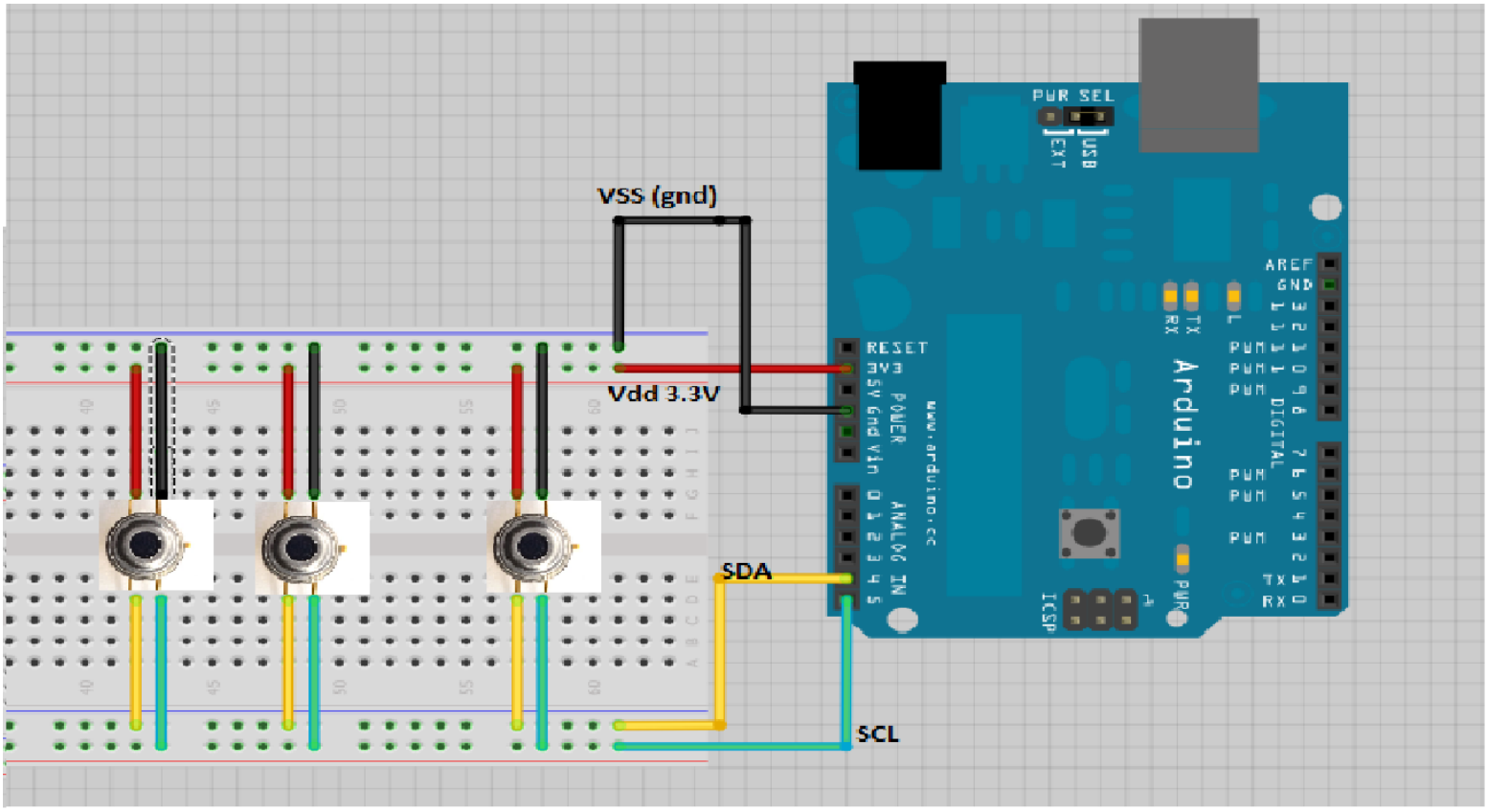
5.3 lettura della temperatura da tre sensori MLX90614 in parallelo sull'I2C bus

Sketch per il rilevamento della temperatura da tre sensori MLX90614 e se la temperatura rilevata da almeno uno dei tre è superiore a valore di soglia viene attivato un led.

```
#define calore 30 //soglia di temperatura

void Misura_Calore()
{
    int dev = ADDRESS<<1;     //indirizzo del dispositivo (si può
mettere anche 0x00) traslato a sinsitra di 1 per lasciare spazio al
bit read/write

    int data_low = 0;
    int data_high = 0;
    int pec = 0;
```

```
     i2c_start_wait(dev+I2C_WRITE);     //bit di start + slave address
                                                    + bit di read
    i2c_write(0x07);                   //indirizzo della RAM del
dispositivo in cui verrà memorizzata la temperatura rilevata (0x07
SEMPRE, valore fisso che indica l'indirizzo della locazione di
memoria)

    // read
    i2c_rep_start(dev+I2C_READ);    //invio del bit di repeated start
    data_low = i2c_readAck();       //leggi 1 byte e poi invia l'ack
    data_high = i2c_readAck();     //leggi 1 byte e poi invia l'ack
    pec = i2c_readNak();
    i2c_stop();

     //Questo trasforma i byte alti e bassi insieme ed trasforma la
temperatura, MSB is a error bit and is ignored for temps.
    double tempFactor = 0.02;      // 0.02 gradi LSB( risoluzione
della misura del MLX90614)
    double tempData = 0x0000; // zero out the data
    int frac;                 // data past the decimal point

    // This masks off the error bit of the high byte, then moves it
left 8 bits and adds the low byte.
    tempData = (double)(((data_high & 0x007F) << 8) + data_low);
    tempData = (tempData * tempFactor)-0.01;

    float celcius = tempData - 273.15;
    float fahrenheit = (celcius*1.8) + 32;

    Serial.print("Celcius: ");
    Serial.println(celcius);
}
```

```
void (int device)
{
 int dev = 0;
 switch(device)
 {
   case(1):
   {
    dev = 0x5A << 1;  //indirizzo sensore dx(l'indirizzo cambia a
seconda di quello che uno scrive utilizando il programma cambio
indirizzo)
    break;
   }
   case(2):
   {
    dev = 0x5B << 1;  //indirizzo sensore davanti
    break;
   }
   case(3):
   {
    dev = 0x5C << 1;  //indirizzo sensore sinistro
    break;
   }
 }

 if(dev > calore) //se la temperatura letta da uno dei tre sensori
supera la nostra soglia (30°) faremo accendere un led.
 {
  //accendi LED
 }
```



manca collegamento con led