Laboratorio di fisica
con Arduino

Massimiliano Virdis

Indice

INDICE

1 Introduzione

1.1 Premessa

Caro lettore,

credo che chiunque insegni fisica alle superiori abbia sentito ormai parlare di Arduino. Alcuni di noi hanno cominciato ad usarlo, sentendo di meravigliose e facili applicazioni anche per lo studio della fisica. Tuttavia, quando poi si passa all'azione, è facile entrare nello sconforto, per il fatto che tra fare accendere un led lampeggiante e preparare un'esperienza completa c'è molto più lavoro di quanto si immaginasse.

Anch'io, animato dal desiderio di usare Arduino nelle attività di laboratorio, ho cominciato a raccogliere del materiale. La prima parte di quest'opera raccoglie una panoramica dei sensori che ho potuto utilizzare, degli schemi di montaggio e dei codici per il loro funzionamento: è una sintesi, spero ordinata, di quel che si può trovare sparso in rete. Nella seconda parte descrivo invece qualche esperienza completa con i dati sperimentali che in esse ho raccolto. Osservo che quasi tutte le esperienze che ho riportate attualmente sono state svolte in casa con il materiale che mi era disponibile. Le esperienze sono state svolte tutte almeno con Arduino Uno R3; sto iniziando a verificare che si possano eseguire anche con la nuova R4.

Questo *documento è stato scritto innanzi tutto per insegnanti delle superiori* e poi per gli alunni più appassionati. *Non è un manuale su Arduino, ma di fisica con Arduino.*

Spero che quanto riportato in quest'opera sia se non di aiuto almeno non dannoso. Per migliorare quanto scritto e evidenziare qualsiasi errore non esitate a scrivermi.

email: prof.virdis@tiscali.it

1.2 Licenza e Copyright

δωρεὰν ἐλάβετε, δωρεὰν δότε (Mt. 7.8)

1.3 Ringraziamenti

Si ringraziano coloro che hanno avuto la pazienza di leggere queste pagine e di segnalare errori di vario tipo. In particolare Aldo Fasano.

Parte I

Esperienze e schemi fondamentali

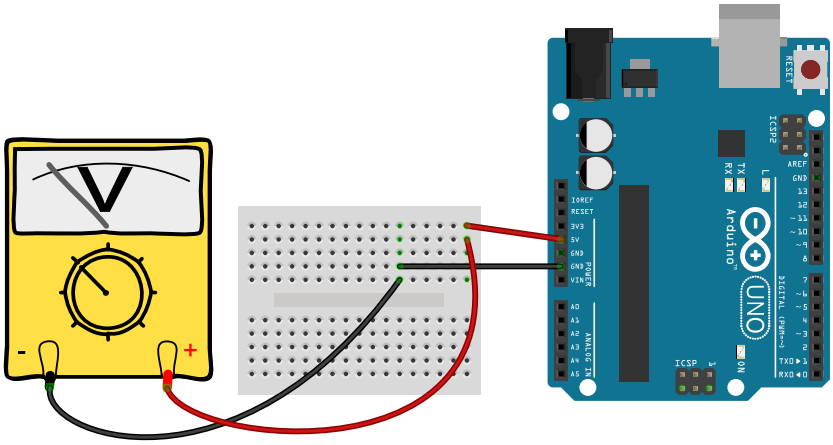
2 Prove e schemi fondamentali

2.1 Introduzione

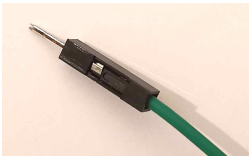
Quanto riportato in queste pagine non è un corso di programmazione su Arduino, ma una serie di prove fondamentali per capirne il funzionamento anche dal punto di vista elettrico. Sono inoltre presentati alcuni schemi fondamentali che poi ritroviamo utilizzati nelle varie esperienze successive.

2.2 Misuriamo la ddp tra GND e 5V con un tester

La d.d.p. nominale tra il pin GND e il pin dei 5 V è appunto 5 V, ma quella effettiva è solitamente leggermente minore. Misuriamo questa tensione con un tester (impostiamo la portata a 20 V o 10 V). Per poter collegare i terminali del tester con i pin colleghiamo i pin della scheda con una breadboard e i pin di questa con i terminali del tester con un filo elettrico spellato avvolto alle punte dei terminali. *Evitiamo sempre di inserire direttamente dei cavi nei pin della scheda, perché potrebbero spezzarsi e rimanere definitivamente incastrati nel pin, rendendolo inutilizzabile insieme a tutta la scheda.* Normalmente la tensione effettivamente disponibile è intorno ai 4,9 V e quasi sempre inferiore a quella nominale.

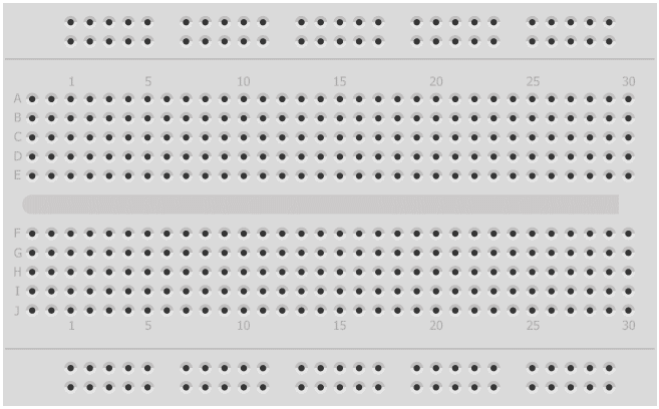


Per il collegamento della scheda alla breadboard usiamo cavi con terminali Dupont come quelli mostrati nella foto.



2.3 Breadboard

La breadboard è una struttura in plastica con centinaia di fori interconnessi elettricamente. Analizziamo l'esempio qui proposto.



Abbiamo innanzi due righe di fori nella parte superiore ed inferiore. In molte breadboard di questo tipo su queste due coppie di righe è serigrafato un più e un meno: è un suggerimento per usare queste righe come linea comune per il polo positivo e negativo dell'alimentazione. I fori all'interno di ogni riga sono interconnessi; le righe sono isolate tra loro. La resistenza di isolamento tra i pin è elevata, ma comunque dell'ordine dei 10 megaohm e non infinita.

Nella parte centrale abbiamo dieci righe indicate con lettere e 32 colonne indicate con numeri. In questo caso sono in comune i fori delle colonne in gruppi di cinque: cinque nelle righe più in alto e cinque in quelle più in basso.

Nell'esempio della pagina precedente i terminali Dupont sono nella stessa colonna dei terminali del tester e quindi in collegamento tra di loro.

2.4 Accendiamo un led

I led sono diodi capaci di emettere luce. In essi la corrente può fondamentale fluire in un solo verso, dal polo positivo a quello negativo. Ponendo il led al contrario non lo danneggiamo, ma non passerà corrente. Dobbiamo però fare attenzione a che il led non sia attraversato da una corrente eccessiva, pena bruciarlo.

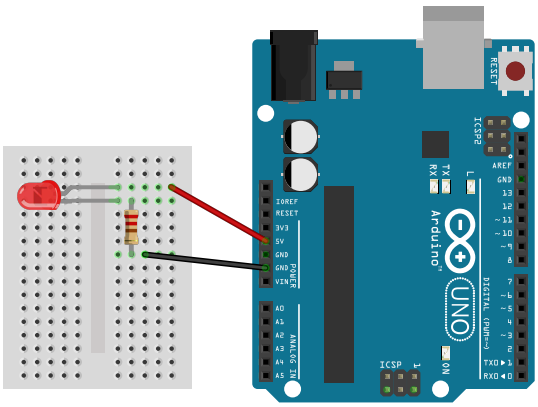
Vogliamo accendere un led utilizzando la differenza di potenziale offerta da Arduino. Per fare questo lo alimentiamo interponendo una opportuna resistenza in serie con esso. La corrente che può attraversare un comune led fornito in kit con Arduino deve essere quasi sempre inferiore a 20 mA, mentre la tensione di alimentazione varia tra 1,3 V per un led infrarosso a 3,5 V per un led blu. Applichiamo la seconda legge di Kirchhoff al circuito ora descritto e qui di seguito rappresentato.

$$V_A - Ri - V_{led} = 0 \tag{2.1}$$

dove $V_A = 5\,\text{V}$, R è la resistenza da trovare, i la corrente massima che vogliamo attraversi il led e V_{led} la caduta di tensione ai capi del led. Mettiamo in evidenza R per un led verde da 2 V.

$$R = \frac{V_A - V_{led}}{i} = \frac{5\,\text{V} - 2\,\text{V}}{20\,\text{mA}} = 150\,\Omega \tag{2.2}$$

Possiamo utilizzare una resistenza da 220 Ω. Il led ha il polo positivo associato al filo più lungo, rappresentato in figura dal terminale superiore.



2.5 Blink

Finora abbiamo usato Arduino come una pila: vogliamo iniziare a fargli svolgere delle operazioni sia pur elementari. Proviamo a fargli accendere e spegnere un led a intervalli di un secondo. Per istruire Arduino dobbiamo caricare nella sua memoria un apposito programma. Lo strumento per scrivere e caricare un programma su Arduino è l'IDE di Arduino. Non indichiamo in questo documento come installarlo; è facilmente indicato in rete dal sito di Arduino.org.

Il programma che potremmo definire l'"Hello world!" di Arduino si chiama "blink" ovvero lampeggiamento. Lo possiamo caricare dagli esempi disponibili nell'IDE e lo riportiamo qui di seguito senza una parte delle righe commentate.

I codici per Arduino sono scritti in una variante del C/C++. In ogni programma devono obbligatoriamente essere presenti le due function setup e loop. Nella prima sono contenuti dei settaggi preliminari, nella seconda tutte le elaborazioni che vanno ripetute incessantemente durante il funzionamento della scheda.

Bisogna sapere che sulla scheda, visibili alla destra del logo di Arduino, sono presenti tre led: quello superiore, affiancato dalla scritta L, è controllabile dall'utente. Con la riga 3 del programma stiamo dicendo alla scheda che il led integrato va fatto funzionare come un pin che riceve dati in uscita. La scritta LED_BUILTIN è una variabile di sistema; può essere un numero che indica il pin sulla scheda o una variabile come in questo caso.

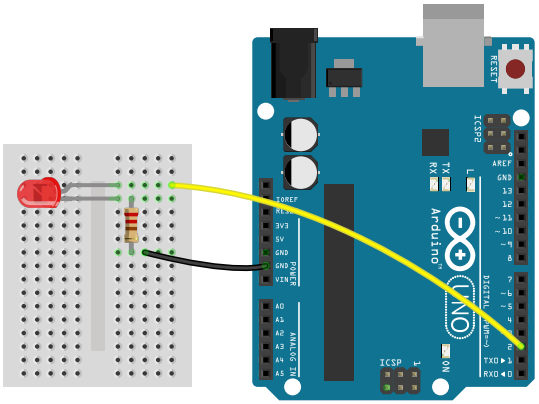
```
1  void setup() {
     // initialize digital pin LED_BUILTIN as an output.
3    pinMode(LED_BUILTIN, OUTPUT);
   }
5
   // the loop function runs over and over again forever
7  void loop() {
     digitalWrite(LED_BUILTIN, HIGH);
9    delay(1000);
     digitalWrite(LED_BUILTIN, LOW);
11   delay(1000);
   }
```

I pin della scheda sono classificabili in digitali e analogici. Quelli digitali possono funzionare in output o input e possono mandare e ricevere segnali su due stati: HIGH ovvero un segnale a 5 V o LOW ovvero un segnale a 0 V.

Nella function loop, alla riga 8, stiamo impostando il pin del led in modo che dia in uscita un segnale di tipo HIGH, cioè si accenda. Nella riga successiva diciamo alla scheda di attendere per 1000 millisecondi. Nelle due righe successive riportiamo lo stato del led su LOW ovvero lo spegniamo.

2.6 Controlliamo un led

L'esempio fondamentale precedente opera sul led della scheda. Possiamo controllare anche il led esterno che abbiamo fatto accendere nell'esempio prima mostrato. Come si può vedere dallo scheda di montaggio qui mostrato invece di alimentare il led dal pin dei 5 V lo alimentiamo dal pin digitale 2. Quando da questo, come nell'esempio del Blink, mandiamo in uscita un segnale HIGH da esso uscirà una tensione continua a 5 V che alimenterà il led esterno.

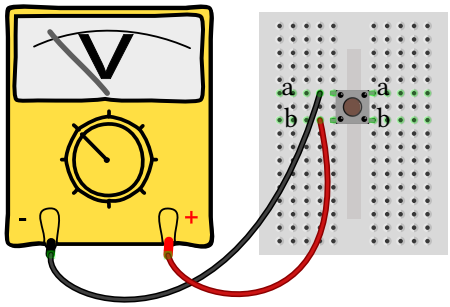


Il codice per gestire questa nuova configurazione è praticamente identico al precedente mostrato. L'unica differenza è il cambiamento del nome del pin da LED_BUILTIN, cioè il pin associato al led interno alla scheda, a 2, il pin a cui colleghiamo il led esterno.

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(2, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(2, HIGH);
  delay(1000);
  digitalWrite(2, LOW);
  delay(1000);
}
```

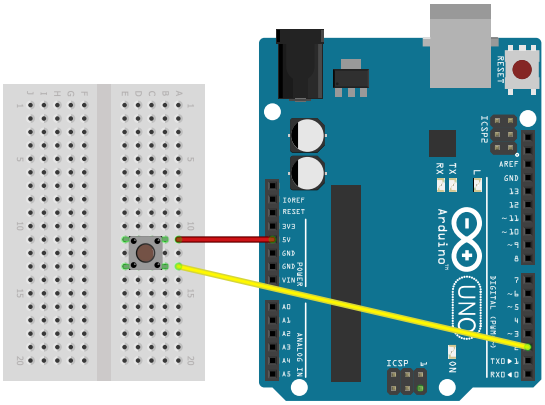
2.7 Mandiamo un segnale con un pulsante

Un modo per inviare dei segnali alla scheda è usando un pulsante. La tipologia di pulsante che più frequentemente è disponibile con i kit venduti per Arduino è del tipo monostabile: quando non è premuto l'interruttore è aperto e non passa corrente; viceversa quando è premuto. In particolare sul pulsante sono presente due coppie di terminali: tra il terminale a e b non passa corrente finché il pulsante non è schiacciato. Per verificare preliminarmente che il nostro pulsante stia funzionando correttamente installiamolo su una breadboard e colleghiamo i terminali di un tester come mostrato in figura.



Impostiamo il tester sulla misura di resistenze. Se il pulsante non è schiacciato il tester digitale dovrebbe mostrare il valore -1; se il pulsante è schiacciato dovrebbe mostrare il valore 0. Se qualcosa va storto è verosimile che il pulsante non sia ben inserito sulla breadboard.

Ora con il pulsante vogliamo controllare l'invio di un segnale alla scheda. Al terminale *a* colleghiamo l'alimentazione a 5 V e al terminale *b* un cavo verso il pin 2, che questa volta utilizzeremo in modalità input.



Prendiamo l'esempio del blink e inseriamo una riga (la 3 qui di seguito) in cui diciamo alla scheda che il pin digitale 2 deve funzionare come input. Inoltre (siamo alla riga 7) se lo stato letto dal pin 2 con l'istruzione digitalRead è LOW cioè a potenziale zero, allora (riga 8) accendiamo il led di Arduino. Altrimenti (riga 9) teniamo il led spento.

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(2,INPUT);
}

void loop() {
  if (digitalRead(2) == LOW) {
    digitalWrite(LED_BUILTIN, HIGH);
  } else {
    digitalWrite(LED_BUILTIN, LOW);
  }
  delay(100);
}
```

Se tutto va secondo le aspettative *questo esempio non dovrebbe funzionare!* Cosa è successo?

2.8 Mandiamo un segnale con un pulsante, che funzioni!

Comincio dicendovi cosa cambiare nel codice per fare funzionare il programma, anche se formalmente non dovrebbe cambiare niente: togliete la riga 12 che introduce una pausa di un decimo di secondo. Invece se diminuite la pausa a 50 millisecondi quasi funziona ovvero il led lampeggia quando non si preme il pulsante e sta spento quando lo si preme.

Non vi so indicare del perché di questo anomalo comportamento con la sola introduzione di una pausa, ma possiamo vedere cosa cambia nel comportamento elettrico di Arduino (o di una scheda clone) e come rimediare definitivamente.

Per svolgere un'indagine di quel che succede chiediamo ad Arduino di mostrarci cosa "vede" dal pin 2. Per fare ciò introduciamo il seguente codice.

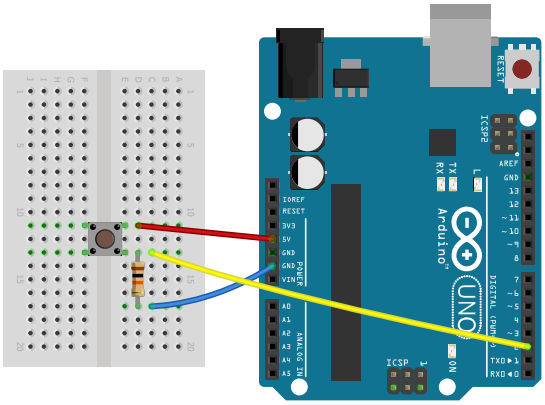
```
1  void setup() {
     pinMode(LED_BUILTIN, OUTPUT);
3    pinMode(2,INPUT);
     Serial.begin (9600);
5  }

7  void loop() {
     Serial.println(digitalRead(2));
9    if (digitalRead(2) == LOW) {
       digitalWrite(LED_BUILTIN, HIGH);
11   } else {
       digitalWrite(LED_BUILTIN, LOW);
13   }
     delay(100);
15 }
```

Per stabilire una comunicazione tra Arduino e computer dobbiamo aprire una porta di comunicazione, stabilendo per essa una velocità di trasferimento dati in bit al secondo: è quello che facciamo con la riga 4. Ora, con la riga 6, possiamo stampare sul terminale di Arduino, chiamato *Monitor seriale*, il valore letto dal pin 2. La scritta "ln" al termine di print sta ad indicare che una volta visualizzato il dato il Monitor deve andare a capo.

Se carichiamo questo codice vedremo sul Monitor una serie di zeri e uno che si susseguono ad intervalli di qualche secondo e, in corrispondenza di essi, il led che si accende e spegne senza che noi schiacciamo il pulsante. Se il tempo indicato in delay scende a 50 millisecondi, gli zeri e gli uno si

susseguono rapidamente, da cui il lampeggiamento del led. Ma persino eliminando completamente la riga con il delay vedremo la serie di uno e zeri, anche se il led non sembra più lampeggiare!

Il problema alla base di questo curioso comportamento è dovuto al fatto che quando non dovrebbe arrivare segnale al pin 2 pur tuttavia in esso può esserci un potenziale residuo diverso da zero: questa condizione è espressa dicendo che il pin si trova in uno *stato flottante*. Per assicurarci che ciò non avvenga è possibile inserire una apposita resistenza di circa 10 kΩ, come mostrato nella figura successiva.



In questo schema, la cui resistenza è detta di *pulldown*, quando l'interruttore è aperto allora il pin 2 è allo stesso potenziale del GND ovvero zero. Quando l'interruttore è chiuso, cioè quando premiamo il tasto, la corrente fluisce dal pin dei 5 V e si diparte tra la resistenza e il filo giallo: essendo questo quasi senza resistenza la corrente finisce quasi soltanto lì.

2.9 Costruiamo un partitore di tensione

Ci sono delle situazioni, specialmente quando usiamo una scheda diversa da Arduino Uno, ad esempio con Arduino Due, in cui la logica interna della nostra scheda funziona a 3,3 V e non a 5 V. Di per sé questo non è un problema; lo diventa quando usiamo un sensore predisposto per funzionare a 5 V in una scheda da 3,3 V. In questi casi possiamo ancora alimentare correttamente il sensore, dato che quasi tutte le scheda sono fornite anche di una alimentazione a 5 V, ma dobbiamo abbassare la tensione del segnale in uscita dal sensore affinché si adatti alla logica del nostro microcontrollore o cpu.

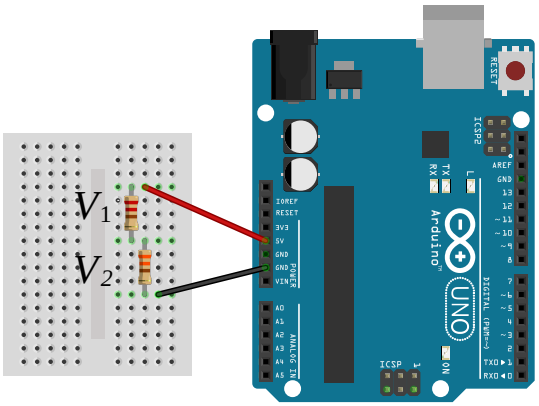
Per realizzare questo cambio di tensione possiamo usare un *partitore di tensione* realizzato con due resistenze in serie. Se abbiamo due resistenze in serie, R_1 e R_2, alimentate con una differenza di potenziale V allora la corrente che le attraversa vale:

$$i = \frac{V}{R_1 + R_2} \tag{2.3}$$

Ai capi delle due resistenze avremo una caduta di tensione che vale:

$$V_1 = R_1 \cdot i = V\frac{R_1}{R_1 + R_2} \quad ; \quad V_2 = R_2 \cdot i = V\frac{R_1}{R_1 + R_2} \tag{2.4}$$

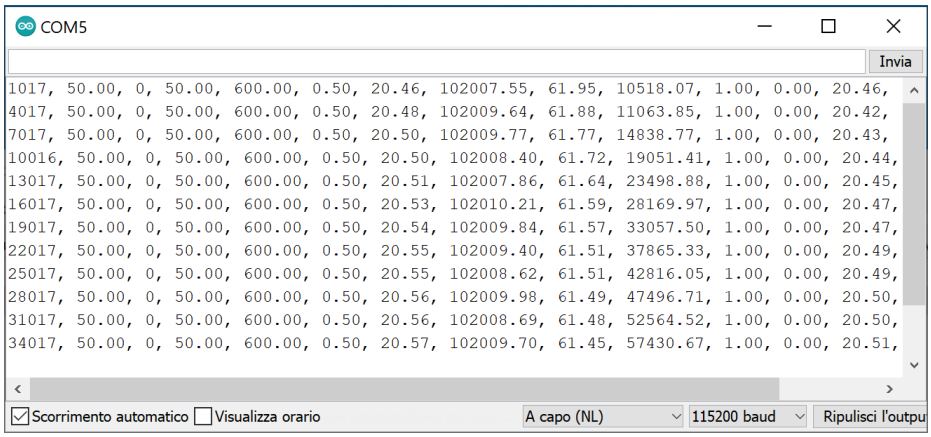
Ad esempio se V = 5,0 V, $R_1 = 220\,\Omega$ e $R_2 = 330\,\Omega$ allora la teoria ci darebbe $V_1 = 2{,}0$ V e $V_2 = 3{,}0$ V.



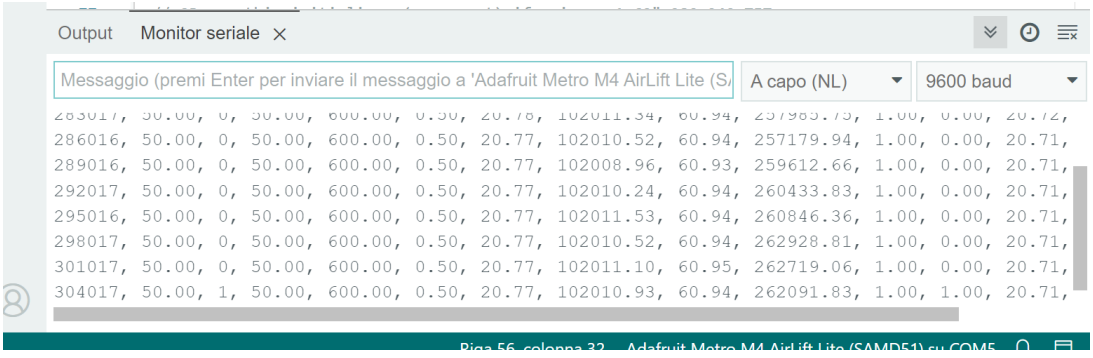
3 Raccolta dati

3.1 Lettura dati dallo schermo

Il modo fondamentale per visualizzare i dati provenienti dalle schede Arduino è attraverso il *monitor seriale* disponibile nell'IDE di Arduino. Il monitor è disponibile al menu *strumenti -> monitor seriale* sia nella versione 1.8 dell'IDE che nella 2.0. Tuttavia nella vecchia versione il monitor si apriva in una finestra separata, condivisa tra tutti gli sketch.



Invece nella nuova versione il monitor compare solo nella parte inferiore dello finestra di ogni singolo sketch: questo è stato espressamente voluto dagli sviluppatori della nuova versione.



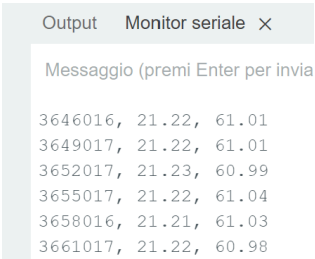
Lo scorrimento dei dati è preimpostato per mostrare sempre l'ultima riga ricevuta, o con la voce "scorrimento automatico" della versione 1.8 o le doppie freccette verso il basso che compaiono nella parte superiore destra del riquadro della versione 2.0.

Un aspetto fondamentale per una corretta visualizzazione è l'impostazione della velocità dell'interfaccia seriale: deve essere la stessa impostata nel programma che stiamo osservando in esecuzione.

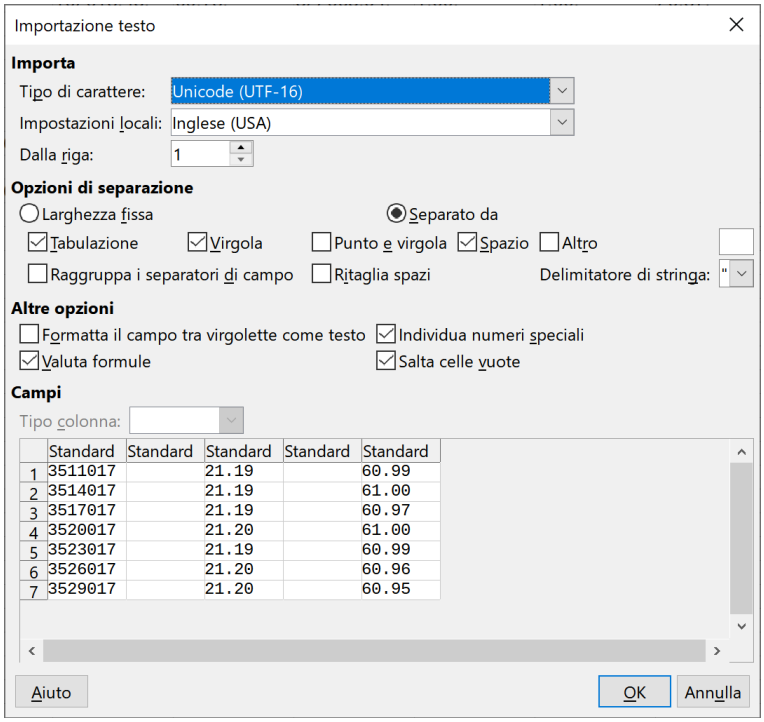
3.2 Copia dati dallo schermo

Con questa modalità di interazione con Arduino possiamo solo leggere i dati mentre scorrono. Per salvare più dati e copiarli in un file di testo o in un foglio elettronico possiamo fermare lo scorrimento automatico, selezionare col mouse la parte di testo da copiare e infine premere i tasti Ctrl + C. Al momento in cui scrivo questa possibilità non funziona con l'IDE 2: si tratta di un baco e non di una mancanza di funzionalità.

L'inserimento di questi dati in un foglio elettronico può avvenire in questa maniera. Partiamo dall'avere dei dati formattati come segue. I dati sono nella forma inglese, con il separatore di decimale dato da un punto, con i dati separati da virgole e anche da uno spazio bianco.



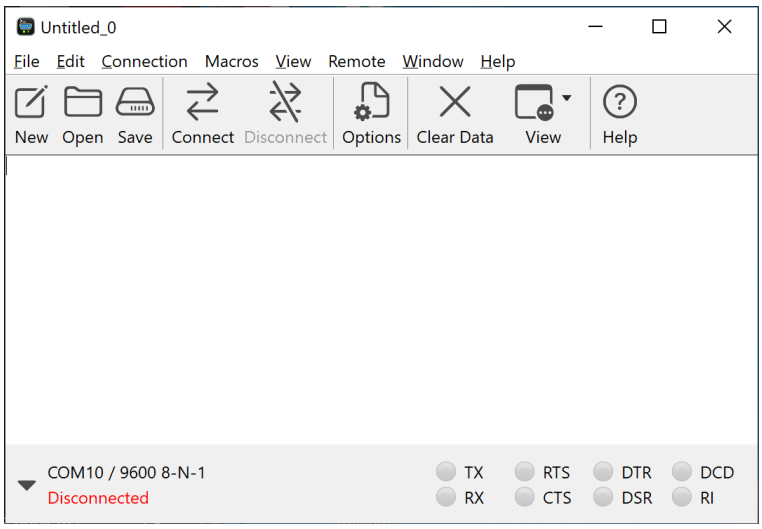
Se usiamo Libreoffice, apriamo il nostro nuovo file e a partire dalla cella in cui vogliamo inserire i dati andiamo nel menu alla voce modifica -> incolla speciale -> incolla testo non formattato: comparirà una finestra di questo tipo.



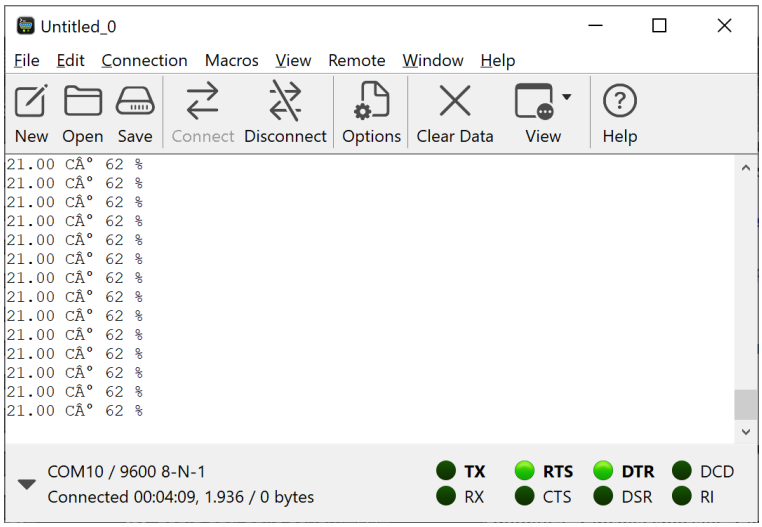
In questa finestra possiamo osservare che le impostazioni locali sono sul formato inglese. Come indicatore di separazione tra i dati sono stati indicati sia la virgola che lo spazio bianco: scegliamo quel che ci conviene e in particolare formattiamo i dati in uscita da Arduino in funzione delle possibilità di importazione che osserviamo qui. Una volta che i dati sono stati inseriti nel foglio elettronico compariranno con la nostra impostazione locale ovvero con il separatore di decimale con la virgola come usiamo in italiano (se abbiamo impostato Libreoffice per l'italiano!).

3.3 Coolterm

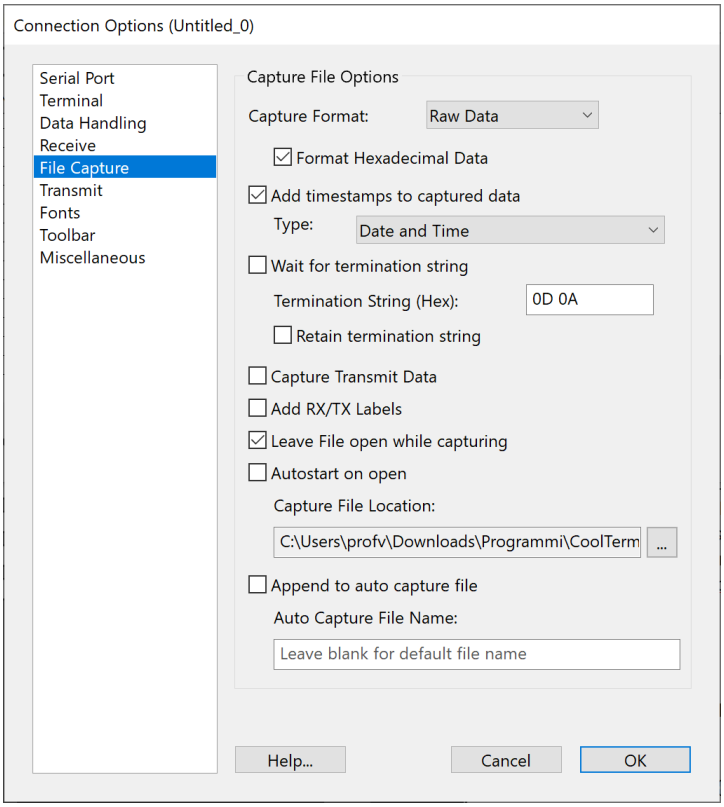
Per una acquisizione dati più automatizzata è possibile usare un programma esterno che monitori ed elabori i dati che passano attraverso la porta seriale del computer, senza usare l'IDE di Arduino. Uno dei più apprezzati programmi di questo tipo è Coolterm, che possiamo facilmente trovare e scaricare da Internet. Una volta installato e avviato dovremmo vedere una finestra simile a questa.



Il primo passo per il suo corretto funzionamento è stabilire la connessione con l'opportuna porta seriale pigiando il tasto "connect". Nel mio caso la finestra assume il seguente aspetto.

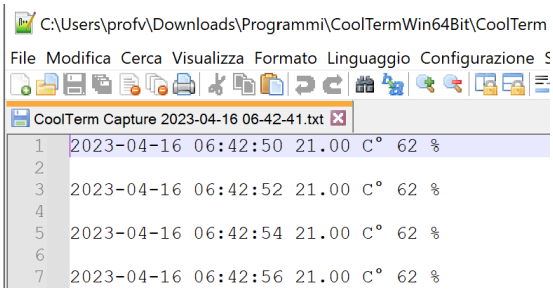


Il programma ha capito quale porta osservare, probabilmente perché solo una era in esecuzione. Tuttavia la velocità di comunicazione deve essere esplicitamente inserita e dobbiamo quindi conoscerla preventivamente. Per selezionare quest'ultimo parametro ed altri pigiamo il tasto "Options". La prima scheda che compare permette di selezionare il baudrate, ed è il parametro più importante. Ci sono innumerevoli altre voci e su essa non diciamo nulla, meglio lasciare tutto come preimpostato. Una scheda importante, nella prospettiva di salvare i dati, è "File Capture".



In questa scheda possiamo selezionare la cartella in cui salvare il flusso di dati proveniente dalla porta. Inoltre, con la la casella “Add timestamps to captured data”, possiamo aggiungere al flusso dati l’indicazione della data e dell’ora, come qui io ho fatto.

Ritornando alla schermata iniziale, per salvare i dati in un file andiamo al menu Connection -> Capture to text -> Start. Avrete notato il carattere anomalo che compariva nel flusso dati ovvero l’indicazione dei gradi affianco alla C di Celsius. Quel carattere viene poi salvato correttamente e può essere visualizzato opportunamente, come mostrato qui di seguito. Il file salvato è stato aperto col programma Notepad++; in esso compare in ogni riga anche la data e l’ora.



Quando finiamo la sessione di lavoro e chiudiamo il programma ci viene chiedo il salvataggio dei dati: non si tratta dei dati passati dalla porta seriale, ma le impostazioni del programma che abbiamo inserito; le possiamo così aprire la volta successiva per una immediata nuova disponibilità.

4 Schermi

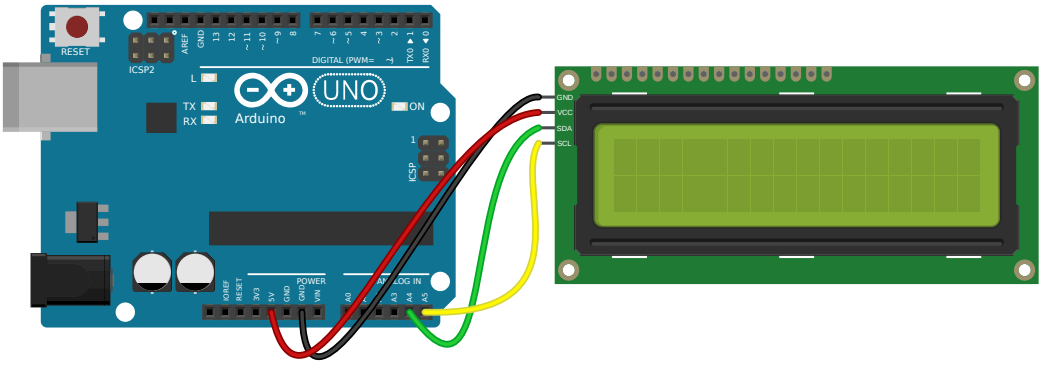
4.1 Schermo LCD 1602

Per visualizzare i dati senza computer abbiamo bisogno di uno schermo. Se ne trovano in commercio di varia tipologia. Non saprei cosa consigliare come primo acquisto: oggi si trovano schermi oled ed LCD per pochi euro. L'affidabilità e robustezza di un classico LCD di poche righe dovrebbe essere migliore di schermi oled di dubbia provenienza. Tradizionalmente in molti kit per Arduino (tra cui l'Arduino starter kit) si trova uno schermo LCD a due righe e sedici caratteri chiamato "lcd 1602". La versione base ha numerosi pin e un interfacciamento un po' complicato. La versione I2C completa ha solo quattro pin ed è molto più semplice da usare: questa è la versione che qui propongo.

4.1.1 Montaggio delle apparecchiature

Il display è dotato di quattro pin:

1. Il pin denominato GND va messo a terra e collegato al pin GND su Arduino;
2. Il pin VCC va collegato all'alimentazione: lo colleghiamo al pin a 5,0 V di Arduino;
3. Il pin SCL è indicato come "I2C serial bus clock input": lo colleghiamo al pin A5;
4. Il pin SDA è indicato come "I2C serial bus data (or SPI input)": lo colleghiamo al pin A4.



La comunicazione tra sensore e Arduino è realizzata attraverso il protocollo di comunicazione I2C. Non abbiamo bisogno di conoscere le specifiche di questo protocollo: la libreria Wire si occupa di questi particolari. Abbiamo però bisogno anche della libreria "LiquidCrystal_I2C" che può essere scaricata all'indirizzo LiquidCrystal_I2C.

Codice per Arduino

```
1  #include <Wire.h>
   #include <LiquidCrystal_I2C.h>
3
   LiquidCrystal_I2C lcd(0x27,16,2);
5  void setup()
   {
7    lcd.init();
     lcd.backlight();
9  }
   void loop()
11 {
       lcd.setCursor(0, 0);
13     lcd.print("Hello world");
       delay(2000);
15     lcd.clear();

17     lcd.setCursor(0, 1);
       lcd.print("Ciao mondo");
19     delay(2000);
       lcd.clear();
21 }
```

- La riga 2 richiama la libreria LiquidCrystal_I2C;
- La riga 4 indica l'indirizzo a cui porre il dispositivo (0x27) e il fatto che lo schermo abbia 16 righe e 2 colonne.
- La riga 7 inizializza il dispositivo.
- La riga 8 accende la retroilluminazione; senza di essa lo schermo è veramente poco visibile.
- La riga 12 porta il cursore nella prima riga e prima colonna.
- La riga 13 stampa la scritta partendo dalla posizione del cursore prima selezionata
- La riga 14 mette Arduino in pausa per due secondi.
- La riga 15 cancella il contenuto dello schermo.
- Le righe successive stampano un'altra scritta sulla seconda riga e poi cancellano tutto.

Parte II

Misurare grandezze fisiche

5 Misure di distanza

5.1 HC-SR04

5.1.1 Introduzione

In natura alcuni animali, il più noto dei quali è il pipistrello, possono usare la ecolocalizzazione per determinare le distanze degli oggetti davanti a sé. L'idea è quella di usare un'onda sonora in modo da farla riflettere sugli oggetti, ascoltandone l'eco: più tardi arriva l'eco, maggiore è la distanza dell'oggetto. La tecnologia ha concepito un apparecchio che usa i suoni in maniera simile, chiamandolo sonar. Il sonar (SOund Navigation And Ranging) è usato soprattutto per la localizzazione in mare, sott'acqua. Una delle applicazioni più frequenti del sonar, usata però in aria, è quella dei sensori di parcheggio.

Tra i sensori disponibili per le schede Arduino vi è la scheda HC-SR04, dal costo di pochi euro. Questo sensore permette di determinare la distanza da un oggetto con l'emissione di un breve segnale sonoro, misurando il tempo intercorrente tra la fine del suono emesso e l'eventuale ricezione dell'eco riflesso dall'oggetto. Non è il sensore che determina direttamente la distanza dell'oggetto: il sensore misura solamente il ritardo dell'eco rispetto al suono di partenza.

Tra i misuratori di distanza tramite sonar disponibili per i laboratori di fisica RTL abbiamo degli strumenti che funzionano come black box, senza alcuna possibilità di personalizzare il modo in cui i dati diventano disponibili o vengono elaborati. Al contrario, lo strumento che andiamo costruendo, pur avendo caratteristiche tecniche inferiori (ma solo per una scelta di costo), sarà completamente personalizzabile. Questo ci consentirà una piena comprensione di come avvenga l'acquisizione dati, pur facendo riferimento a principi di fisica elementare.

Poiché la distanza determinata dipende dalla velocità del suono in aria e questa dipende significativamente dalla temperatura, avremo bisogno di integrare il tempo misurato dal sensore con una misura di temperatura.

La misura della distanza avviene seguendo questo processo.

1. Il sensore invia un suono di alta frequenza e breve durata (una serie di otto impulsi della durata complessiva di 10 μs).
2. L'oggetto investito dal suono lo riflette generando un eco.
3. Il sensore, in ascolto, riceve l'eco e misura il tempo intercorso tra la fine del suono emesso e la sua ricezione.
4. Calcoliamo la distanza tenendo conto della velocità v (costante) del suono nell'aria (circa 343 m/s). Lo spazio percorso dall'onda sonora è $s = v \cdot t$.
5. La distanza a cui si trova l'oggetto è la metà della distanza percorsa dall'onda.

Inoltre bisogna tener conto che la velocità del suono nell'aria dipende significativamente dalla sua temperatura (a 0 °C è solamente 331 m/s contro i 343 m/s alla temperatura standard di 20 °C). Tuttavia la relazione che lega questa velocità alla temperatura è (con ottima approssimazione) lineare nel campo delle usuali temperature che possiamo incontrare in laboratorio.

Di questa scheda sono esistiti in commercio due tipologie indicate con lo stesso nome: la seconda (quella consigliata per un acquisto attuale) è caratterizzata nella parte posteriore da due file parallele di resistenze. Le immagini che si trovano in molti siti fanno riferimento alla prima versione, che non dovrebbe essere più in commercio.

5.1.2 Strumenti necessari

Proponiamo qui di seguito tre modalità di utilizzo del sensore: la prima è quella che viene indicata dal costruttore. La seconda fa uso di un condensatore da 100 μF e una resistenza da 10 Ω ed è la scelta da me consigliata (con o senza resistenza). La terza è quella necessaria per l'uso con schede tipo Arduino Due, con una logica a 3,3 V. In precedenza avevo proposto un unico altro schema con un sensore di temperatura: lo ho eliminato non ritenendolo più significativo.

Prima versione:

1. una scheda Arduino (Uno)
2. sensore di distanza HC-SR04

Seconda versione:

1. una scheda Arduino (Uno)
2. sensore di distanza HC-SR04
3. una breadboard per gestire tutte le linee di alimentazione
4. un condensatore da 100 μF
5. una resistenza da 10 Ω

Terza versione:

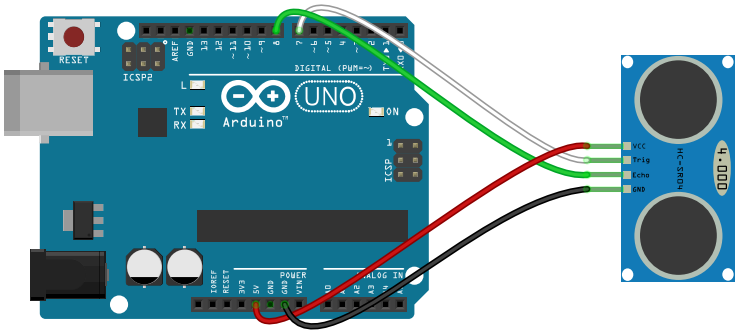
1. una scheda Arduino (Due)
2. sensore di distanza HC-SR04
3. una breadboard per gestire tutte le linee di alimentazione
4. una resistenza da 330 Ω
5. una resistenza da 470 Ω

5.1.3 Schema di montaggio

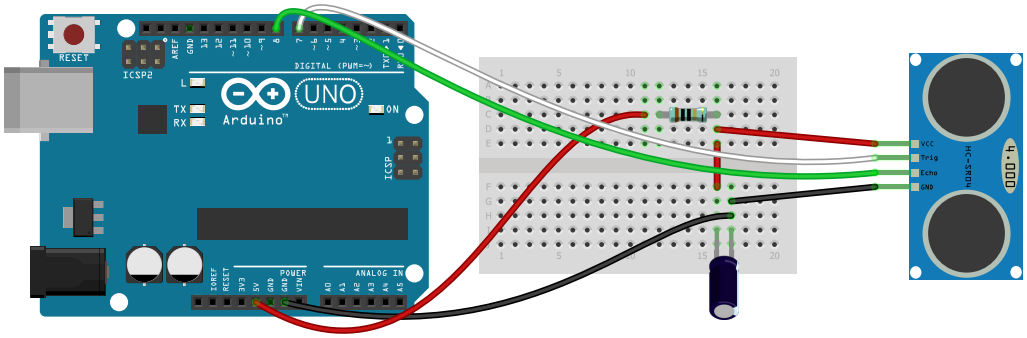
La scheda HC-SR04 è materialmente costituita da una schedina su cui sono montati un trasmettitore, un ricevitore (i due oggetti cilindrici più grandi visibili sulla scheda) e una serie di circuiti le cui caratteristiche sono per noi di scarso interesse. Fondamentale è invece conoscere i quattro pin che permettono il collegamento del sensore. Due pin sono dedicati all'alimentazione della scheda: uno va a massa (GND) e uno va a 5 V (vcc). Gli altri due pin servono per inviare il comando che lancia un impulso sonoro (Trig) e per ricevere l'indicazione del tempo trascorso fino all'avvenuta ricezione dell'eco (Echo). La scheda è accreditata per poter misurare distanze comprese tra 2 cm e 4 m con una accuratezza massima di 3 mm. Il segnale sonoro inviato è un suono a 40 kHz.

La scheda funziona solo a 5 V: non è direttamente compatibile con schede (come Arduino Due) con cpu a 3,3 V. In particolare il segnale inviato da Arduino alla scheda (sul pin del Trig) può anche essere a 3,3 V, ma l'alimentazione e il segnale di ritorno che passa sul pin Echo è a 5 V: se la scheda non li può gestire è necessario fare uso di un partitore di tensione, come descritto nella terza versione qui di seguito indicata.

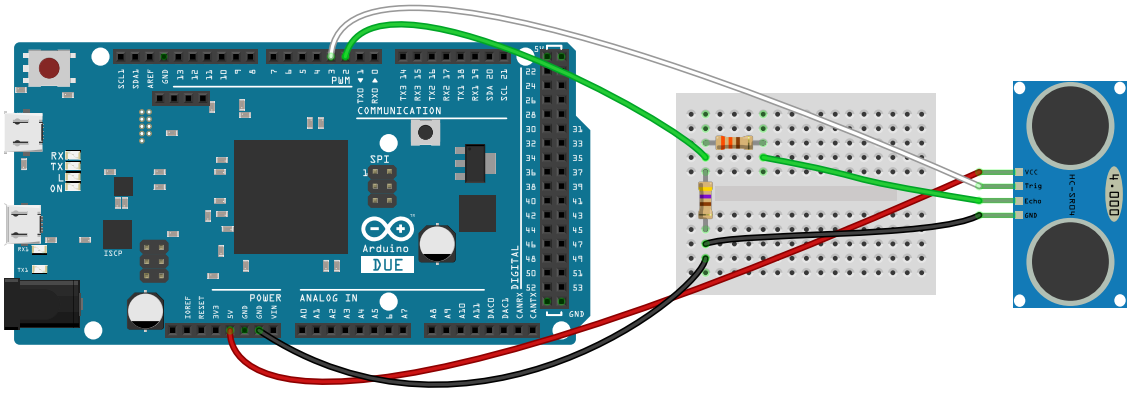
Prima versione



Seconda versione



Terza versione



Prima versione

La prima versione ripropone lo schema di montaggio ufficialmente proposta dal produttore. Per le ragioni che indico nel paragrafo relativo alle prestazioni questo schema è da me sconsigliato.

Seconda versione

Questa versione del collegamento è stata realizzando seguendo le indicazioni che la Maxbotix propone per i suoi sonar. Il condensatore e la resistenza sono utili a uniformare la tensione di alimentazione del sensore e questo contribuisce in maniera determinante alle sue prestazioni. Secondo le mie prove usare un condensatore più grande non cambia le prestazioni, invece la presenza della resistenza può essere superflua.

Terza versione

In questa terza versione, variante della prima, il segnale del pin Echo viene inviato all'estremità di un partitore di tensione costituito da due resistenze. Il segnale viene poi estratto dal punto tra le due resistenze e il GND: in questo modo il segnale viene abbassato a circa 3 V e può essere gestito (senza fare danno) da Arduino Due o schede analoghe. In particolare i valori delle due resistenze sono motivati dalla loro reperibilità nei kit standard di resistenze. Si può dimostrare che la differenza di potenziale tra GND e il punto tra le due resistenze vale:

$$\Delta V = V_{\text{in}} \cdot \frac{R_1}{R_1 + R_2} = 5{,}0\,\text{V} \cdot \frac{470\,\Omega}{(470\,\Omega + 330\,\Omega)} \simeq 2{,}9\,\text{V} \tag{5.1}$$

Qualche autore propone di usare un'unica resistenza compresa tra 1000 Ω e 20000 Ω sulla linea a 5 V, ma senza una logica precisa per questa scelta: per questo motivo non la propongo in questi appunti.

5.1.4 Codice per l'utilizzo

```
1  int trigPin = 7;
   int echoPin = 8;
3  unsigned long tUs = 0;
   void setup()
5  {
     Serial.begin(115200);
7    pinMode(trigPin, OUTPUT);
     pinMode(echoPin, INPUT);
9    digitalWrite(trigPin, LOW);
   }
11 void loop()
   {
13   digitalWrite(trigPin, LOW);
     delayMicroseconds(2);
15   digitalWrite(trigPin, HIGH);
     delayMicroseconds(10);
17   digitalWrite(trigPin, LOW);
     tUs = pulseIn(echoPin, HIGH);
19   delay(50);
     float v = 331.5+0.6*26; // m/s
21   float d = tUs*v/2000.0;
     Serial.println(d,0);
23 }
```

- La riga 1 definisce il nome della variabile che contiene il pin a cui inviare il segnale per fare partire l'emissione del suono.
- La riga 2 definisce il nome della variabile che contiene il pin da cui ricevere il tempo per ricevere il suono riflesso.
- *Dalla riga 4 inizia la funzione iniziale* di setup.
- La riga 6 seleziona la velocità di comunicazione tra Arduino e l'esterno.
- La riga 7 attiva il pin trigPin in modalità di output.
- La riga 8 attiva il pin echoPin in modalità di input.
- La riga 9 porta al livello basso l'uscita del trigger, ovvero la scheda non sta emettendo alcun suono.
- *Dalla riga 11 inizia la funzione che ci dà la distanza*.
- La riga 13 riporta al livello basso l'uscita del trigger,
- La riga 15 emette un segnale dal trigger.
- La riga 16 aspetta almeno 10 microsecondi.
- La riga 17 spegne il segnale dal trigger. Qualche autore mette una pausa di qualche microsecondo dopo questa funzione.
  In realtà, provando a mettere un tempo di attesa eccezionalmente lungo dopo questa istruzione, ho potuto costatare che anche con intervalli di mille e più microsecondi non si altera in alcun modo il tempo che viene misurato con l'istruzione successiva.
- La riga 18 aspetta una risposta dal microfono; se l'eco viene ricevuto allora viene riportato dopo quanti microsecondi dall'invio del segnale l'eco è stato ricevuto. Questa funzione mette in attesa tutta la scheda Arduino.

Ho potuto provare sperimentalmente che il tempo rilevato non è calcolato da Arduino, ma dal sensore: ne è riprova quanto osservato alla riga precedente.

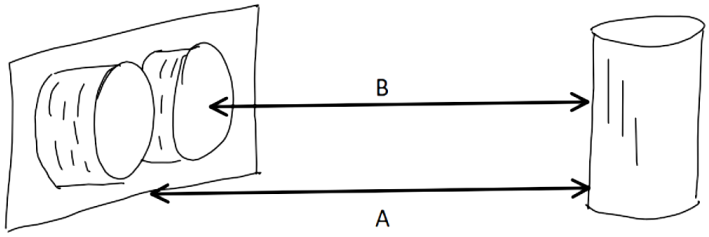
- pausa di 50 millisecondi.
- La riga 20 determina la velocità del suono in funzione della temperatura (qui è settata a 26 °C)
- La riga 21 determina la distanza = velocità per tempo (diviso 2000 per trasformare il risultato in millimetri, tenendo conto che lo spazio percorso è il doppio della distanza del sensore dal target).
- La riga 22 stampa la distanza in millimetri.

5.1.5 Studio delle prestazioni

Il sensore HC-SR04 costituisce gioe e dolori per i tanti che lo utilizzano. Il comportamento sembra a volte imprevedibile e a seconda della sua modalità di utilizzo emergono fastidiosi problemi di costanza di comportamento e fluttuazione casuale delle misure. Nei grafici successivi ho riportato, nella forma di istogramma, i risultati della raccolta di 1000 misure successive, effettuate ponendo il sensore davanti ad una parete piatta: questa raccolta è stata fatta per distanza crescenti da 10 cm a 40 cm, distanze scelte in quanto funzionali alle esperienze che ho svolto successivamente. La scheda utilizzata è di quelle di seconda generazione (riconoscibili per le due file parallele e verticali di resistenze accanto al chip centrale).

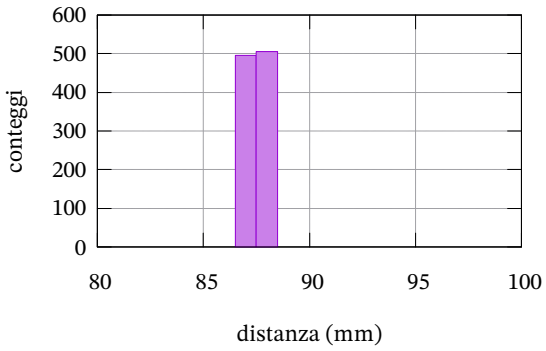
La distanza presa in considerazione per le misure è quella tra la scheda e l'oggetto target, indicata nel disegno successivo come A. La distanza che emergerà da queste misurazioni come quella più corretta è la distanza B tra l'estremità delle capsule del sensore e l'oggetto target.

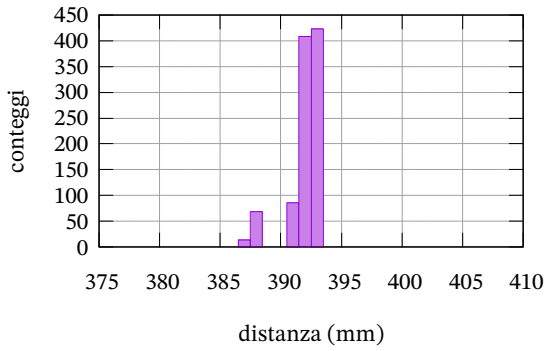
Le medie e la deviazione standard sono calcolate sui dati presi in millimetri. La deviazione standard appare sempre più contenuta di quanto farebbe pensare il grafico perché abbiamo sempre un picco prevalente di valori misurati.



Comportamento senza condensatore e resistenza su Arduino Uno originale

In questa modalità le misure sono sistematicamente inferiori al valore atteso, di una quantità corrispondente alla lunghezza del cilindro che costituisce il microfono e l'altoparlante del sensore: non so se sia una particolare casualità o meno. La dispersione dei dati è ottima per la distanza più breve, buona per quella maggiore. Si nota la tendenza alla comparsa di un secondo picco. L'accuratezza delle misure (compensando il sistematico valore inferiore) può essere valuta in circa mezzo centimetro, tenendo conto anche del secondo picco che quasi sempre compare per le distanze maggiori.

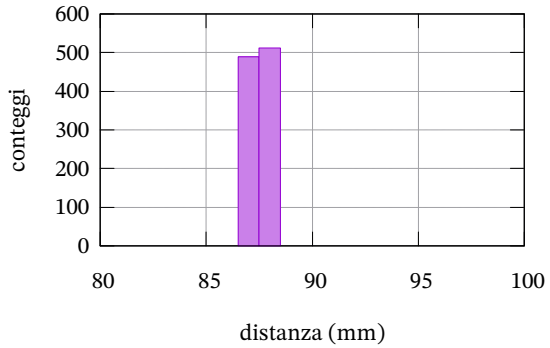


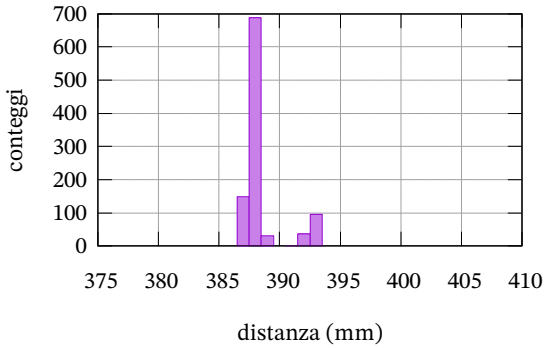


distanza target (mm)	distanza media misurata (mm)	σ (mm)
100	87,5	0,5
400	392	1,4

Comportamento senza condensatore e resistenza su Arduino Uno compatibile

Ci si potrebbero aspettare gli stessi risultati, ma per la distanza maggiore l'altezza dei due picchi precedenti si è invertita. La motivazione la associo con il problema dell'allineamento tra piano del sensore e piano dell'oggetto target: basta una lievissima rotazione e i risultati possono oscillare tra l'avere il primo o il secondo picco più elevato.

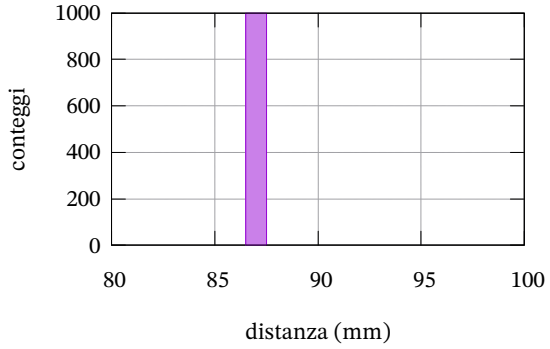


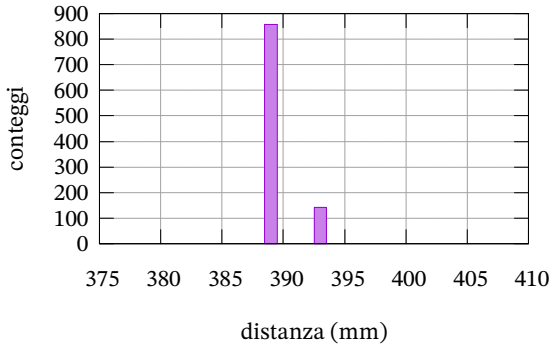


distanza target (mm)	distanza media misurata (mm)	σ (mm)
100	87,5	0,5
400	388,5	1,7

Comportamento senza condensatore e resistenza su Adafruit M4 Metro

La scheda è basata su un chip ARM Cortex M4: la sua programmazione e aspetto esteriore è del tutto simile ad Arduino Uno, ma non mancano le differenze. La cpu è molto più veloce ed a 32 bit e questo costringere a volte a cambiare qualche piccola istruzione. Anche qui ci si potrebbero aspettare gli stessi risultati, ma si nota una minore dispersione anche se i valori fondamentali sono gli stessi. La deviazione standard per la distanza maggiore è uguale a quella ottenuta con Arduino uno, ma i due picchi sono più netti ed in certe condizioni si possono ridurre quasi completamente al primo.

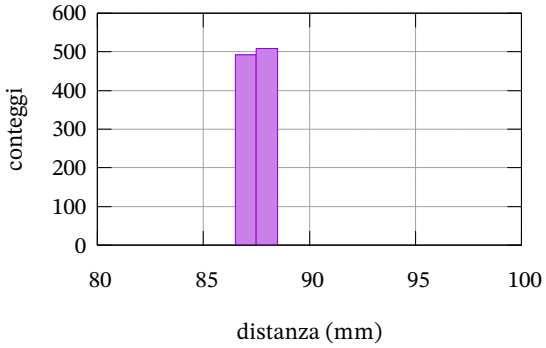


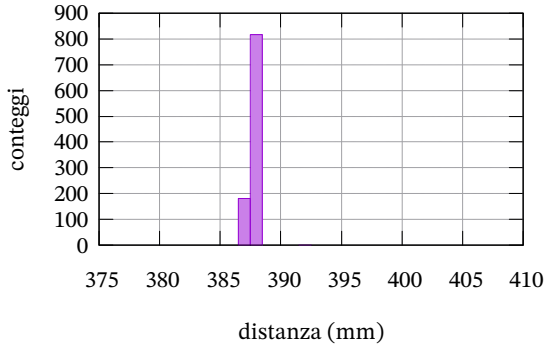


distanza target (mm)	distanza media misurata (mm)	σ (mm)
100	87	0,0
400	389,6	1,4

Comportamento con condensatore su Arduino Uno originale

Abbiamo un netto miglioramento per la distanza maggiore. Per la distanza minore i risultati già buoni non migliorano ulteriormente.

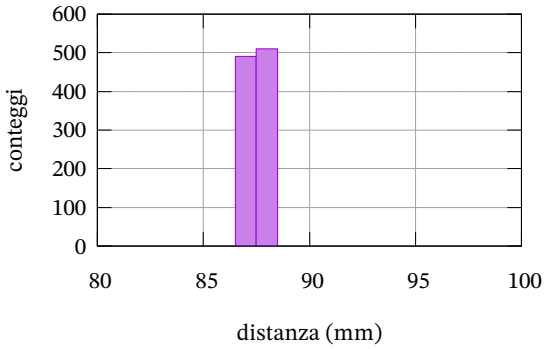


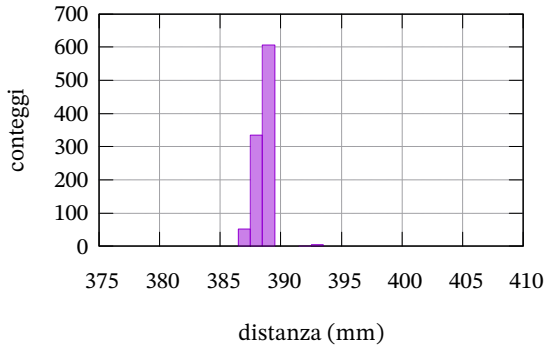


distanza target (mm)	distanza media misurata (mm)	σ (mm)
100	87,5	0,5
400	387,8	0,4

Comportamento con condensatore e resistenza su Arduino Uno originale

Purtroppo non si notano ulteriori miglioramenti. Anzi le curve si allargano leggermente.





distanza target (mm)	distanza media misurata (mm)	σ (mm)
100	87,5	0,5
400	388,6	0,7

Il problema dell'alimentazione

Il sensore, per il suo funzionamento, ha bisogno di una corrente piccola, ma non trascurabile. É possibile migliorarne le prestazioni fornendoli un'alimentazione esterna rispetto a quella direttamente fornita da Arduino? Per le prove che ho potuto fare posso dire di no: le prestazioni rimangono uguali.

Conclusioni

1. Bisogna controllare sempre, prima dell'utilizzo, che i contatti tra le varie componenti del circuito siano ben stabili.
2. Già in condizioni statiche basta un lieve disallineamento tra sensore e oggetto target per dare segnali spuri e complessi.
3. Il sensore dà le migliori prestazioni nella versione con il solo condensatore.
4. Le misure effettuate hanno preso come punto di riferimento per la distanza la base della scheda del sensore: tutto lascia pensare che il riferimento più opportuno sia l'estremità delle capsule del microfono e altoparlante (alte 11 mm): in questa maniera le misure risulterebbero estremamente allineate al valore atteso.
5. L'accuratezza delle misure non supera il mezzo centimetro nelle condizioni reali di utilizzo per la singola misura, se non per le distanze minori.
6. I dati ottenuti riguardano misurazioni in rapida successione, ma in condizioni statiche. Per misure in condizioni dinamiche rimando ad esperienze successive.

5.2 VL53L1X

5.2.1 Introduzione

L'idea alla base del sonar può essere usata anche con altri tipi di segnali oltre alle onde sonore. Ad esempio si può usare un fascio di microonde, come nei radar, o addirittura la luce. Il sensore di distanza VL53L1X della ST usa un raggio laser ad infrarossi per determinare la distanza di oggetti compresa tra pochi cm a circa 4 m. Il concetto alla base del sensore è la misura del "time of light", del tempo di volo, di un raggio laser che il sensore emette ad intervalli regolari ricevendo la luce riflessa in una matrice di sensori $16 \cdot 16$ e determinando il tempo intercorso (il time of flight) tra l'emissione e la ricezione. Essendo la velocità della luce nel vuoto c (ovvero nell'aria) costante e nota, questo tempo Δt ci consente di determinare la distanza d considerata:

$$d = \frac{c\Delta t}{2} \tag{5.2}$$

L'espressione è divisa per 2 perché il tempo è quello che il raggio emesso impiega per coprire la distanza all'andata e ritorno.

L'aspetto straordinario è che, considerando la velocità della luce, questo tempo è inferiore al nanosecondo, ma può essere misurato determinando la distanza con una accuratezza che può raggiungere qualche millimetro su distanze di 2 m. Tutta questa tecnologia è inoltre inglobata in un oggetto grande come un seme di grano.

Questo sensore può essere utilizzato per misure statiche molto precise o per misure continue ad un tasso massimo di 50 misurazioni al secondo, ma con minor precisione. Nelle righe seguenti propongo due varianti di codice per le due configurazioni. C'è inoltre la possibilità di utilizzare un sottoinsieme della matrice di fotosensori di cui è dotata per ottimizzare l'angolo di apertura del fascio riflesso ricevibile, ma per questi aspetti rimando alla documentazione tecnica fornita dalla ST.

Il sensore emette un fascio laser nell'infrarosso ed è sensibile alla luce ambientale. Tra le funzionalità presenti c'è anche quella di indicarci il livello di luce ambientale e il livello del segnale ricevuto, in modo da trovare le condizioni per un miglior utilizzo. Il "campo visivo" del sensore è di circa 27° e può essere ristretto operando sulla matrice dei fotosensori. A seconda della luce ambientale possiamo settare tre modalità di funzionamento dette Long, Medium e Short, dalla portata differente. Nell'utilizzo in laboratorio le più indicate sono le ultime due.

Esiste un parametro, detto Time budget, che determina il tempo utile per una misurazione: maggiore è questo tempo e più accurata e precisa sarà la misurazione. Tuttavia per un utilizzo ad alta frequenza dovremo diminuire anche la portata.

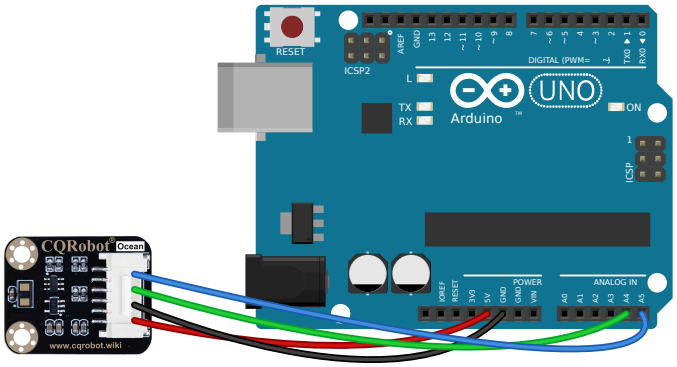
Il sensore, per essere utilizzato, ha bisogno di una apposita libreria. Quella da noi utilizzata può essere trovata al seguente indirizzo `https://www.arduino.cc/reference/en/libraries/vl53l1x/` in formato zip. Questo file va caricato nell'IDE di Arduino: dalla voce del menù princiale "Sketch" si passa a "#include libreria" e infine "Aggiungi libreria da file .zip" con la quale si apre una finestra di dialogo dalla quale possiamo specificare il file precedentemente scaricato. L'esempio di utilizzo che segue si base sugli esempi forniti nella libreria.

5.2.2 Strumenti necessari

Il nostro apparato di misurazione è costituito da tre componenti:

1. una scheda Arduino (uno)
2. sensore di distanza VL53L1X (CQRobot)
3. terza mano

5.2.3 Schema di montaggio



Il sensore è fornito di un connettore con sei cavi: per il suo utilizzo ne usiamo solo quattro come rappresentato in figura.

- Il pin VCC va collegato al pin 5 V di Arduino.
- Il pin GND va collegato al pin GND di Arduino.
- Il pin SDA va collegato al pin A4 di Arduino.
- Il pin SCL va collegato al pin A5 di Arduino.

I nomi dei pin del sensore sono chiaramente indicati sul suo retro.
La terza mano ci serve per tenere in posizione il sensore.

5.2.4 Codice per l'utilizzo

```
1  #include <Wire.h>
   #include <VL53L1X.h>
3  VL53L1X sensor;
   int d = 0;
5  void setup()
   {
7    Serial.begin(115200);
     Wire.begin();
9    Wire.setClock(400000);
     sensor.setTimeout(500);
11   if (!sensor.init())
     {
13     Serial.println("Failed to detect and initialize sensor!");
       while (1);
15   }
     sensor.setDistanceMode(VL53L1X::Long);
17   sensor.setMeasurementTimingBudget(250000);
     // sensor.setROISize(16,16);
19   }
   void loop()
21 {
     d = sensor.readSingle(true);
23   Serial.print(d);
     Serial.println();
25 }
```

- Le righe 1 e 2 caricano le librerie relative al protocollo I2C e al sensore.
- La riga 3 cambia il nome con cui richiamare le funzioni della libreria del sensore.
- La riga 4 definisce la variabile in cui memorizzare la distanza rilevata dal sensore.
- *Dalla riga 5 inizia la funzione iniziale* di setup.
- La riga 7 seleziona la velocità di comunicazione tra Arduino e l'esterno.
- Le righe 8 e 9 attivano il protocollo I2C e la frequenza a cui deve funzionare.
- La righe da 11 a 15 fanno un controllo iniziale per sapere se il sensore sta funzionando correttamente; altrimenti si blocca l'esecuzione.
- La riga 16 imposta il sensore in modalità "Long".
- La riga 17 imposta in tempo di attesa del sensore.
- La riga 18 (commentata) permette di selezionare il numero di sensori attivi nella matrice dei fotosensori.
- *La riga 20 inizia il ciclo principale.*
- La riga 22: viene fatta una sola misura della distanza; il parametro "true" significa che si aspetta che il sensore sia pronto per una nuova misurazione.
- La riga 23: la distanza viene stampata.

Il codice prima presentato è adatto per misure precise e statiche. Per misure ripetute e dinamiche (come seguire il moto di un pendolo) possiamo sostituire le righe 16 e 17 con le seguenti.

```
1  sensor.setDistanceMode(VL53L1X::Short);
   sensor.setMeasurementTimingBudget(50000);
```

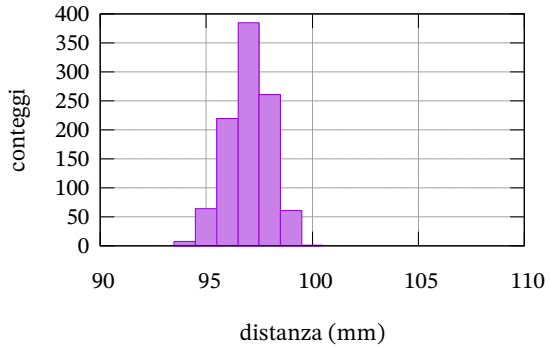
5.2.5 Studio delle prestazioni

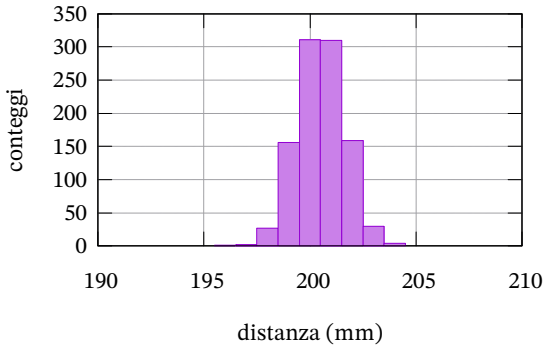
Il sensore vl53l1x fornisce un'ottima risposta, con la maggior parte delle misure centrate sulla distanza prevista e con una dispersione che descrive chiaramente una gaussiana, come ci aspetteremmo quando gli eventuali errori sistematici sono ridotti al minimo. La media misurata è in ottimo accordo con la misura della distanza impostata, con la tendenza alla sottostima per le distanze minori e la sovrastima per quelle maggiori. La deviazione standard è molto contenuta.

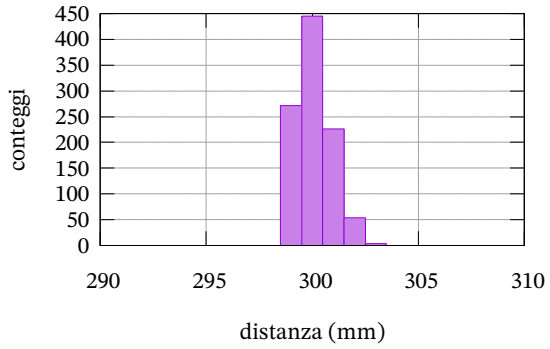
Il sensore può essere collegato direttamente ad Arduino. Il piano della scheda deve essere parallelo alla superficie dalla quale misurare la distanza.

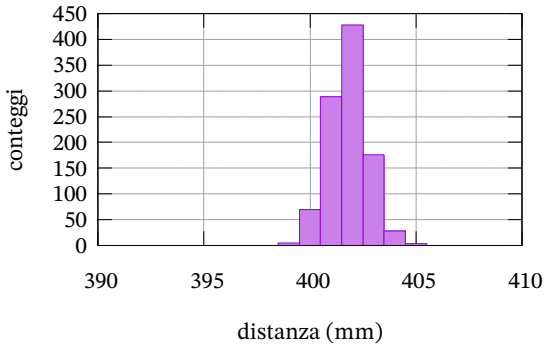
I dati successivi sono presi con le linee di codice suggerite per brevi distanze e misure ripetute. Sono state prese circa 20 misure al secondo. L'oggetto target era un'ampia superficie bianca, liscia e opaca. Il codice per i grafici si trova al capitolo [C].

La dispersione osservata è di circa 1 mm, ma l'accuratezza complessiva è di circa 3 mm, considerando la larghezza delle curve ottenute.









distanza target (mm)	distanza media misurata (mm)	σ (mm)
100	97,0	1,0
200	200,5	1,1
300	300,1	0,9
400	401,8	1,0

5.3 GP2Y0A21YK0F

5.3.1 Introduzione

Esiste una tipologia di sensori di distanza che utilizza, come la precedente, un fascio di luce infrarossa, ma che determina la distanza con la tecnologia detta ”position sensitive detector”. Questa tecnologia ha differenti implementazioni e non sappiamo quale sia stata utilizzata dalla Sharp. Possiamo però dire che l'idea alla base è quella di misurare l'intensità del raggio di luce riflesso proveniente dall'oggetto di cui si vuol misurare la distanza. Questa intensità è associata all'intensità della corrente in uscita dal fotosensore posto nell'apparecchio. Nel caso particolare del nostro apparecchio noi abbiamo in uscita una tensione che è approssimativamente inversamente proporzionale alla distanza da misurare.

Legame tensione - distanza

Purtroppo la correlazione tra questa tensione e la distanza non è facilmente definibile. Per quanto sappiamo la Sharp fornisce una curva di questa correlazione, ma non una libreria e nemmeno una funzione che possa approssimare questa curva. Le alternative a questo punto sono due e noi seguiremo la seconda.

La prima è quella di usare un'apposita libreria che funziona solo con le schede Arduino Uno; l'altra è quella di ricavare la funzione di correlazione tra tensione d'uscita e distanza da dati sperimentali specifici per il nostro particolare esemplare di sensore, ricavando per interpolazione una legge personalizzata che potremo usare anche con schede alternative ad Arduino Uno. Una delle curve proposte in rete è del tipo $d = aV^{-b}$, dove d è la distanza misurata, V è la tensione in uscita dal sensore e a e b sono due parametri da ricavare sperimentalmente. Se facciamo il logaritmo di entrambi i membri possiamo trasformarla in una relazione lineare e procedere ad interpolazione lineare: $\ln d = \ln a - b \ln V$. Questa proposta non si è rilevata adeguata a sfruttare tutte le potenzialità del sensore. Per cui proporrò anche una funzione interpolatrice basata su spline, di facile utilizzo e forse persino più immediata della legge di potenza.

Lettura dei dati

Possiamo domandarci se usare l'ingresso analogico digitale di Arduino Uno consenta di sfruttare al meglio il sensore. Ricordiamoci che tale ADC è a 10 bit e permette di discriminare tra $2^{10} = 1024$ livelli di una tensione massima di 5 V, quindi circa 5 mV per bit. Alla distanza di circa 6 cm (la portata minima del sensore) la tensione di riferimento è circa 3,3 V; se la tensione varia di 5 mV la distanza osservata varia di soli 0,2 mm. Invece, alla distanza di circa 70 cm (quasi la portata massima del sensore) la tensione di riferimento è circa 0,5 V; se la tensione varia di 5 mV la distanza osservata varia di ben 7 mm. Se quindi usiamo il sensore per piccole distanze la sensibilità di questo ADC sembra del tutto adeguata. Noi proponiamo qui di seguito anche l'uso di un ADC alternativo a 16 bit, l'ADS 1115.

Condizioni operative

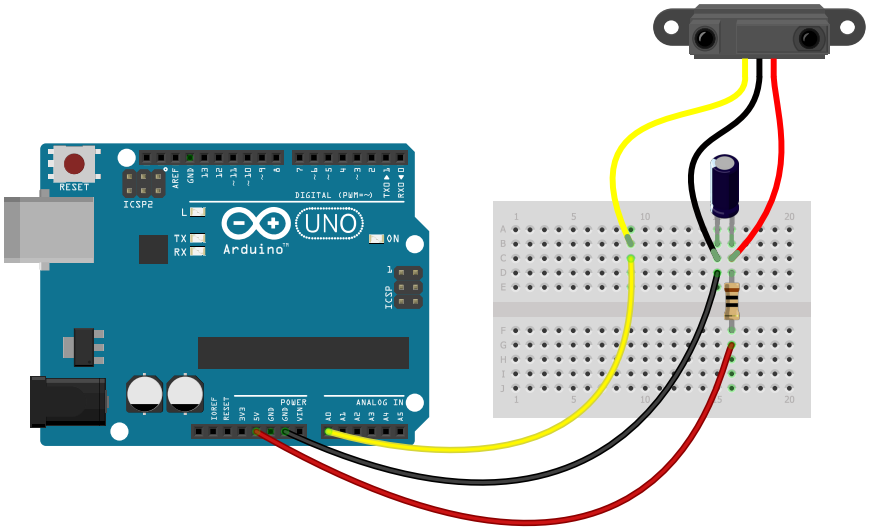
Il sensore emette un fascio luminoso nell'infrarosso ed è sensibile alla luce ambientale. Al contrario del sensore precedente l'uscita dati è pressoché continua, consentendoci di meglio monitorare il movimento di oggetti, se pur con minore accuratezza. Le specifiche del sensore parlano propriamente di un tempo di circa 50 ms tra una misura e la successiva. Più problematica è la dimensione e la riflettività dell'oggetto da individuare: sperimentalmente si trova che l'oggetto deve essere preferibilmente bianco e largo almeno due centimetri. Inoltre è estremamente importante, come illustreremo in seguito, la qualità dell'alimentazione del sensore.

5.3.2 Strumenti necessari

Il nostro apparato di misurazione è costituito dai seguenti componenti:

1. una scheda Arduino (uno)
2. sensore di distanza Sharp GP2Y0A21YK0F
3. un condensatore da 100 µF
4. una resistenza da 10 Ω
5. cavi di collegamento
6. breadboard
7. terza mano
8. (ADC Texas Instruments ADS1115) solo per un uso più avanzato
9. Alimentazione esterna stabilizzata a 5 V (pila da 9 V + convertitore DC DC con uscita a 5 V)

5.3.3 Schema di montaggio

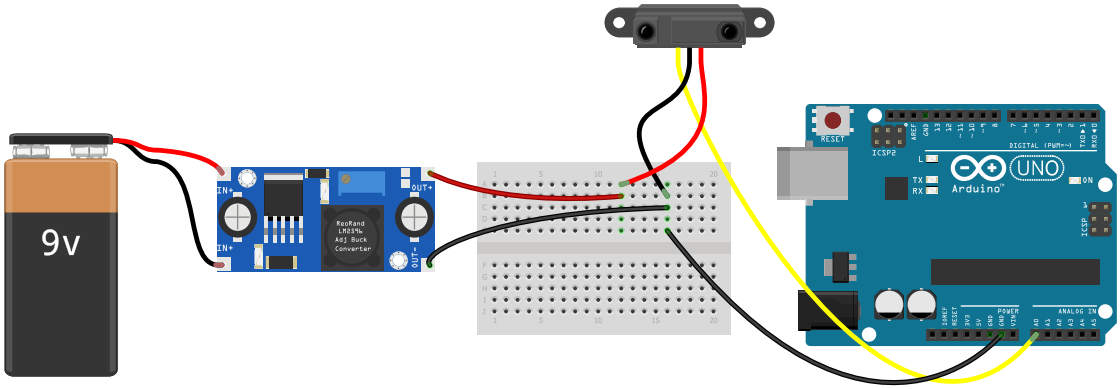


Il sensore è fornito di un connettore con tre cavi.

- Il pin VCC (sulla destra) va collegato al pin 5 V di Arduino.
- Il pin GND (centrale) va collegato al pin GND di Arduino.
- Il pin A0 (sulla sinistra) va collegato al pin A0 di Arduino.

I nomi dei pin del sensore non sono indicati da nessuna parte. Tuttavia il connettore è dotato di fili dal colore chiaramente riconoscibile. La terza mano ci serve per tenere in posizione il sensore. Il condensatore, che va collegato tra la linea del VCC e del GND, non è formalmente necessario, ma è fortemente consigliato per stabilizzare la tensione in ingresso a causa dei picchi di assorbimento di corrente da parte del sensore e in un'ultima analisi per rendere più stabile i valori in uscita. Le specifiche del sensore parlano di un assorbimento tipico di 30 mA e uno di picco di 40 mA; siamo al limite di quanto possibile con Arduino Uno per il quale, sul singolo pin, non si possono superare 40 mA, ma si consiglia di non superare i 20 mA. Invece la corrente massima che può uscire da Arduino Uno è complessivamente di 200 mA. Dalle prove successive emergerà la problematicità dell'alimentazione di questo sensore.

Questo è il motivo per cui siamo ricorsi ad un'alimentazione esterna come di seguito rappresentato.



In questo schema alternativo il sensore è alimentato con un regolatore di tensione esterno regolato su 5 V, a sua volta alimentato ad esempio con una batteria da 9 V. Il regolatore indicato in figura non è quello da noi effettivamente usato, ma non abbiamo trovato una rappresentazione migliore di questa. Questi dispositivi hanno tipicamente due connettori di ingresso e due di uscita, come qui indicato. La massa dell'alimentatore, del sensore e di Arduino devono essere in comune. Il polo positivo dell'alimentatore è posto in contatto solo con il sensore.

I codici che seguono sono:

1. Un codice per l'utilizzo con una legge interpolata specifica per il nostro sensore e l'ADC di Arduino.
2. Un codice per l'utilizzo con il convertitore analogico digitale a 16 bit ADS 1115.
3. Un codice per la sua taratura, ricavando una funzione interpolatrice con il convertitore analogico digitale a 16 bit ADS 1115, funzione che possiamo utilizzare anche direttamente con Arduino Uno o Due.

5.3.4 Codice per l'utilizzo 1

```
const int sensorPin = A0;

#include "InterpolationLib.h"
// alimentazione di arduino
const int numValues = 9;
double xValues[9] = {0.679,0.744,0.821,0.918,1.076,1.286,1.646,2.340,2.927};
double yValues[9] = {  450,  400,  350,  300,  250,  200,  150,  100,   75};
// alimentazione esterna
//const int numValues = 9;
//double xValues[9] = {0.652,0.712,0.796,0.911,1.069,1.299,1.641,2.335,2.919};
//double yValues[9] = {  450,  400,  350,  300,  250,  200,  150,  100,   75};

void setup(void)
{
  Serial.begin(115200);
  pinMode(A0, INPUT);
  // analogReadResolution(12); // Arduino Due
}
void loop(void)
{
  int mm, adc0;
  float volt;
  adc0 = analogRead(sensorPin);
  volt = adc0*5/1024.0; // Arduino Uno
  // volt = adc0*3.3/4096.0; // Arduino Due
  mm = Interpolation::SmoothStep(xValues, yValues, numValues, volt);
  // mm = int(274.57 * pow(volt, -1.2083));
  Serial.println(mm);
  delay(100);
}
```

- La riga 1 definisce il pin in ingresso dal sensore.
- La riga 3 richiama la libreria relativa alla funzione interpolatrice spline.
- La riga 5 definisce il numero di punti su cui costruire la spline.
- Le righe 6 e 7 riportano i valori interpolanti. Quelli relativi alla x devono essere in ordine crescente.
- Le righe 8 - 11 sono l'alternativa alle precedenti, riferiti all'uso di una alimentazione esterna.
- La riga 15 seleziona la velocità di comunicazione tra Arduino e l'esterno.
- La riga 17 è opzionale per Arduino Due per sfruttare tutta la risoluzione dell'ADC.
- Le righe 21 e 22 definiscono tre variabili: il valore intero letto dal pin analogico di ingresso, la tensione corrispondente e la distanza.
- La riga 23 legge il valore nell'ingresso analogico.
- La riga 24 trasforma quel valore in volt, considerando che la tensione di riferimento di Arduino Uno è 5,0 V e il pin legge valori su una scala di 1024 livelli.
- La riga 25 è l'alternativa alla riga 24 per Arduino Due che ha come tensione di riferimento 3,3 V su 4096 livelli.
- La riga 26 trasforma la tensione letta in una distanza in millimetri secondo il valore interpolato con una spline, sulla base dei valori indicati all'inizio del codice.

- La riga 27 trasforma la tensione letta in una distanza in millimetri secondo una legge di potenza specifica per il nostro sensore: è la scelta alternativa che non consigliamo.
- La riga 28 stampa il valore letto.
- La riga 29 attende 100 ms: otteniamo circa 10 letture al secondo.

5.3.5 Codice per l'utilizzo 2

Il seguente codice prevede l'uso dell'ADC ADS 1115. Il collegamento viene descritto nel paragrafo successivo, relativo alla taratura del sensore.

```
#include "Wire.h"
#include <Adafruit_ADS1X15.h>
Adafruit_ADS1115 ads;

#include "InterpolationLib.h"
// alimentazione di arduino
const int numValues = 9;
double xValues[9] = {0.679,0.744,0.821,0.918,1.076,1.286,1.646,2.340,2.927};
double yValues[9] = {  450,  400,  350,  300,  250,  200,  150,  100,   75};
// alimentazione esterna
//const int numValues = 9;
//double xValues[9] = {0.652,0.712,0.796,0.911,1.069,1.299,1.641,2.335,2.919};
//double yValues[9] = {  450,  400,  350,  300,  250,  200,  150,  100,   75};

void setup(void)
{
  Serial.begin(115200);
  ads.setGain(GAIN_ONE);        // 1x gain   +/-4.096V
  ads.setDataRate(64);
  ads.begin();
}

void loop(void)
{
  int adc0, mm;
  float volt;
  adc0 = ads.readADC_SingleEnded(0);
  volt = adc0/8000.0;
  mm = Interpolation::SmoothStep(xValues, yValues, numValues, volt);
  // mm = int(274.57 * pow(volt, -1.2083));
  Serial.println(mm);
}
```

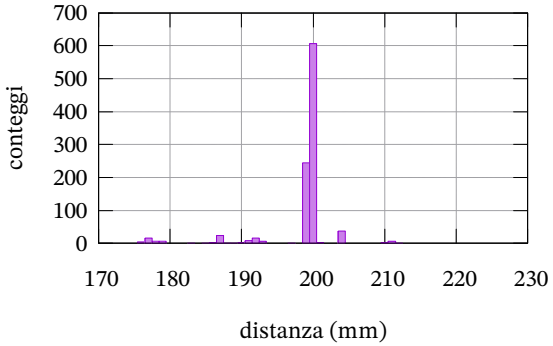
- Le prime due righe richiamano la libreria per la comunicazione dati con l'ADC e quella specifica per il suo funzionamento.
- La riga 3 fa un cambio di variabile per una scrittura più semplice di ciò che segue.
- La riga 18 definisce il livello di guadagno dell'ADC: viene scelto quello unitario che prevede una tensione in ingresso compresa tra più o meno 4,096 V, adeguata al nostro sensore di distanza che fornisce una tensione massima in uscita di circa 3,3 V.
- La riga 19 selezione il data rate: il valore indicato consente 30 letture al secondo.
- Le righe 24 e 25 definiscono tre variabili: il valore intero letto dal pin analogico di ingresso, la tensione e la distanza.
- La riga 26 legge il valore nell'ingresso analogico.
- La riga 27 trasforma quel valore in volt considerando che la tensione di riferimento dell'ADC è 4,096 V ($2^{10} = 4096$) e il pin legge valori su una scala di $2^{15} = 32768$ livelli.

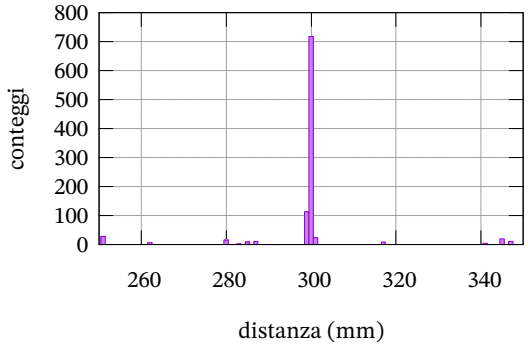
5.3.6 Studio delle prestazioni

Comportamento senza condensatore e resistenza

In questa modalità le misure sono concentrate sul valore atteso. Le misure che presento sono state fatte con un computer desktop. Lo stesso identico apparecchio collegato ad un portatile mi ha dato misure sovrastimate anche di due centimetri per le distanze maggiori, segno evidente della sensibilità del sensore alla variazione anche minima delle condizioni di lavoro.

Mentre la deviazione standard appare del tutto ragionevole, ciò che è del tutto preoccupante è la dispersione dei dati: essa è molto elevata, con fluttuazioni causali fino a 5 cm per la distanza maggiore. A meno di non effettuare una qualche media sulle letture i dati sono difficilmente utilizzabili.

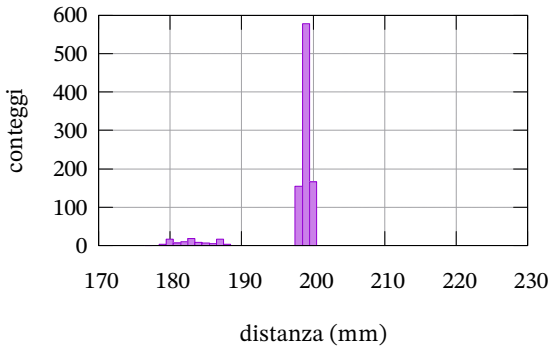


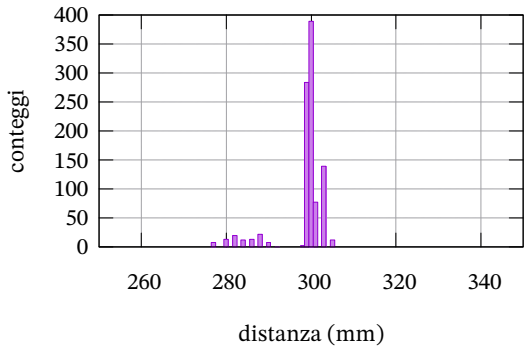


distanza target (mm)	distanza media misurata (mm)	σ (mm)
200	199	5
300	300	14

Comportamento con condensatore e resistenza

Ho provato ad utilizzare anche il solo condensatore con vantaggi limitati e non riporto i risultati ottenuti. In questa modalità le misure sono concentrate sul valore atteso come nel caso precedente, ma la dispersione dei dati è decisamente più ristretta rispetto alla precedente modalità, mentre rimane un fastidioso secondo picco inferiore che si manifesta con periodiche letture sottostimate.

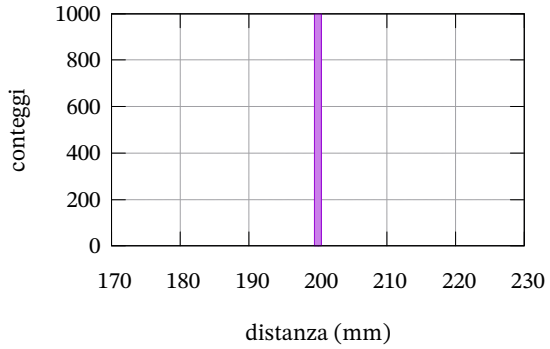


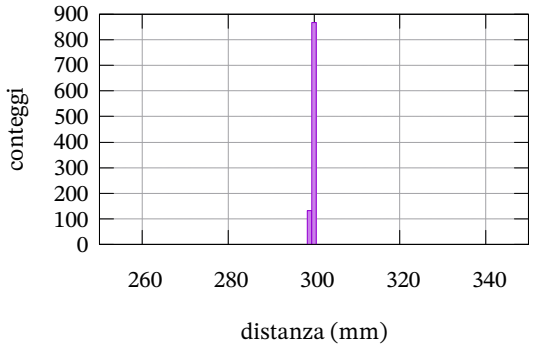


distanza target (mm)	distanza media misurata (mm)	σ (mm)
200	198	5
300	299	5

Comportamento con condensatore e resistenza tramite ADS 1115

Il comportamento è quasi inaspettatamente buono. La dispersione è minima e si riduce a meno di un millimetro. L'accuratezza è legata in maniera più significativa alla funzione interpolatrice.





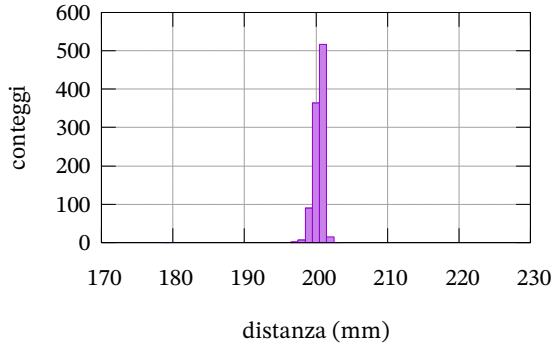
distanza target (mm)	distanza media misurata (mm)	σ (mm)
200	200,00	0,01
300	299,99	0,03

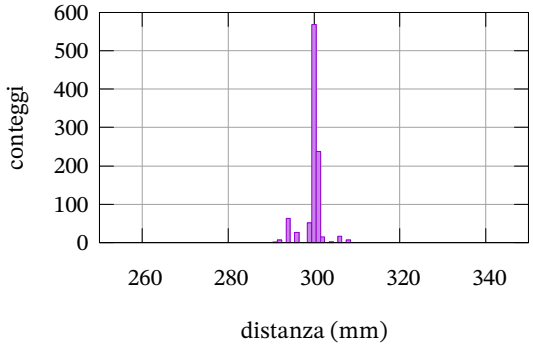
5.3.7 Studio delle prestazioni - il problema dell'alimentazione

Comportamento con Arduino Uno

Come emerso nelle prove precedenti per ottenere un buon comportamento è essenziale una corretta alimentazione del sensore. Provando con diverse combinazioni di condensatori e resistenze alla fine mi sono chiesto se non fosse il caso di cambiare completamente strategia. Analizzando la tensione di alimentazione ai capi del sensore in tutte le modalità prima illustrate si nota una sistematica fluttuazione dell'ordine dei centesimi di volt o superiore e un abbassamento della tensione di riferimento di Arduino di un decimo di volt. Queste fluttuazioni influenzano anche l'ADC di Arduino perché la tensione di riferimento per esso è essa stessa quella di alimentazione a 5 V nominali. Allora è possibile superare il problema fornendo al sistema una alimentazione esterna in grado di erogare anche valori relativamente elevati di corrente? Ho trovato una risposta a questa domanda alimentando il solo sensore con una alimentazione esterna come illustrata nel paragrafo [D].

I dati ottenuti sono stati molto buoni, sia per quanto riguarda i valori di deviazione standard, che per il valor medio e la dispersione effettiva di circa 1 cm per la distanza maggiore.

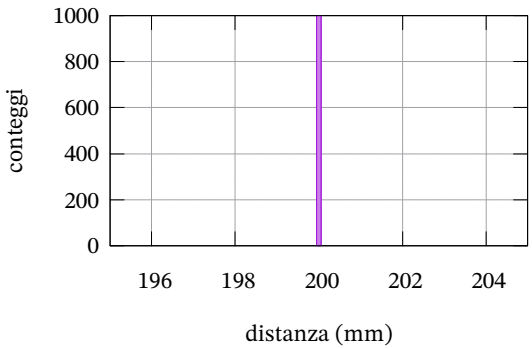


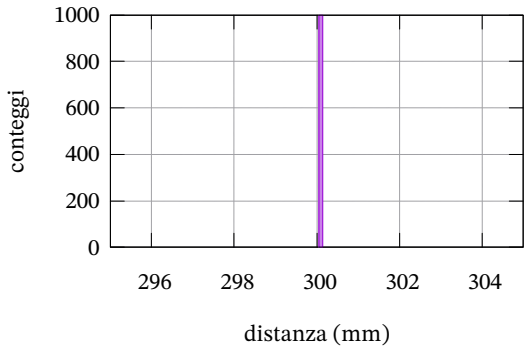


distanza target (mm)	distanza media misurata (mm)	σ (mm)
200	201	1
300	300	2

Comportamento con condensatore e resistenza tramite ADS 1115

Il comportamento è quasi perfetto, con sporadiche fluttuazioni, poco visibili nei grafici successivi, minori di un decimo di millimetro.





distanza target (mm)	distanza media misurata (mm)	σ (mm)
200	200,04	0,01
300	300,17	0,01

Conclusioni

1. Il sensore usato senza condensatore e resistenza è secondo me inusabile: le misure variano casualmente in un intervallo troppo ampio e neanche facendo la media tra molte misure si ottengono valori sufficientemente concentrati.
2. Il sensore dà le migliori prestazioni nella versione con il condensatore e la resistenza, se usato senza alimentazione esterna.
3. Meglio ancora si ottiene con alimentazione esterna.
4. L'accuratezza delle misure non supera il mezzo centimetro nelle condizioni reali di utilizzo per la singola misura, se non per le distanze minori, se usato con l'ADC di Arduino Uno.
5. I risultati con la scheda ADS 1115 sono ottimi, con un'accuratezza di un millimetro, segno evidente che il sensore ha delle buone potenzialità, ma c'è assoluta necessità di una interfaccia ben più sofisticata del solo Arduino Uno.
6. Anche se non emerge in queste prove è importante usare, per le misure più accurate, una buona funzione interpolatrice e non una semplice legge di potenza.
7. I dati ottenuti riguardano misurazioni in rapida successione, ma in condizioni statiche. Per misure in condizioni dinamiche rimando ad esperienze successive.

5.4 Taratura di GP2Y0A21YK0F

5.4.1 Introduzione

Come già detto nell'introduzione precedente il sensore non è ufficialmente fornito di una libreria dalla Sharp e quelle disponibili in rete hanno una utilità limitata, in particolare per sfruttare appieno le potenzialità del sensore per la misura di distanze più brevi, là dove il sensore è in grado di apprezzare anche i millimetri, mentre le librerie che ho individuato riportano soltanto i centimetri. Quello che intendo mostrare qui di seguito è come ricavare una funzione personalizzata per le proprie esigenze. Il mio personale interesse era avere una funzione ottimizzata per distanze tra 10 cm e 40 cm sfruttando un convertitore analogico digitale più preciso di quello fornito con Arduino Uno.

Quello che è stato fatto con il codice e lo schema di montaggio successivo è misurare il valor medio e la deviazione standard della tensione in uscita dal sensore per distanze crescenti a partire da 7,5 cm con passi di 5 cm fino a 45 cm. I valori ottenuti sono stati utilizzati per ottenere (inizialmente) una funzione personalizzata per interpolazione lineare su una *legge di potenza*. Come emerge dai dati sperimentali successivi la legge di potenza non fornisce una descrizione adeguata dell'andamento della tensione in funzione della distanza. Per questo motivo sono poi passato a una funzione interpolatrice più sofisticata come una *smooth spline*. Le spline hanno la caratteristica di interpolare perfettamente i dati di riferimento e di fornire una funzione sufficientemente "morbida" tra i punti interpolanti. La variante di queste funzioni definita smooth ha la caratteristica di attenuare gli eventuali artefatti che possono presentarsi per punti interpolanti troppo variabili. Per fortuna è disponibile per Arduino una facile libreria chiamata "InterpolationLib" che permette di trovare molto facilmente delle spline. Nel codice seguente implemento entrambe le curve, la curva di potenza e la spline.

La libreria per utilizzare le spline è scaricabile all'indirizzo `https://www.arduino.cc/reference/en/libraries/interpolationlib/` in formato zip. Questo file va caricato nell'IDE di Arduino. Dalla voce del menù principale "Sketch" si passa a "#include libreria" e infine "Aggiungi libreria da file .zip" con la quale si apre una finestra di dialogo dalla quale possiamo specificare il file precedentemente scaricato.

Per determinare la tensione il codice qui illustrato prevede 200 letture successive. Il valor medio può essere ottenuto così:

$$\overline{V} = \frac{\sum_{i=1}^{n} V_i}{n} \tag{5.3}$$

Invece la deviazione standard può essere valutata con:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (\overline{V} - V_i)^2}{n}} \tag{5.4}$$

Per utilizzare queste relazioni si dovrebbero memorizzare i valori V_i. Possiamo evitare tutto ciò accumulando la somma parziale delle tensioni V_i e delle tensioni al quadrato ${V_i}^2$ e usando la seguente relazione per valutare la deviazione standard:

$$\sigma = \frac{1}{n} \sqrt{n \sum_{i=1}^{n} (V_i)^2 - \left(\sum_{i=1}^{n} V_i \right)^2} \tag{5.5}$$

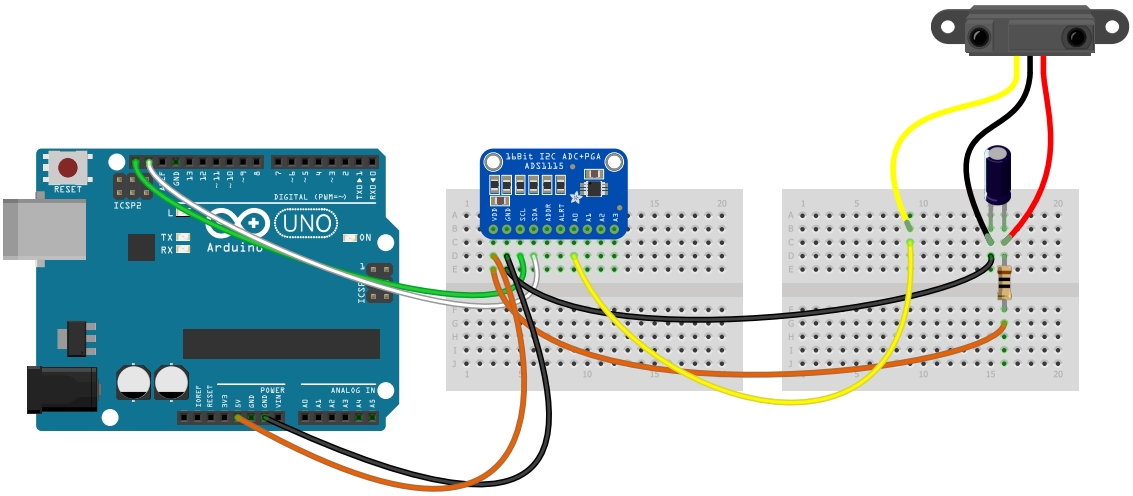
Le misurazioni sono state effettuate posizionando il sensore parallelamente ad un reggilibri bianco e verticale: è importante che sensore e corpo posto dinnanzi ad esso siano i più paralleli possibili.

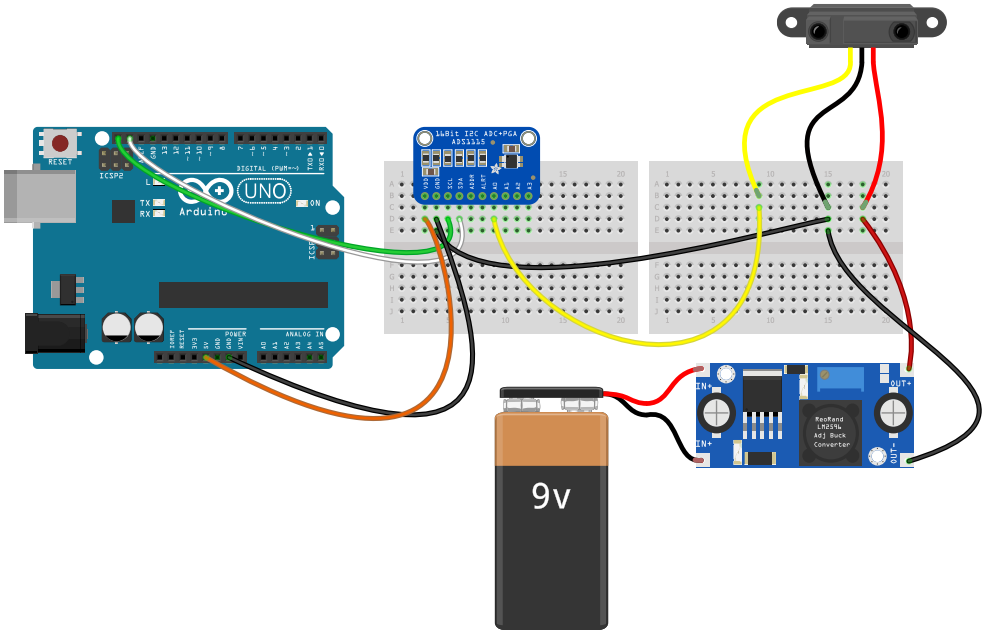
Con l'ADC ADS 1115 abbiamo 15 bit a disposizione per la lettura. La deviazione standard osservata è stata di circa 0,002 V. Usando Arduino Due, con in suoi 12 bit, la deviazione standard è stata circa 0,008 V e con Arduino Uno circa 0,020 V. Tuttavia l'ADS 1115 ha una velocità di lettura relativamente limitata che non può superare qualche centinaio di letture al secondo, mentre Arduino Uno può arrivare a circa 5000 letture al secondo.

É importare osservare che le funzioni interpolatrici qui trovate si adattano comunque a tutte le tipologie di schede.

5.4.2 Schema di montaggio

Qui di seguito viene indicato lo schema di montaggio senza e con alimentazione esterna.





5.4.3 Codice per l'utilizzo

```
#include "Wire.h"
#include <Adafruit_ADS1X15.h>
Adafruit_ADS1115 ads;
int i = 0;
float volt;
float v_sum = 0;
float v_sum_q = 0;
float sigma = 0.0;
int N = 50;
int mm1, mm2, adc0;
#include "InterpolationLib.h"
// alimentazione di arduino
const int numValues = 9;
double xValues[9] = {0.679,0.744,0.821,0.918,1.076,1.286,1.646,2.340,2.927};
double yValues[9] = {  450,  400,  350,  300,  250,  200,  150,  100,   75};
// alimentazione esterna
//const int numValues = 9;
//double xValues[9] = {0.652,0.712,0.796,0.911,1.069,1.299,1.641,2.335,2.919};
//double yValues[9] = {  450,  400,  350,  300,  250,  200,  150,  100,   75};

void setup(void)
{
  Serial.begin(115200);
  ads.setGain(GAIN_ONE);        // 1x gain   +/-4.096V
  ads.setDataRate(8);
  ads.begin();
}
void loop(void)
{
  volt = 0;
  v_sum = 0;
  v_sum_q = 0;
  for(i=1;i<=N;i++){
    adc0 = ads.readADC_SingleEnded(0);
    volt = adc0/8000.0;
    delay(50);
    v_sum = v_sum + volt;
    v_sum_q = v_sum_q + volt*volt;
  }
  volt = v_sum/N;
  sigma = sqrt(N*v_sum_q - v_sum*v_sum )/N;
  mm1 = Interpolation::SmoothStep(xValues, yValues, numValues, volt);
  mm2 = int(274.57 * pow(volt, -1.2083));
  Serial.println("␣");
  Serial.print(volt,3);
  Serial.print("␣±␣");
  Serial.print(sigma,6);
  Serial.print("␣");
  Serial.println(mm1);
}
```

5.4.4 Dati sperimentali

Nella seguente tabella riportiamo:

1. La tensione in volt rilevata dall'ADC: la deviazione standard è dell'ordine di 0,002 V.
2. La distanza in millimetri: l'accuratezza è di un millimetro.
3. La distanza interpolata secondo la legge di potenza. La distanza interpolata con la spline non è riportata perché per definizione è identica a quella misurata nei punti presi a riferimento.

	V (V)	distanza reale (mm)	distanza interpolata (mm) legge di potenza
1	2,927	75	75
2	2,340	100	98
3	1,646	150	151
4	1,286	200	203
5	1,076	250	252
6	0,918	300	306
7	0,821	350	350
8	0,744	400	395
9	0,679	450	441

Facciamo il logaritmo naturale dei dati qui riportati, ricordando che la grandezza indipendente è la distanza. Se la tensione segue una legge di potenza allora possiamo considerare le due grandezze in relazione lineare e secondo quanto indicato nel capitolo [A] otteniamo la retta di equazione:

$$\ln(d) = -1{,}2139 \cdot \ln(V) + 5{,}6198 \tag{5.6}$$

Con un coefficiente di correlazione $r = 0{,}9995$.
Associata a questa relazione lineare tra logaritmi possiamo scrivere la nostra legge di potenza.

$$\boxed{d = 275{,}83 \cdot V^{-1{,}2139}\ \text{mm}} \tag{5.7}$$

Osserviamo che le distanze interpolate secondo la legge indicata sono in alcuni casi eccedenti e in altri casi sottostimate, segno che l'andamento reale della tensione è più complicato della legge interpolatrice proposta: questo è il motivo per il quale è stata introdotta la spline.

Alimentazione esterna

Qui di seguito riporto invece i dati sperimentali con l'alimentazione esterna. Il sensore è decisamente sensibile alla tensione di alimentazione e il solo cambiare computer con cui alimentare Arduino può cambiare i dati. Mentre dalle prese USB arrivano solitamente poco meno di 5 V invece con l'alimentazione esterna erano disponili circa 5,02 V.

	V (V)	distanza reale (mm)	distanza interpolata (mm) legge di potenza
1	2,919	75	76
2	2,335	100	99
3	1,641	150	150
4	1,299	200	198
5	1,069	250	249
6	0,911	300	301
7	0,796	350	353
8	0,712	400	403
9	0,652	450	447

Facciamo il logaritmo naturale dei dati qui riportati e otteniamo la retta di equazione:

$$\ln(d) = -1{,}1822 \cdot \ln(V) + 5{,}5968 \tag{5.8}$$

Con un coefficiente di correlazione $r = 0{,}9998$.
Associata a questa relazione lineare tra logaritmi possiamo scrivere la nostra legge di potenza.

$$\boxed{d = 269{,}56 \cdot V^{-1{,}1822} \text{ mm}} \tag{5.9}$$

5.5 MB1003

5.5.1 Introduzione

Il sensore di distanza MB 1003, della serie HRLV-MaxSonar®-EZ della MaxBotix, è un sensore a sonar con una portata compresa tra 30 cm e 5 m, una risoluzione di 1 mm, una frequenza di aggiornamento di 10 hz e una accuratezza dell'ordine del 1% o migliore.

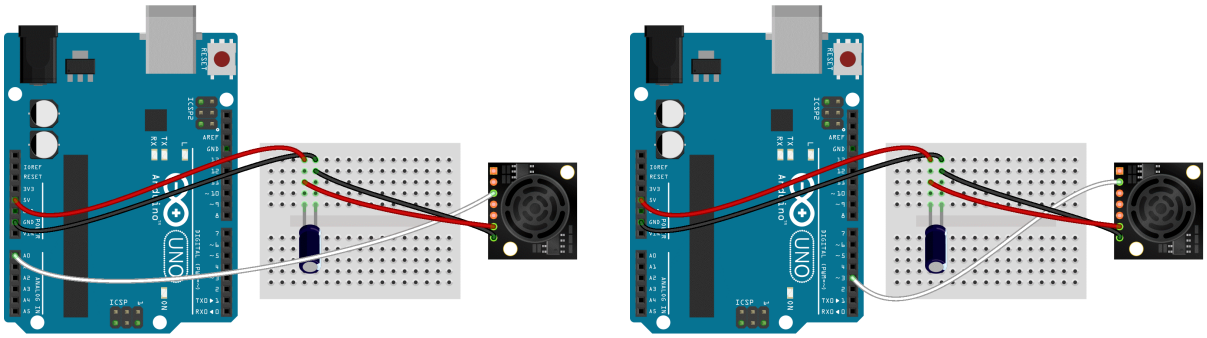
Il sensore fornisce la distanza direttamente e non ha bisogno di librerie per la sua gestione. Questo vuol dire anche che non possiamo conoscere direttamente il tempo di andata e ritorno del segnale del sonar.

Per l'utilizzo sono indicate due modalità di funzionamento:

- Una fa riferimento ad un segnale modulato in tensione e proporzionale alla distanza misurata. Il sensore produce una tensione in uscita il cui valore (se usiamo l'alimentazione a 5 V) è pari alla distanza in metri. La risoluzione disponibile con Arduino Uno R3 è solamente di 5 mm a causa della limitata risoluzione dell'ADC a 10 bit della scheda. Questo problema non si pone con Arduino Due o Arduino Uno R4, a patto di impostare la risoluzione dell'ADC al massimo valore possibile; altrimenti la modalità predefinita è anche lì a 10 bit.
- L'altra modalità acquisisce il valore della distanza tramite un impulso che viene acquisito da Arduino con la funzione pulseIn. Questa funzione restituisce la durata di un impulso in microsecondi. Questa durata è uguale alla distanza misurata in millimetri.

In entrambe le modalità, per quanto ci sia dato di sapere, non è possibile impostare la scheda in modo da fare una nuova lettura solo quando la scheda è pronta. Per cui è necessario, per lo meno in modalità analogica, attendere forzatamente almeno un decimo di secondo prima di una nuova lettura; altrimenti otterremo un apparente flusso dati ad alta frequenza, ma sostanzialmente fittizio. In modalità ad impulso questo problema si pone meno.

5.5.2 Schema di montaggio



(a) *modalità analogica*.

(b) *modalità pw*.

5.5.3 Codice per l'utilizzo

Modalità analogica

```
const int anPin = 0;
int anVolt1 mm;
void setup() {
Serial.begin(115200);
// analogReadResolution (12); // Arduino Due
// analogReadResolution (14); // Arduino UNO R4
}

void loop() {
anVolt = analogRead(anPin);
mm = (anVolt1)*5.0; // Arduino UNO R3
// mm = (anVolt1)*5.0/4.0; // Arduino Due
// mm = (anVolt1)*5.0/16.0; // Arduino UNO R4
Serial.println(mm);
delay(100);
}
```

- La riga 1 dice che il pin della scheda con l'uscita analogica va collegato al pin A0 di Arduino.
- Definiamo la distanza in mm con variabile intera.
- La riga 4 definisce la velocità di comunicazione con Arduino.
- Alle righe 5 e 6, in cui togliere il commento iniziale secondo necessità, definiscono esplicitamente il numero di bit da impostare. Con Arduino Uno R3 non sono necessarie.
- Alla riga 10 acquisiamo il valore di tensione dal pin A0.
- Che poi trasformiamo in un valore in volt.
- Le righe 12 e 13 sono l'alternativa per le altre due schede.
- Alla riga 14 stampiamo il risultato.
- Alla riga 15 aspettiamo 100 millisecondi.

Modalità pulseIn

```
const int pwPin1 = 3;
long mm;
void setup(){
Serial.begin(115200);
pinMode(pwPin1, INPUT);
}

void loop() {
mm = pulseIn(pwPin1, HIGH);
Serial.println(mm);
delay(100);
}
```

- La riga 1 dice che il pin della scheda con l'uscita digitale va collegato al pin D3 di Arduino.
- Definiamo la distanza in mm con variabile intera.
- La riga 4 definisce la velocità di comunicazione con Arduino.
- Alla riga 5 impostiamo il pin in modalità di input.

- Alla riga 9 acquisiamo la durata dell'impulso.
- Alla riga 10 stampiamo il risultato.
- Alla riga 11 aspettiamo 100 millisecondi.

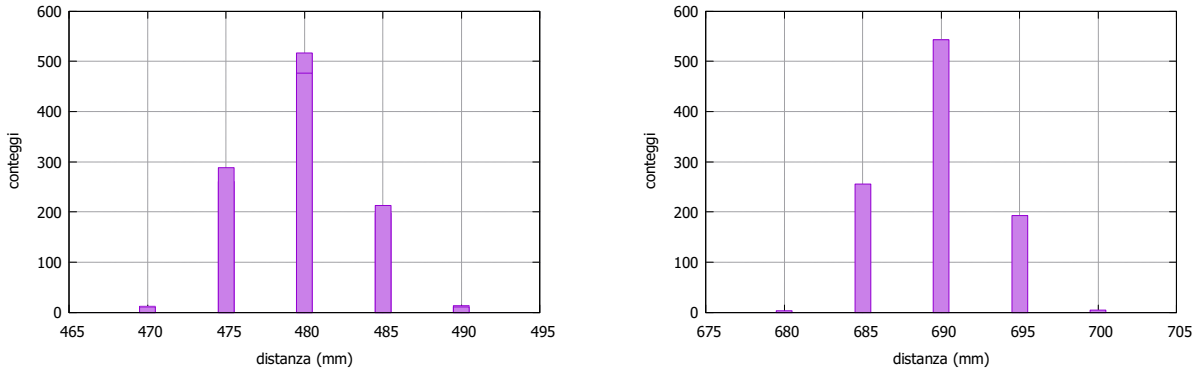
5.5.4 Studio delle prestazioni

La Maxbotix consiglia di usare il sensore con un circuito con condensatore e resistenza per migliorare la stabilità delle misure. Nelle prove da me fatte la presenza della resistenza nel circuito non migliora la stabilità rispetto alla presenza del solo condensatore a fronte di una minore accuratezza di un paio di centimetri: per questa ragione la resistenza non è stata inserita nei circuiti prima proposti.

Qui di seguito propongo alcune misure fatte con Arduino Uno R3 e R4. Con la prima scheda notiamo la limitata risoluzione con le misure in analogico, ma la dispersione è comparabile a quella dell'altra scheda. *Migliori entrambe nella versione ad impulsi*, ma Arduino R4 da risultati più accurati e con meno dispersione. In particolare la versione R3 mostra sistematicamente due picchi, che non sono riuscito a motivare ed eliminare.

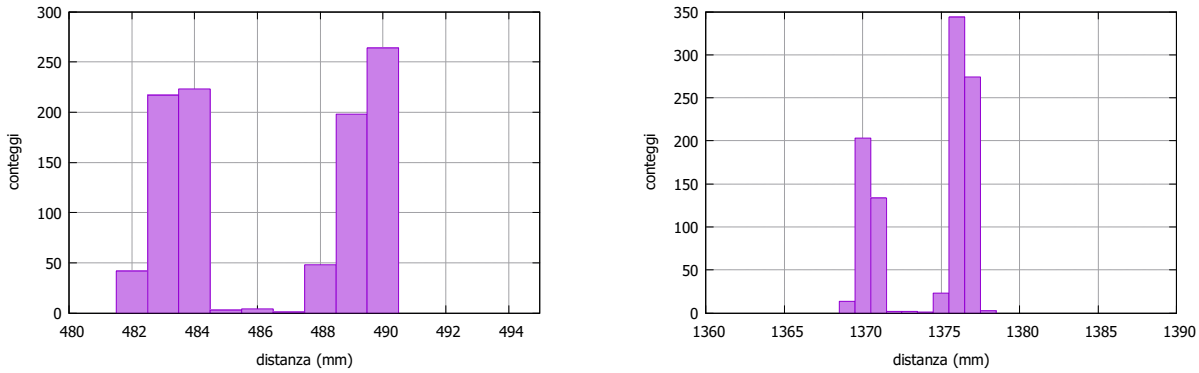
Arduino Uno R3

analogico



distanza target (mm)	distanza media misurata (mm)	σ (mm)
500	480	4
700	690	3

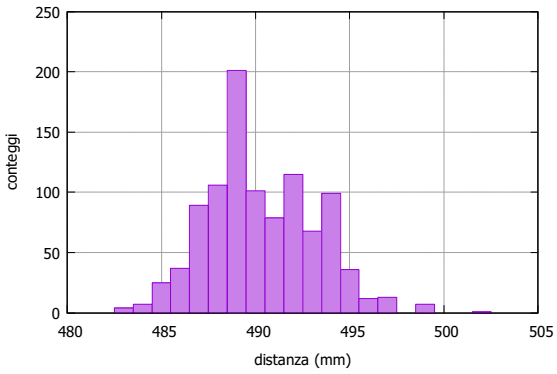
impulsi

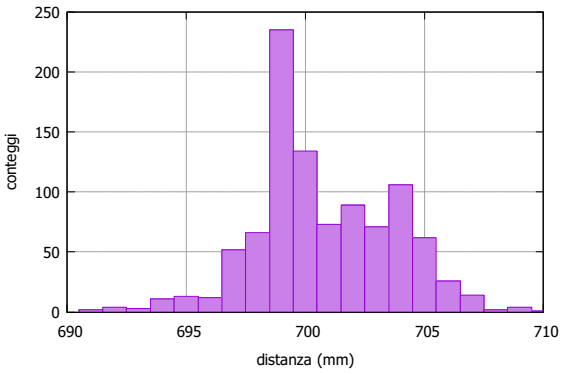


distanza target (mm)	distanza media misurata (mm)	σ (mm)
500	486	3
700	696	3
1000	987	1
1400	1374	3

Arduino Uno R4

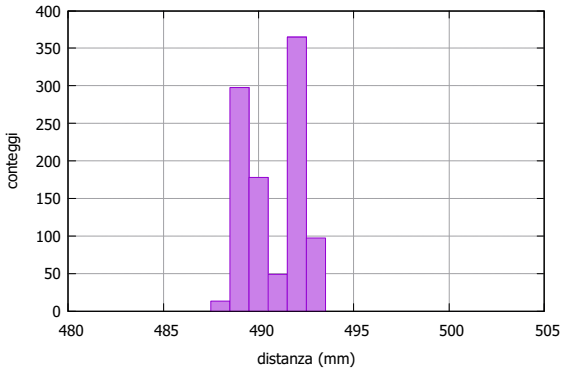
analogico

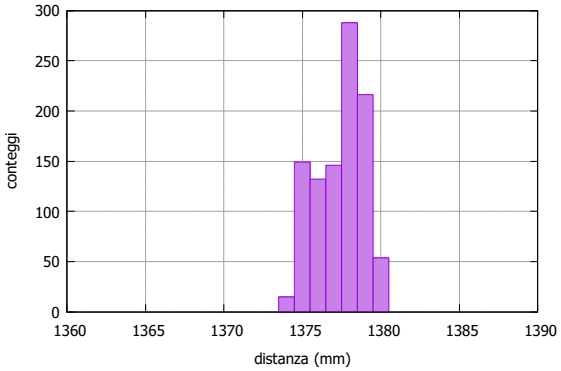




distanza target (mm)	distanza media misurata (mm)	σ (mm)
500	490	3
700	701	4

impulsi





distanza target (mm)	distanza media misurata (mm)	σ (mm)
500	491	1
700	699	1
1000	986	1
1400	1377	2

Conclusioni

La modalità ad impulsi sembra la migliore sia riguardo alla dispersione delle misure che alla risoluzione, soprattutto con Arduino Uno R3. Con entrambe le schede e modalità l'accuratezza non supera il centimetro e tende a peggiorare con le maggiori distanze. Il test è stato condotto sino a circa un metro e mezzo a causa degli spazi in cui è stato effettuato, sia perché quelle distanze erano le più interessanti in relazione a possibili esperimenti.

5.5 MB1003

6 Confronto tra sensori di distanza

In questo capitolo illustro una estesa comparativa tra i sensori di distanza illustrati nel capitolo precedente. I sensori verranno utilizzati per misurare una distanza fissa e poi per seguire il moto di un pendolo. In questa sezione non descrivo la fisica del pendolo, ma mi limito a utilizzare i sensori per seguire il moto del corpo oscillante. Questa comparativa nasce proprio dall'esperienza (per me più volte fallimentare) di utilizzare uno di quei sensori per seguire oggetti in movimento, trovando che molte delle caratteristiche positive descritte in condizioni statiche si perdevano totalmente.

I problemi fondamentali che si incontrano sono la costanza della misura e la capacità di individuare un corpo non troppo esteso e in movimento a qualche decina di centimetri di distanza. Per studiare il primo aspetto posizioneremo i sensori a tre distanze fisse da un oggetto esteso, piatto e chiaro, osservando la distribuzione delle misure ottenute. Osserveremo in particolare la varianza delle misure e la dispersione tra valor medio e valori più grandi e più piccoli ottenuti. Per studiare il secondo aspetto cercheremo di riprendere il moto di un pendolo realizzato con una massa di 2 cm di lato e poi con una di 4 cm.

I sensori che useremo sono:

1. HC-SR04
2. US-015
3. VL53L1X
4. GP2Y0A21YK0F
5. GP2Y0E03

6.1 Schema di montaggio e codice per i vari sensori

7 Misure di pressione

7.1 Introduzione

La pressione è una quantità scalare definita come il rapporto tra il modulo della forza che agisce normalmente ad una superficie e l'area della superficie stessa:

$$p = \frac{F^{\perp}}{A} \tag{7.1}$$

La pressione dell'atmosfera al livello del mare in condizioni standard (0 °C) ha il valore $p_0 = 101325$ Pa. Questa pressione segue in prima approssimazione la legge di Stevino:

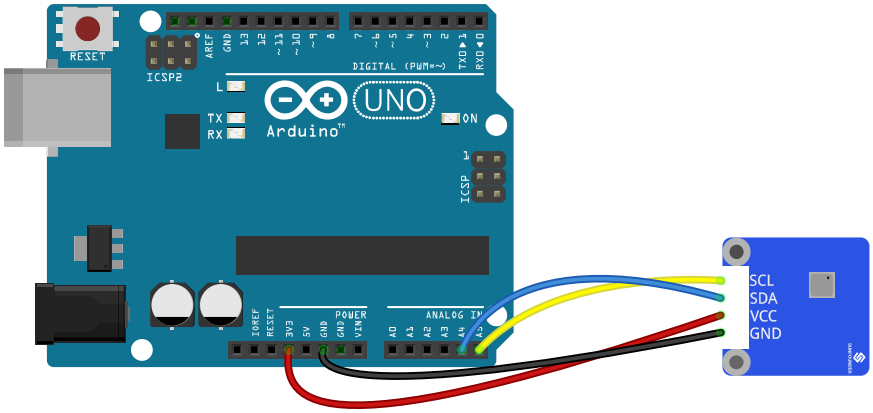
$$p(h) = dgh \tag{7.2}$$

dove d è la densità dell'ari, g l'accelerazione di gravità e h l'altezza dell'ideale colonna d'aria di densità costante posta sopra di noi. La pressione, secondo questo modello, diminuisce linearmente con il variare dell'altitudine. Questa approssimazione è del tutto adeguata se ci si trova a basse altitudini.

7.2 BMP 180

Come misuratore di pressione atmosferica è disponibile il sensore BMP180 della Bosch. Questo sensore può misurare una pressione compresa tra 300 e 1100 hPa: queste sono le pressioni che si possono misurare nel luogo con altitudine più alta e più bassa presenti sulla Terra.L'accuratezza relativa ottenibile è 0,12 hPa, ma se settato opportunamente l'accuratezza può arrivare a 0,02 hPa (secondo quanto riportato nelle specifiche del microcircuito integrato) che corrisponde ad una variazione di altitudine di soli 17 cm. In pratica il rumore di fondo dello strumento è decisamente peggiore di questi valori.

7.2.1 Schema di montaggio



1. Il pin denominato GND va messo a terra e collegato al pin GND su Arduino;
2. Il pin VCC va collegato all'alimentazione, preferibilmente a 3,3 V: lo colleghiamo al pin a 3,3 V di Arduino;
3. Il pin SCL è indicato come "I2C serial bus clock input": lo colleghiamo al pin A5;
4. Il pin SDA è indicato come "I2C serial bus data (or SPI input)" lo colleghiamo al pin A4.

La comunicazione tra sensore e Arduino è realizzata attraverso il protocollo di comunicazione I2C. Non abbiamo bisogno di conoscere le specifiche di questo protocollo: la libreria Wire si occupa di questi particolari. Inoltre la comunicazione con il sensore è mediata dalla libreria `SFE_BMP180`. Questa libreria non è solitamente compresa tra quelle di cui è dotato l'IDE di Arduino e va installa a parte.

7.2.2 Codice per l'utilizzo

```
#include <SFE_BMP180.h>
#include <Wire.h>
SFE_BMP180 pressure;
void setup()
{
  Serial.begin(9600);
 if (pressure.begin())
    Serial.println("BMP180 init success");
  else
```

```
10    {
        Serial.println("BMP180 init fail\n\n");
12      while(1); // Pause forever.
      }
14  }
    void loop()
16  {
      char status;
18    double T,P,p0,a;
      status = pressure.startTemperature();
20    if (status != 0)
      {
22      delay(status);
        status = pressure.getTemperature(T);
24      if (status != 0)
        {
26        Serial.print(T,2);
        status = pressure.startPressure(3);
28        if (status != 0)
          {
30          delay(status);
            status = pressure.getPressure(P,T);
32          if (status != 0)
            {
34            // Print out the measurement:
              Serial.print(" ");
36            Serial.print(P,3);
              Serial.println(" "); //
38          }
            else Serial.println("error retrieving pressure measurement\n");
40        }
          else Serial.println("error starting pressure measurement\n");
42      }
        else Serial.println("error retrieving temperature measurement\n");
44    }
      else Serial.println("error starting temperature measurement\n");
46    delay(1000);
    }
```

- Le righe 1 e 2 richiamano le librerie necessarie per utilizzare il protocollo I2C e per interfacciarsi più facilmente con il sensore.
- La riga 3 è un cambio di nome di variabile.
- La riga 6 seleziona la velocità di comunicazione tra Arduino e l'esterno.
- La riga 7 inizializza il sensore: se la funzione, tramite variabile status, restituisce un valore booleano vero allora si stampa una scritta di ok; altrimenti Arduino viene messo in uno stato di attesa infinita (riga 12).
- La riga 19 inizia ad inizializzare il sensore di temperatura: viene ottenuto un tempo di attesa o un messaggio di errore se il risultato è zero.
- La riga 23 legge la temperatura che viene assegnata alla variabile T.
- La riga 26 stampa la temperatura in celsius con due cifre decimali.
- La riga 27 inizializza il sensore di pressione: il parametro 3 indica che vengono raccolti 8

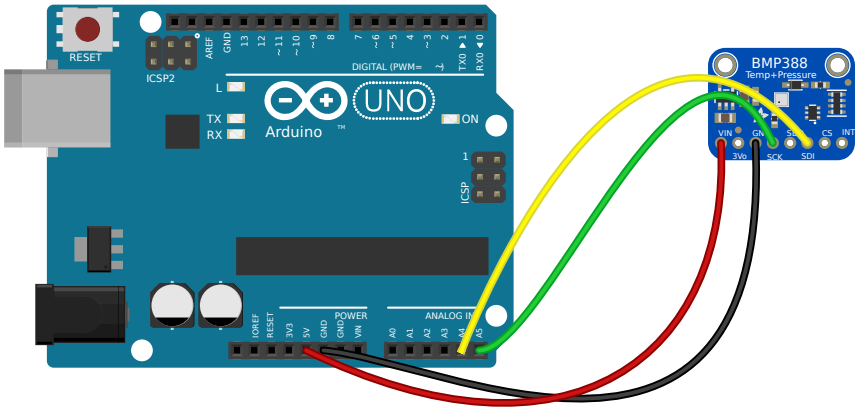
samples e viene fatta la media tra essi.

- La riga 31 fornisce la pressione P dopo aver passato la temperatura T precedentemente ottenuta.
- La riga 36 stampa la pressione in hPa.
- La riga 46 mette Arduino in pausa per un secondo.

7.3 BMP 388

Il sensore BMP180 è il primo che ho avuto modo di acquistare pochissimi anni fa, ma è stato sostituito da altri prodotti con migliori prestazioni. Tra questi, al momento in cui scrivo, c'è il BMP388, sempre della Bosch. La portata è rimasta la stessa; ciò che è stato migliorato è l'accuratezza (0,08 hPa), ma soprattutto il rumore di fondo, ora molto più basso, determinante per avere misure precise e ripetibili. Il sensore di temperatura integrato ha un'accuratezza di 0,5 °C. La risoluzione ottenibile in uscita, sia per la pressione che per la temperatura, è invece molto inferiore a questi valori, ma va valutata con attenzione.

7.3.1 Schema di montaggio



Il sensore è dotato di otto pin; ne usiamo quattro con una connessione I2C:

1. Il pin denominato GND va messo a terra e collegato al pin GND su Arduino;
2. Il pin VCC va collegato all'alimentazione, preferibilmente a 5,0 V: lo colleghiamo al pin a 5,0 V di Arduino;
3. Il pin SCK è indicato come "I2C serial bus clock input": lo colleghiamo al pin A5;
4. Il pin SDI è indicato come "I2C serial bus data (or SPI input)": lo colleghiamo al pin A4.

La comunicazione tra sensore e Arduino è realizzata attraverso il protocollo di comunicazione I2C. Non abbiamo bisogno di conoscere le specifiche di questo protocollo: la libreria Wire si occupa di questi particolari. Come libreria per gestire il sensore useremo quella della Adafruit, che è possibile scaricare all'indirizzo Adafruit_BMP3XX. Il file che segue è una versione leggermente semplificata dell'esempio fornito con la libreria.

7.3.2 Codice per l'utilizzo

```
/***************************************************************************
  This is a library for the BMP3XX temperature & pressure sensor

  Designed specifically to work with the Adafruit BMP388 Breakout
  ----> http://www.adafruit.com/products/3966

  These sensors use I2C or SPI to communicate, 2 or 4 pins are required
  to interface.

  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing products
  from Adafruit!

  Written by Limor Fried & Kevin Townsend for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 ***************************************************************************/
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BMP3XX.h"
Adafruit_BMP3XX bmp; // I2C

void setup() {
  Serial.begin(115200);
  while (!Serial);
  Serial.println("BMP388 test");

  if (!bmp.begin_I2C()) {
    Serial.println("Could not find a valid BMP3 sensor, check wiring!");
    while (1);
  }
  // Set up oversampling and filter initialization
  bmp.setTemperatureOversampling(BMP3_OVERSAMPLING_8X);
  bmp.setPressureOversampling(BMP3_OVERSAMPLING_16X);
  bmp.setIIRFilterCoeff(BMP3_IIR_FILTER_COEFF_3);
}
void loop() {
  if (! bmp.performReading()) {
    Serial.println("Failed to perform reading :(");
    return;
  }
  Serial.print("Temperatura = ");
  Serial.print(bmp.temperature);
  Serial.println(" C°");

  Serial.print("Pressione = ");
  Serial.print(bmp.pressure / 100.0);
  Serial.println(" hPa");

  Serial.println();
  delay(2000);
}
```

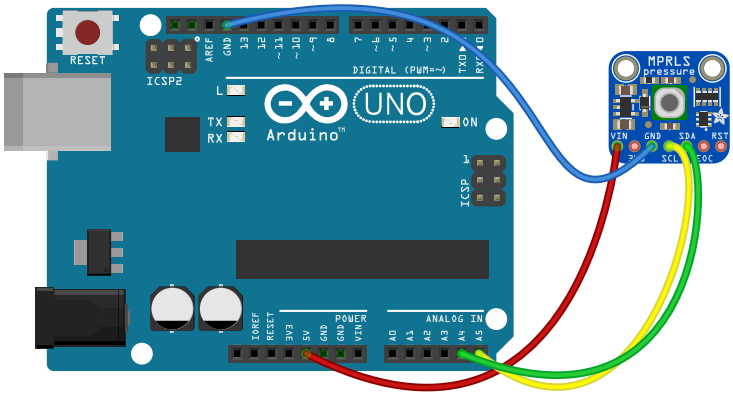
- Le righe 17-19 richiamano le librerie necessarie per utilizzare il protocollo I2C e per interfacciarsi più facilmente con il sensore.
- La riga 20 è un cambio di nome di variabile.
- La riga 23 seleziona la velocità di comunicazione tra Arduino e l'esterno.
- La riga 24 attende che la comunicazione seriale sia disponibile.
- La riga 27 controlla che il sensore sia attivo; altrimenti vengono inviati opportuni messaggi.
- Le righe 32-34 servono a selezionare le modalità interne di campionamento dei dati. Con le opzioni selezionate aumentiamo la risoluzione delle misure di pressione, con un lieve filtro IIR; avremo precisione e stabilità nelle misure a fronte di una certa inerzia ai cambiamenti: è quello che ci servirà nell'esperienza in cui useremo questo sensore.
- La riga 37 controlla se è possibile leggere i dati dal sensore; altrimenti viene dato un messaggio di errore.
- Le riga da 41 a 49 stampano la temperatura in celsius e la pressione in ettopascal.
- La riga 50 mette Arduino in pausa per un secondo.

7.4 Adafuit MPRLS

La particolarità di questa scheda è che il sensore è inserito in un involucro metallico con beccuccio che può essere connesso ad un tubicino di gomma. In questa maniera possiamo mettere il sensore nelle condizioni di misurare la pressione anche all'interno di liquidi come l'acqua o in luoghi dove la scheda non potrebbe essere posizionata.

Il sensore propriamente contenuto nella scheda non è ufficialmente citato dalla Adafruit. Si dovrebbe trattare di un Honeywell 0160KA o 0160KG della serie MPR. Questo sensore ha una portata massima di 1,6 bar e minima di 0 bar. Viene consigliato di usare un tubicino opaco perché il sensore può essere sensibile alla luce. La sensibilità non viene riportata; viene data una valutazione particolare dell'accuratezza: rimandiamo alla scheda tecnica del sensore per i particolari.

7.4.1 Schema di montaggio



Il sensore è dotato di sette pin: ne useremo quattro.

1. Il pin denominato GND va messo a terra e collegato al pin GND su Arduino;
2. Il pin VCC va collegato all'alimentazione, nel nostro caso lo colleghiamo al pin a 5,0 V di Arduino;
3. Il pin SCL è indicato come "I2C serial bus clock input": lo colleghiamo al pin A5;
4. Il pin SDA è indicato come "I2C serial bus data (or SPI input)": lo colleghiamo al pin A4.

Purtroppo il sensore non è pronto per un immediato collegamento elettrico. Viene consigliato di saldare i piedini forniti in dotazione. Nelle nostre prove è bastato avvolgere del cavo elettrico di sezione 22 AWG negli anelli della scheda. Questo tipo di collegamento è stato usato per due esperienze, ma è molto precario. Nello studio della legge di Boyle si intravvede nell'immagine che invece i pin sono stati saldati per poter collegare il sensore direttamente ad una breadboard.

La comunicazione tra sensore e Arduino è realizzata attraverso il protocollo di comunicazione I2C. Non abbiamo bisogno di conoscere le specifiche di questo protocollo: la libreria Wire si occupa di questi particolari. Inoltre la comunicazione con il sensore è mediata da un apposita libreria scaricabile al link Adafruit_MPRLS. Questa libreria non è solitamente preinstallata nell'IDE di Arduino e va installa a parte. La libreria è concessa con licenza MIT: questo è il motivo per cui riportiamo il seguente codice con tutto il testo ad esso allegato.

7.4.2 Codice per l'utilizzo

Il seguente codice, dopo un controllo iniziale sullo stato della scheda, ha come unico scopo quello di mostrare una volta al secondo la pressione in ettopascal.

```
/*!
 * @file mprls_simpletest.ino
 *
 * A basic test of the sensor with default settings
 *
 * Designed specifically to work with the MPRLS sensor from Adafruit
 * ----> https://www.adafruit.com/products/3965
 *
 * These sensors use I2C to communicate, 2 pins (SCL+SDA) are required
 * to interface with the breakout.
 *
 * Adafruit invests time and resources providing this open source code,
 * please support Adafruit and open-source hardware by purchasing
 * products from Adafruit!
 *
 * Written by Limor Fried/Ladyada for Adafruit Industries.
 *
 * MIT license, all text here must be included in any redistribution.
 *
 */

#include <Wire.h>
#include "Adafruit_MPRLS.h"

// You dont *need* a reset and EOC pin for most uses,
// so we set to -1 and don't connect
#define RESET_PIN  -1 // set to any GPIO pin # to hard-reset on begin()
#define EOC_PIN    -1 // set to any GPIO pin to read end-of-conversion by pin
Adafruit_MPRLS mpr = Adafruit_MPRLS(RESET_PIN, EOC_PIN);

void setup() {
  Serial.begin(115200);
  Serial.println("MPRLS Simple Test");
  if (! mpr.begin()) {
    Serial.println("Failed to communicate with MPRLS sensor, check wiring?");
    while (1) {
      delay(10);
    }
  }
  Serial.println("Found MPRLS sensor");
}

void loop() {
  float pressure_hPa = mpr.readPressure();
  Serial.print("Pressione (hPa): ");
  Serial.println(pressure_hPa);
  delay(1000);
}
```

- Le righe 22 e 23 richiamano le librerie necessarie per utilizzare il protocollo I2C e per interfacciarsi con il sensore.
- Le righe 27 - 29 hanno una funzione che non ho capito, ma il commento del codice dice di lasciarle così per la maggior parte degli utilizzi.
- La riga 32 seleziona la velocità di comunicazione tra Arduino e l'esterno.
- La riga 34 richiama un test sul sensore: se fallisce si entra in un loop infinito alle righe 36 - 38.
- La riga 44 esegue la funzione principale per ottenere la pressione dal sensore in ettopascal.
- La riga 46 stampa la pressione letta.
- La riga 47 manda Arduino in pausa per un secondo.

8 Misure di campo magnetico

8.1 Introduzione

Il campo magnetico (o induzione magnetica) può essere definito come il campo che può esercitare una forza su una carica elettrica in moto e che è in relazione alla velocità della carica. Se la carica è ferma non sentirà alcuna forza dovuta al campo magnetico.

I sensori di campo magnetico oggi in commercio sono essenzialmente basati su quello che è detto “Effetto Hall”. Se un conduttore attraversato da corrente i si trova in una regione di spazio in cui è presente un campo magnetico perpendicolare alla velocità di propagazione dei portatori di carica, in generale, alle estremità del conduttore, si manifesterà una differenza di potenziale:

$$\Delta V_H = ivd \tag{8.1}$$

dove v è la velocità dei portatori di carica e d è la larghezza del conduttore.

8.2 Sunfounder - magnetic sensor

Il nostro sensore è realizzato con il sensore Hall SS49E e con l'integrato LM393 (amplificatore lineare). Il sensore Hall emette un tensione direttamente proporzionale al campo magnetico, purché questo abbia un'intensità massima di circa 100 mT. La sensibilità tipica è 1,4 mV/Gauss con alimentazione a 5 V (1 Gauss = 0,0001 T). Se il campo è solo quello terrestre la tensione in uscita è centrata su circa 2,5 V. É possibile alimentare il sensore anche a 3,3 V diminuendo portata e sensibilità.

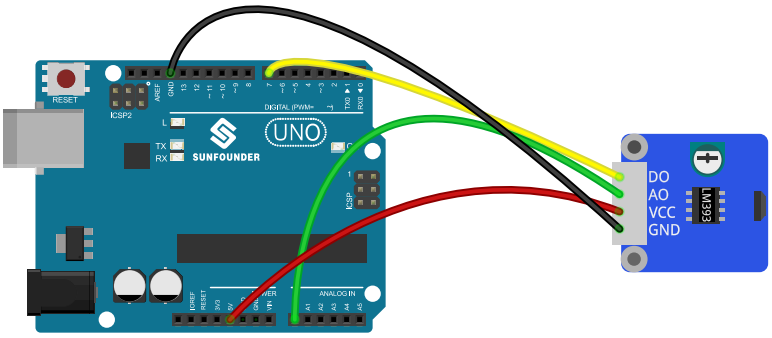
Per quanto riguarda l'interpretazione dei dati osserviamo che non otteniamo direttamente un valore per il campo magnetico, ma solo una tensione ad esso proporzionale. Se poniamo un polo nord di un magnete difronte alla faccia marchiata del sensore esso emetterà una tensione inferiore a quello di riferimento; viceversa se siamo in presenza di un polo sud. Inoltre il sensore misura l'intensità di un campo magnetico perpendicolare alla faccia marchiata del sensore, ovvero il flusso del campo che attraversa il piano del sensore.

Nei grafici allegati all'integrato SS49E si osserva che il segnale in uscita vale circa 2,5 V con un campo pressoché nullo e circa 4,0 V per un campo di 0,1 T. Arduino Uno può discriminare tra 1024 livelli per un segnale di valore massimo 5,0 V. Il nostro segnale può variare di ±1,5 V; Arduino vedrà circa 307 livelli e quindi una sensibilità di circa $3{,}2\times10^{-4}$ T. Per trasformare il valore di tensione letto in tesla possiamo moltiplicare il valore numerico della porta analogica di Arduino (che si ha rispetto al valore di equilibrio) per la sensibilità. La sensibilità è superiore al campo magnetico terrestre, che quindi non è rilevante ai fini delle misure con tale sensore.

Nell'uso si trova che quando il sensore opera oltre la sua portata va in saturazione; per valori di campo vicini alla portata il segnale non è più direttamente proporzionale al campo.

Questo tipo di sensori è soggetto ad isteresi. La documentazione del nostro sensore non riporta nulla a riguardo, ma teniamo conto dell'eventuale presenza di questo comportamento.

8.2.1 Schema di montaggio



Il sensore è dotato di quattro pin:

1. Il pin denominato GND va messo a terra e collegato al pin GND su Arduino;
2. Il pin VCC va collegato all'alimentazione; lo colleghiamo al pin a 5,0 V;
3. Il pin D0 fornisce un'indicazione sulla presenza o meno di un campo magnetico; può assumere solo due stati; lo colleghiamo al pin D7 di Arduino;
4. Il pin A0 fornisce un valore analogico correlato all'intensità del campo magnetico; lo colleghiamo al pin A0 di Arduino.

8.2.2 Codice per l'utilizzo

```
const int ledPin = 13;
int sensorPin = A0;
int digitalPin = 7;
int sensorValue = 0;
boolean digitalValue = 0;

void setup()
{
  pinMode(digitalPin,INPUT);
  pinMode(ledPin,OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  sensorValue = analogRead(sensorPin);
  digitalValue = digitalRead(digitalPin);
  Serial.print("Valore␣del␣sensore␣␣");
  Serial.print(sensorValue);
  Serial.println("␣␣");

  if( digitalValue==HIGH )
  {
    digitalWrite(ledPin,LOW);
  }
  if( digitalValue==LOW)
  {
    digitalWrite(ledPin,HIGH);
  }
  delay(200);//delay 200ms
}
```

- La riga 1 dice che il led del sensore è collegato al pin 13.
- La riga 2 seleziona il pin di input per il potenziometro.
- La riga 3 seleziona il pin di input digitale della porta D0.
- La riga 4 nomina la variabile che contiene il valore letto dalla porta A0.
- La riga 5 nomina la variabile che contiene il valore letto dalla porta D0.
- La riga 9 seleziona il pin 7 (D0) come input.
- La riga 10 selezione il pin 13 (led) come output.
- La riga 11 seleziona la velocità di comunicazione tra Arduino e l'esterno.
- La riga 16 legge il valore di A0.
- La riga 17 legge il valore di D0.
- La riga 19 stampa sulla porta seriale il valore del segnale inviato dal sensore.
- La righe 22-29 accendono il led sul sensore se viene misurato un segnale che supera una soglia di riferimento. Se se il valore di D0 è HIGH spegne il led; altrimenti accende il led.
- La riga 30 mette Arduino in pausa per 200 ms.

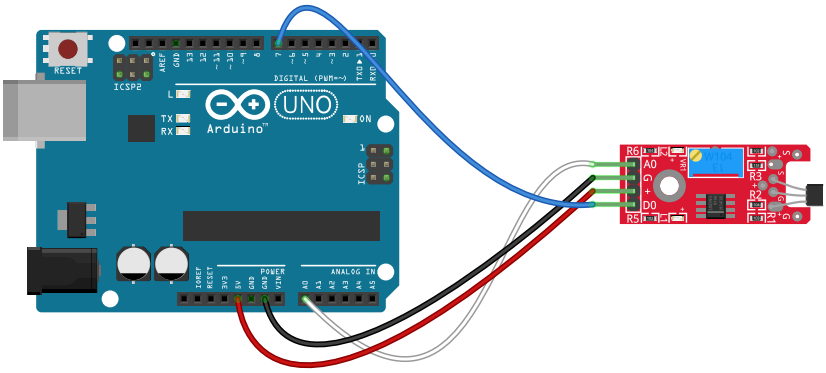
8.3 Az-delivery KY-024

Anche questo sensore è realizzato con l'integrato LM393 (amplificatore lineare) e il sensore Hall SS49E. A differenza dell'altra realizzazione questa è di molto più facile reperibilità e con un costo inferiore. Avendo gli stessi componenti base valgono le considerazioni fatte prima. Anche il codice per il suo utilizzo può essere lo stesso.

Una differenza con il sensore precedente è nella vite presente sulla scheda, che si intravede nello schema di montaggio sul componente azzurro presente sulla scheda. L'utilità della vite, per me limitata nell'uso nel laboratorio di fisica, è quella di variare la sensibilità del sensore: questo è quel che indica il manuale. Da quel che ho potuto capire la rotazione della vite cambia il comportamento della scala dei valori per valori di campo positivi rispetto a quelli negativi. A seconda della rotazione prevalente della vite in senso antiorario piuttosto che orario prevale la sensibilità per valori del campo di un certo segno rispetto all'altro. Sarebbe opportuno trovare il punto di equilibrio tra le due sensibilità in modo da avere una scala che non dipende dal verso del campo, ottenendo un comportamento del tutto simile a quello del sensore Sunfounder.

Purtroppo il sensore è montato, come si vede nella figura, sullo stesso piano della scheda. Nell'uso pratico ho trovato più utile avere il piano del sensore perpendicolare al piano della scheda. Si può cercare di ovviare a questo inconveniente inclinando delicatamente (e definitivamente) il sensore in quel modo con le mani o delle pinzette: questa operazione è da intendersi quasi definitiva perché i piedini del sensore sicuramente si romperebbero dopo pochissimi aggiustamenti.

8.3.1 Schema di montaggio



Il sensore è dotato di quattro pin:

1. Il pin denominato G va messo a terra e collegato al pin GND su Arduino;
2. Il pin + va collegato all'alimentazione; lo colleghiamo al pin a 5,0 V;
3. Il pin D0 fornisce un'indicazione sulla presenza o meno di un campo magnetico; può assumere solo due stati; lo colleghiamo al pin D7 di Arduino;
4. Il pin A0 fornisce un valore analogico correlato all'intensità del campo magnetico; lo colleghiamo al pin A0 di Arduino.

8.4 Adafruit tlv493d

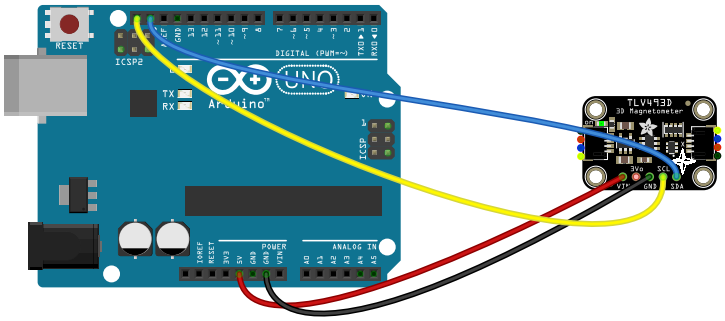
I due sensori precedenti hanno due difetti: non sono tarati e misurano il campo lungo un solo asse. Il sensore tlv493 della Infineon è invece tarato dal costruttore e misura il campo su tre assi XYZ. Il costo è solamente un po' superiore agli altri due.

Questo sensore fornisce direttamente la misura del campo in forma digitale con una risoluzione a 12 bit; il comportamento è lineare sino a circa 130 mT; lo zero ha una fluttuazione di circa 0,2 mT e quindi non è adatto a misurare il campo magnetico terrestre. L'indicazione del costruttore sulla sensibilità è un po' complessa e rimandiamo al datasheet del sensore: possiamo dire che con misure di qualche millitesla essa è dell'ordine di qualche decimo di millitesla. L'accuratezza non è riportata, ma per i nostri scopi la compariamo alla sensibilità.

Infine, per l'indicazione su quali siano gli assi di riferimento per la misurazione, oltre a rimandare ugualmente ai disegni del datasheet del sensore, possiamo dire (in riferimento all'immagine dello schema di montaggio) che l'asse x è in verticale, quello y in orizzontale e l'asse z è perpendicolare al piano del disegno.

Il sensore necessita di apposita libreria scaricabile all'indirizzo `https://www.arduino.cc/reference/en/libraries/tlv493d-a1b6/` in formato zip. Questo file va caricato nell'IDE di Arduino. Dalla voce del menù principale "Sketch" si passa a "#include libreria" e infine "Aggiungi libreria da file .zip" con la quale si apre una finestra di dialogo dalla quale possiamo specificare il file precedentemente scaricato.

8.4.1 Schema di montaggio



La comunicazione tra sensore e Arduino è realizzata attraverso il protocollo di comunicazione I2C: tutto è gestito dalla libreria TLV494d.

Il sensore è dotato di cinque pin:

1. Il pin denominato GND va messo a terra e collegato al pin GND su Arduino;
2. Il pin VIN va collegato all'alimentazione a 5,0 V; lo possiamo alimentare anche a 3,3 V col pin 3Vo.
3. Il pin SCL è indicato come "I2C serial bus clock input": lo colleghiamo al pin SCL;
4. Il pin SDA è indicato come "I2C serial bus data (or SPI input)" lo colleghiamo al pin SDA.

Possiamo effettuare il collegamento come indicato in figura, eventualmente saldando i pin sulla scheda. Oppure (come ho preferito fare personalmente) si può collegare un apposito connettore sulla sinistra della scheda che ha in uscita gli stessi quattro cavi indicati nello schema precedente.

8.4.2 Codice per l'utilizzo

```
1  #include <Tlv493d.h>
   Tlv493d tlv = Tlv493d();
3  float mx;
   float my;
5  float mz;
   float mtot;
7  void setup() {
     Serial.begin(9600);
9    while(!Serial);
     tlv.begin();
11 }
   void loop() {
13   tlv.updateData();
     delay(100);
15   mx = tlv.getX();
     my = tlv.getY();
17   mz = tlv.getZ();
     mtot = tlv.getAmount();
19   Serial.print("X = ");
     Serial.print(mx);
21   Serial.print(" mT; Y = ");
     Serial.print(my);
23   Serial.print(" mT; Z = ");
     Serial.print(mz);
25   Serial.print(" mT; TOT = ");
     Serial.print(mtot);
27   Serial.println(" mT");
     delay(500);
29 }
```

- La riga 1 richiama la libreria necessaria per il funzionamento del sensore.
- La riga 2 crea un'istanza chiamata tlv.
- Le riga 3-6 creano delle variabili in cui carichiamo le componenti e il modulo del campo.
- La riga 8 seleziona la velocità di comunicazione con Arduino.
- La riga 9 verifica se il collegamento seriale è disponibile.
- La riga 10 inizializza il sensore.
- La riga 13 acquisisce i dati per una lettura.
- La riga 12 dà il tempo al sensore per la lettura stessa.
- Le righe 15-18 leggono le componenti e il modulo del campo.
- Le righe 19-27 stampano i dati ottenuti.
- La riga 28 mette Arduino in pausa per 500 ms.

9 Misure di temperatura

9.1 Introduzione

La temperatura può essere definita come la grandezza che si misura con un termometro. Il termometro tradizionale sfrutta le proprietà di dilatazione di un liquido al variare della temperatura. Nel nostro caso i sensori sono basati prevalentemente sulle proprietà delle termocoppie. In ogni caso le specifiche fisiche di funzionamento non sono date nel dettaglio. Ciò che è fondamentale conoscere sono le condizioni operative, la portata e l'accuratezza delle misure.

9.2 DS18B20

Il primo sensore che analizziamo è il DS18B20 della Maxim Integrated, in origine della Dallas Semiconductor. É possibile utilizzarlo sia nella forma di un piccolo elemento circuitale con tre pin, come lo rappresentiamo qui di seguito, oppure incapsularlo in un contenitore stagno di acciaio in modo da poterlo immergere in un liquido. Il sensore ha la forma esterna di un transistor del tipo TO-92.

Il termometro digitale DS18B20 fornisce misurazioni della temperatura in gradi centigradi in una scala compresa tra $-55\,°C$ e $125\,°C$ con una accuratezza di $\pm 0{,}5\,°C$ nell'intervallo da $-10\,°C$ a $85\,°C$. Il DS18B20 comunica con un sistema di comunicazione ad un filo detto 1-Wire. Questo protocollo di comunicazione permette di interfacciare contemporaneamente più dispositivi logici ad un microcontrollore mediante un solo filo, oltre a quello di collegamento a massa (GND). In questa sede non illustriamo il funzionamento di questo bus, ma ci limitiamo ad utilizzarlo seguendo gli esempi di utilizzo forniti in rete.

Il particolare protocollo di comunicazione ci obbliga ad inserire due apposite librerie, normalmente non fornite, al codice di Arduino.

La prima, chiamata OneWire, ci serve per usare il protocollo 1-Wire: la possiamo scaricare all'indirizzo `https://www.arduinolibraries.info/libraries/one-wire` in formato zip.

La seconda, chiamata DallasTemperature, è specifica per il sensore: la possiamo scaricare all'indirizzo `https://www.arduinolibraries.info/libraries/dallas-temperature` sempre in formato zip. Entrambi questi file vanno caricati nell'IDE di Arduino. Dalla voce del menù principale "Sketch" si passa a "#include libreria" e infine "Aggiungi libreria da file .zip" con la quale si apre una finestra di dialogo dalla quale possiamo specificare, uno alla volta, i due file precedentemente scaricati.

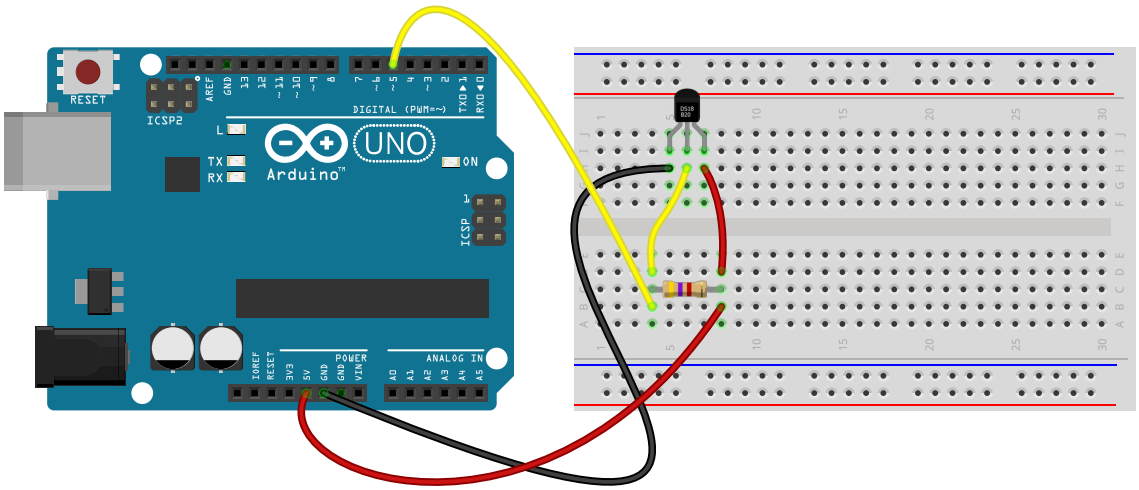
Il sensore è dotato di tre pin:

1. Il pin denominato GND va messo a terra e collegato al pin GND su Arduino;
2. Il pin Vdd va collegato all'alimentazione; lo colleghiamo al pin a 5,0 V;
3. Il pin DQ è relativo all'input e output dei dati: è possibile alimentare il sensore da questa linea, ma non utilizzeremo questa possibilità.

Come osserviamo nello schema di montaggio seguente, il sensore va collegato per mezzo di una resistenza di pull-up da 4,7 kΩ interposta tra la linea dati e quella di alimentazione. Lo scopo di questa resistenza è quello di rendere il segnale della linea dati meno soggetto a disturbi o interferenze esterne.

Il sensore ha un ADC interno la cui risoluzione può essere impostata via software. In particolare possiamo scegliere una risoluzione compresa tra 9 e 12 bit a cui corrisponde una risoluzione per la temperatura compresa tra 0,5 °C e 0,0625 °C. A maggiore risoluzione corrisponde un tempo di lettura maggiore. La risoluzione è memorizzata permanente nella memoria interna del sensore. Essendo possibile un numero limitato di cicli di riscrittura si consiglia di settare la risoluzione prima dell'uso e poi utilizzare un codice in cui la risoluzione non viene più cambiata.

9.2.1 Schema di montaggio



9.2.2 Codice per l'utilizzo

Il codice seguente fornisce il valore della temperatura misurato ad intervalli di un secondo, impostando una risoluzione a 12 bit.

```
1  #include <OneWire.h>
   #include <DallasTemperature.h>
3  #define ONE_WIRE_BUS 5
   OneWire oneWire(ONE_WIRE_BUS);
5  DallasTemperature sensors(&oneWire);
   DeviceAddress tempDeviceAddress;
7
   int  resolution = 12;
9
   void setup(void)
11 {
    Serial.begin(115200);
13  sensors.begin();
    sensors.getAddress(tempDeviceAddress, 0);
15  sensors.setResolution(tempDeviceAddress, resolution);
   }
17
   void loop(void)
19 {
    sensors.requestTemperatures();
21  Serial.print("La␣temperatura␣è:␣");
    Serial.println(sensors.getTempCByIndex(0));
23  delay(1000);
   }
```

- La riga 1 carica la libreria relativa al protocollo 1-Wire.
- La riga 2 carica la libreria relativa al sensore.
- La riga 3 dice che la linea dati è collegata al pin 5 di Arduino.
- La riga 4 stabilisce un'istanza oneWire per comunicare con ogni dispositivo OneWire presente.
- La riga 5 passa il nostro riferimento oneWire al dispositivo.
- La riga 6 stabilisce un'istanza per conoscere e utilizzare l'indirizzo del singolo dispositivo connesso sulla linea OneWire.
- La riga 8 dichiara e inizializza la variabile che definisce la risoluzione.
- La riga 12 seleziona la velocità di comunicazione tra Arduino e l'esterno.
- La riga 13 inizializza la libreria.
- La riga 14 ottiene l'indirizzo del primo dispositivo (il numero 0) connesso.
- La riga 15 definisce a quell'indirizzo la risoluzione voluta.
- La riga 20 chiama l'istruzione sensors.requestTemperatures() come richiesta di fornire la temperatura a tutti i dispositivi connessi al bus.
- La riga 22 comando di fornire la temperatura letta dal primo dispositivo presente sul bus.
- La riga 23 mette Arduino in pausa per 1000 ms.

Se durante l'utilizzo compare una temperatura tipo −127 °C vuol dire che ci sono sicuramente o problemi elettrici nel collegamento o si sta selezionando un pin errato per il funzionamento. Il sensore ha funzionato sia con Arduino UNO R3 che R4.

9.2.3 Studio delle prestazioni e taratura

Il sensore fornisce valori di temperatura estremamente stabili: è sufficiente una stabilizzazione iniziale di qualche minuto affinché la temperatura indicata risulti attendibile e totalmente stabile nel tempo.

Per quanto riguarda l'accuratezza delle misurazioni ho provveduto ad immergere il sensore in un bicchiere di ghiaccio fondente e un recipiente di acqua bollente. Nel primo caso la temperatura segnata è stata $t_0 = 0{,}25\,°\text{C}$ e nel secondo caso $t_{100} = 98{,}5\,°\text{C}$. Questi risultati sono allineati col livello di accuratezza indicato in rete per altri esemplari del sensore.

I due valori prima trovati possono essere utilizzati per riscalare la temperatura t_s indicata dal sensore ad un valore t^* più accurato. La relazione da usare è la seguente.

$$t^* = (t_s - t_0) \cdot \frac{100}{t_{100} - t_0} \tag{9.1}$$

Possiamo modificare il codice di gestione nella seguente maniera.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 5
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
DeviceAddress tempDeviceAddress;

int  resolution = 12;
float t0 = 0.25;
float t100 = 98.5;
float ts, t;

void setup(void)
{
 Serial.begin(115200);
 sensors.begin();
 sensors.getAddress(tempDeviceAddress, 0);
 sensors.setResolution(tempDeviceAddress, resolution);
}

void loop(void)
{
 sensors.requestTemperatures();
 ts = sensors.getTempCByIndex(0);
 t = (ts-t0)*100/(t100-t0);

 Serial.print("La␣temperatura␣è:␣");
 Serial.println(t);
 delay(1000);
}
```

10 Misure di presenza nel tempo

10.1 Introduzione

Alcuni sensori servono a determinare se un qualche oggetto è presente o meno in prossimità del sensore stesso. Questo tipo di sensori possono avere un tempo di reazione molto breve, dell'ordine di qualche millisecondo. Di per sé la determinazione della presenza di un oggetto negli esperimenti del laboratorio di fisica è di relativa importanza. Tuttavia questi sensori, se accoppiati ad un misuratore di tempo, possono essere utili per fare misurazioni di tempo tra eventi, ad esempio per determinare quanto tempo un oggetto impiega per andare da una posizione ad un altra.

Questi sensori sono essenzialmente delle fotocellule. In questo documento facciamo riferimento a due dispositivi distinti.

Il primo, chiamato "IR obstacle sensor" dalla Sunfounder, è un *sensore fotoelettrico a tasteggio diretto*. In questi dispositivi abbiamo una sorgente (tipicamente un LED ad infrarossi) che manda un fascio di fronte a sé. Quando il fascio colpisce un oggetto può essere riflesso tornando indietro. Di fianco alla sorgente è posizionato, nello stesso dispositivo, un sensore. Se il fascio riflesso supera una certa soglia il dispositivo si accorge della presenza dell'oggetto posto in prossimità. L'oggetto da individuare deve essere sia vicino (anche pochi centimetri), sia abbastanza riflettente. Questo dispositivo e anche il successivo operano meglio con poca luce ambientale: è sicuramente da evitare la luce diretta del sole.

Il secondo dispositivo, chiamato "IR break beam sensor" dalla Adafruit, è un *sensore fotoelettrico a barriera*. Anche qui abbiamo una sorgente ad infrarossi e un sensore fotoelettrico, ma in due oggetti distinti. Essi vanno posizionati allineati uno di fronte all'altro. Quando un oggetto passa davanti al fascio di luce il sensore non riceve più segnale, indicando la presenza dell'oggetto stesso. In questo caso la distanza di rilevamento può essere più grande e soprattutto l'oggetto non deve essere riflettente.

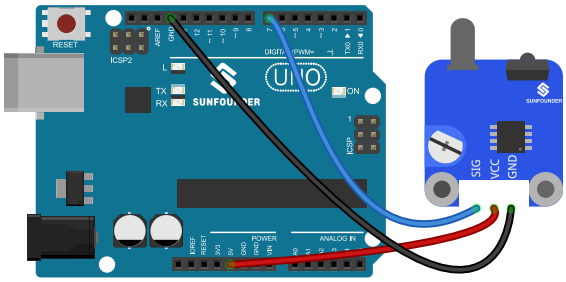
10.2 Schema di montaggio I

Illustriamo qui di seguito due apparecchi funzionanti secondo le due modalità prima illustrate. La modalità di interazione con Arduino è tuttavia del tutto simile.

Il primo apparato di misurazione è costituito da due componenti:

1. Una scheda Arduino (uno)
2. IR obstacle sensor - Sunfounder

Questo sensore ha una distanza di rilevamento compresa tra 3 cm e 30 cm, con un fascio largo probabilmente 25°. Sul dispositivo è presente una vite per regolare la distanza massima di rilevamento; è inoltre presente un led che si accende quando viene individuato un oggetto. Il costo è di circa 10 euro, ma ci sono apparecchi simili anche per meno di due euro se comprati in quantità.



10.3 Codice per l'utilizzo

```
const int sensorPin = 7;
void setup() {
  pinMode(sensorPin, INPUT);
}
void loop(){
  boolean sensorVal = digitalRead(sensorPin);
  if(sensorVal == LOW)
  {
    Serial.println("c'è");
  }
  else
  {
    Serial.println("non c'è");
  }
}
```

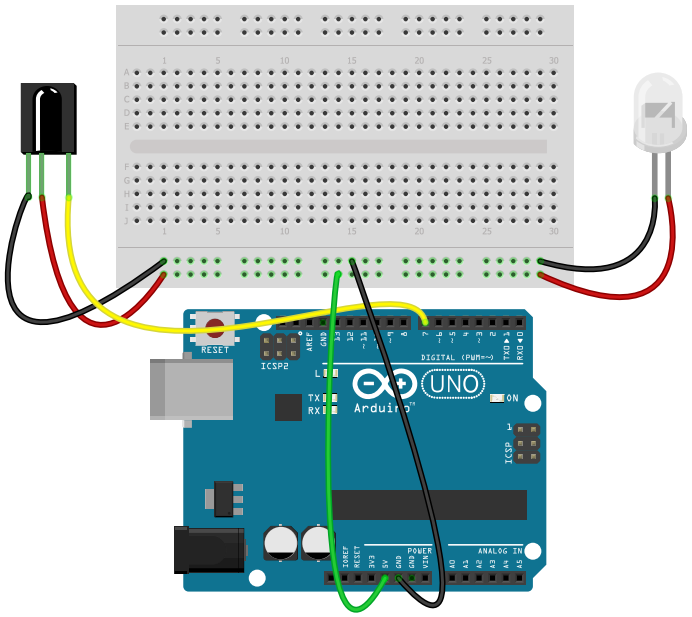
- La riga 1 definisce il nome della variabile che contiene il pin a cui inviare il segnale di presenza
- Dalla riga 2 inizia la funzione iniziale di setup.
- La riga 3 attiva il pin echoPin in modalità di input.
- Dalla riga 5 inizia la funzione iniziale di setup.
- La riga 6 legge dal pin del sensore lo stato e lo memorizza della variabile sensorVal.
- Alla riga 8 se quel valore è "LOW" viene scritto "c'è";
- alla riga 13 altrimenti viene scritto "non c'è".

10.4 Schema di montaggio II

Il secondo strumento è invece realizzato con tre parti, per meglio gestire i cavi presenti.

1. Una scheda Arduino (uno)
2. breadboard
3. IR break beam sensor - 3mm LED - Adafruit

Questo sensore ha una distanza di rilevamento massima di circa 25 cm, con un fascio largo 10°; conosciamo il tempo di risposta tipico: meno di 2 ms. Sui due oggetti è presente un foro per attaccarli ad una superficie di appoggio. Questo modello ha il LED e il sensore larghi 3 mm. Esiste anche una variante da 5 mm con una portata massima raddoppiata. Il costo è di circa 3 euro; circa 10 per la versione più grande.



10.5 Codice per l'utilizzo

```
1  const int sensorPin = 7;
   void setup() {
3    pinMode(sensorPin, INPUT);
     digitalWrite(sensorPin, HIGH);
5    //pinMode(sensorPin, INPUT_PULLUP);
   }
7  void loop() {
     boolean sensorVal = digitalRead(sensorPin);
9    if(sensorVal == LOW)
     {
11     Serial.println("c'è");
     }
13   else
     {
15     Serial.println("non c'è");
     }
17 }
```

La differenza con il codice precedente è solamente la presenza della riga 3 e 4. Con questo sensore abbiamo la necessità di garantire un corretto invio del segnale dal sensore ad Arduino, non avendo il sensore un circuito interno minimale. In condizioni operative normali dal sensore esce un segnale HIGH; quando il traguardo viene interrotto esce un segnale LOW. In questa seconda condizione il pin si troverebbe in uno stato flottante: per impedire questo ultimo stato possiamo inserire una resistenza di pullup nella linea. Questo è quello che facciamo con l'istruzione alla riga 4. In alternativa possiamo scrivere la riga 5 invece delle due precedenti.

11 Misure di tensione

11.1 Introduzione

Una delle funzionalità principali delle schede come Arduino Uno è quella di misurare la tensione in ingresso ad uno dei suoi pin tramite un convertitore analogico digitale ADC. L'ADC di Arduino Uno può misurare una tensione massima di 5,0 V discriminando tra 1024 livelli, avendo un ADC che funziona a 10 bit ($2^{10} = 1024$). In questo manuale non abbiamo le competenze per analizzare in profondità tutte le problematiche inerenti il funzionamento di simili dispositivi. Mi limiterò ad evidenziare alcuni punti.

Se i livelli di lettura sono 1024 è da sapere che il livello 1 corrisponde a 0 V e il livello 1024 non alla tensione di riferimento, ma ad essa per 1023. In particolare la tensione massima leggibile è $1023 \cdot (5/1024) = 4{,}9951$ V.

La tensione di riferimento usata di default per le letture è 5,0 V, tuttavia è possibile usare anche 1,1 V o una tensione esterna. La scelta può essere impostata con la funzione analogReference() a cui passare il valore: DEFAULT, per il riferimento di default di 5,0 V; INTERNAL, per un riferimento integrato, equivalente a 1,1 V sull'ATmega168; EXTERNAL, per la tensione applicata al pin AREF (solo da 0 a 5,0 V).

Osserviamo che in letteratura è riportato che la lettura in DEFAULT ci può dare un errore anche del 4 %, mentre quella fatta in INTERNAL è associabile ad un errore di solo il 0,4 % o meno. Bisogna osservare inoltre che la tensione di riferimento in DEFAULT è la stessa tensione di 5,0 V nominali erogabile dai pin digitali. Se usiamo l'impostazione INTERNAL per l'ADC i pin forniranno ancora 5,0 V nominali.

L'ADC di Arduino Uno o Due è un ADC ad approssimazioni successive, il che vuol dire che la tensione viene letta in step successivi. Normalmente ci vogliono n cicli di clock per una lettura a n bit; per Arduino Uno i cicli necessari sono 13. In pratica l'ADC ha una velocità di acquisizione dati limitata a circa 10000 letture al secondo e quindi esiste un problema di ritardo nella risposta. Arduino Due invece, così come la maggior parte delle schede di ultima generazione, ha un ADC a 12 bit e con una velocità di lettura più elevata, limitati ad una tensione massima di 3,3 V.

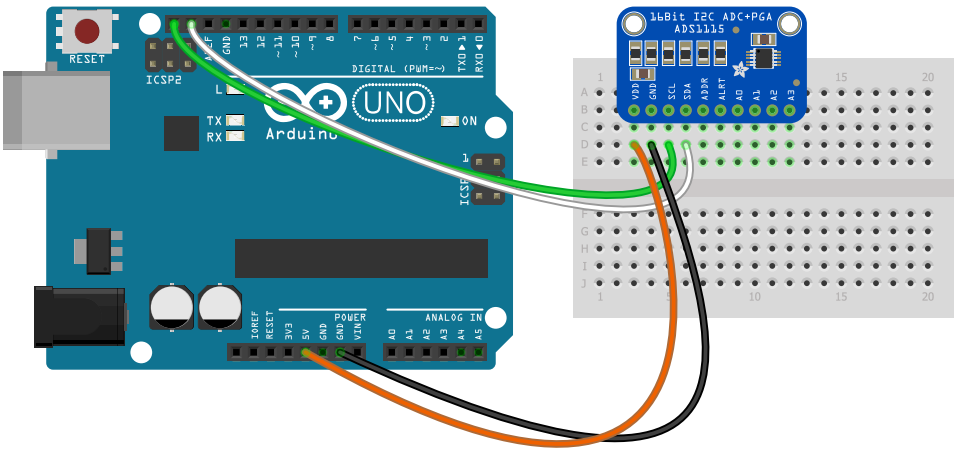
Infine alla lettura su Arduino Uno è associato un errore assoluto di 2 LSB. L'LSB o Least significant bit per una lettura può avere il significato di errore di sensibilità sull'ultima cifra letta. In pratica l'errore è comunque maggiore.

11.2 ADS 1115

Per migliorare la risoluzione nella misura di una differenza di potenziale, a fronte di una velocità di lettura piuttosto limitata ma normalmente del tutto adeguata, è disponibile l'ADC "Texas Instruments ADS 1115": è un ADC a 16 bit con un data rate compreso tra 8 SPS e 860 SPS. La portata è selezionabile tra 0,256 V e 6,144 V. Il circuito funziona con una interfaccia I2C e dispone di quattro ingressi.

1. Una scheda Arduino (uno)
2. Texas Instruments ADS 1115
3. breadboard e cavetteria.

11.2.1 Schema di montaggio



La particolare scheda in nostro possesso è di quelle dotate di connettori ad anello dorato: possiamo collegarci ad essi per mezzo di un filo, ma si tratterebbe di una connessione di scarsa qualità. Quindi è quasi assolutamente necessario saldare dei connettori per poter collegare la scheda ad una breadboard o per inserire nei connettori delle prese Du Pont.

11.2.2 Codice per l'utilizzo

```
1  #include "Wire.h"
   #include <Adafruit_ADS1X15.h>
3  Adafruit_ADS1115 ads;
   void setup(void)
5  {
     Serial.begin(9600);
7    ads.setGain(GAIN_ONE);         // 1x gain   +/-4.096V
     ads.setDataRate(64);
9    ads.begin();
   }
11 void loop(void)
   {
13   int adc0;
     float volt;
15   adc0 = ads.readADC_SingleEnded(0);
     volt = adc0/8000.0;
17   // volt = ads.computeVolts(adc0);
     Serial.println(volt);
19 }
```

- Le prime due righe richiamano la libreria per la comunicazione dati con l'ADC e quella specifica per il suo funzionamento.
- La riga 3 fa un cambio di variabile per una scrittura più semplice di ciò che segue.
- La riga 6 seleziona la velocità di comunicazione tra Arduino e l'esterno.
- La riga 7 definisce il livello di guadagno dell'ADC: viene scelto quello unitario che prevede una tensione in ingresso compresa tra più o meno 4,096 V, adeguata al sensore di distanza con cui lo ho usato e che fornisce una tensione massima in uscita di circa 3,3 V.
- La riga 8 definisce il data rate: è possibile usare i valori 8, 16, 32, 64, 128, 250, 475, 860.
- Le righe 13 e 14 definiscono due variabili: il valore intero letto dal pin analogico di ingresso e la tensione.
- La riga 15 legge il valore nell'ingresso analogico.
- La riga 16 trasforma quel valore in volt considerando che la tensione di riferimento dell'ADC è 4,096 V ($2^{10} = 4096$) e il pin legge valori su una scala di $2^{15} = 32768$ livelli.
- La riga 17, in alternativa, trasforma quel valore in volt con una funzione apposita.
- La riga 18 stampa il valore di tensione letto.

11.3 Dati sperimentali

Per testare le proprietà del sensore ho collegato i terminali di un alimentatore con due pile, come oggetto più semplice ad disposizione con una tensione costante. Ho raccolto mille letture alla volta, con data rate crescenti. La tensione misurata con un voltmetro economico è stata 2,623 V. Questi sono i dati osservati dal sensore.

SPS impostato	V (V)	σ	SPS misurato
8	2,62536	0,00005	8
16	2,62540	0,00002	8
32	2,62543	0,00005	15
64	2,62547	0,00005	30
128	2,62547	0,00005	99
250	2,62542	0,00010	321

Dai risultati ottenuti il comportamento più stabile si ha con un data rate pari a 16. Per gli utilizzi delle esperienze di questo documento anche un data rate di 64 o 128 sembra del tutto adeguato. Andando oltre 250 la velocità di acquisizione effettiva non aumenta e quindi non ho riportato i dati.

Il valore di deviazione standard osservato è molto piccolo. Va confrontato col rumore quadratico medio (RMS noise) riportato nella scheda del sensore: fino ad un data rate di 128 vale circa 125 μV, più del doppio di quanto misurato. Questo valore è anche uguale al LSB size e quindi a quella che potremmo chiamare sensibilità. Per cui il comportamento reale del sensore è totalmente nelle specifiche. Devo purtroppo osservare che per letture di grandezze costanti le capacità di lettura sono probabilmente sovrastimate dal sovracampionamento che opera per i data rate inferiori. Per letture di grandezze variabili le performance diminuiscono sensibilmente, ma non ho dati sperimentali attualmente mostrabili.

12 Misure di forza

12.1 Introduzione

Un modo per misure la forza agente su un oggetto è tramite uno strumento detto *estensimetro elettrico*. In esso un sottile filo è inglobato mediante materiale plastico su un oggetto del quale si vuole misurare la deformazione e quindi la forza che su di esso sta agendo. La deformazione del filo causa una piccola variazione della sua resistenza elettrica che può essere misurata. In particolare, tra gli estensimetri elettrici, citiamo quello a ponte di Wheatstone dove quattro fili sono collegati tra loro in un unico dispositivo per migliorarne la sensibilità.

Questa tecnologia è usata per realizzare una *cella di carico*. Con questo termine possiamo indicare sia un oggetto su cui è applicato un estensimetro elettrico, sia questo stesso apparecchio più il circuito elettronico per la sua gestione. Normalmente, nei siti che vendono questi apparecchi per Arduino, si distinguono le due parti.

Nel nostro caso utilizzeremo una cella di carico metallica, con una portata di 1 kg e una scheda specializzata per la sua gestione basata sul circuito integrato HX711. Questo altro non è se non un convertitore analogico digitale a 24 bit capace di misurare la debole differenza di potenziale in uscita dalla cella di carico.

Per usare la scheda con HX711 useremo la libreria “HX711 Arduino Library” scaricabile all’indirizzo `https://www.arduino.cc/reference/en/libraries/hx711-arduino-library/`. Questo file va caricato nell’IDE di Arduino: dalla voce del menù principale “Sketch” si passa a ”#include libreria” e infine “Aggiungi libreria da file .zip” con la quale si apre una finestra di dialogo dalla quale possiamo specificare il file precedentemente scaricato.

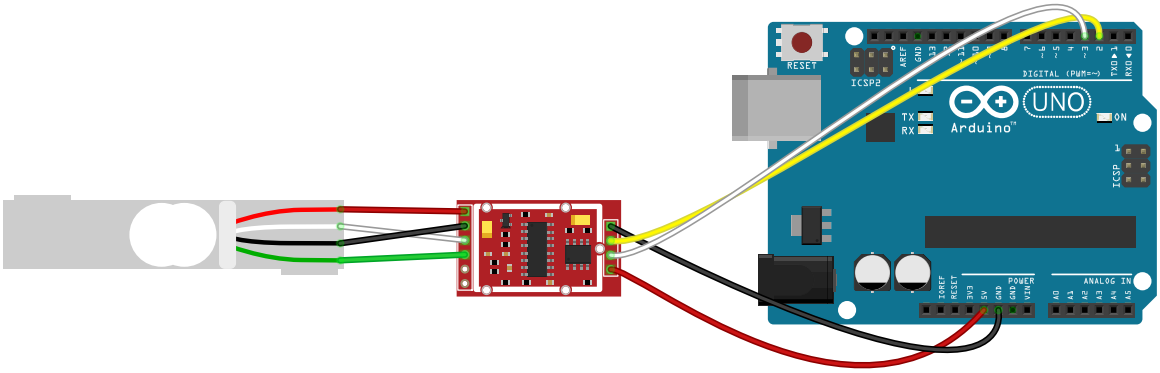
12.2 Cella di carico con HX711

Una cella di carico prima dell’uso va tarata e in particolare va appurato quanto vale la sua misura in assenza di sforzo e quanto vale con una forza nota. Il codice seguente prevede delle righe che andranno utilizzate solo nella prima fase di taratura, per poter poi essere commentate una volta definiti gli opportuni parametri per tutta la sessione di lettura. La libreria prima indicata prevede delle funzioni specifiche per queste fasi, ma preferiamo procedere in maniera più diretta.

Osserviamo che la cella di carico è normalmente venduta con i fili colorati, come indicato in figura. Questi fili tuttavia non sono terminati: potremmo saldarli definitivamente alla scheda HX711 oppure, come da me effettuato, possiamo crimpare dei classici connettori Dupont. Stesso discorso per la scheda stessa, dove ho saldato delle terminazioni Dupont, dal momento che non erano presenti nel modello da me acquistato.

L'apparato di misurazione è costituito da:

1. Una scheda Arduino (Uno).
2. scheda basata sul HX711.
3. cella di carico da 1 kg.
4. base di legno come supporto.
5. un blocco da 100 g per la taratura.



Il blocco metallico della cella di carico va tenuto fisso su una opportuna base; l'altra estremità è quella a cui applicare lo sforzo. Su questo blocco ci sono dei fori per viti: ne ho utilizzato uno per fissarlo ad una bassetta di legno che mi consentisse di tenere il blocco fermo su un piano di appoggio.

12.3 Codice per l'utilizzo

```
#include "HX711.h"
const int LOADCELL_DOUT_PIN = 2;
const int LOADCELL_SCK_PIN = 3;
long shift = 123760;
float riscala = 208105.0f;
HX711 scale;
void setup() {
  Serial.begin(57600);
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
}
void loop() {
  if (scale.is_ready()) {
    long reading = scale.read();
    Serial.print(reading);
    Serial.print("␣");
    Serial.print(reading-shift);
    Serial.print("␣");
    // le righe 14 -18 si possono commentare una volta tarato il sensore.
    Serial.println(100*(reading-shift)/riscala,3);
    }
}
```

- La riga 1 carica i riferimenti per compilare con la libreria HX711.
- La riga 2 assegna il pin 2 al pin DOUT della scheda.
- La riga 2 assegna il pin 3 al pin SCK della scheda.
- La riga 4 definisce la variabile intera "shift": essa rappresenta il valore in uscita dal sensore quando non applichiamo nessuno sforzo.
- La riga 5 definisce la variabile float "riscala": essa rappresenta il fattore di conversione per trasformare il valore in uscita dal sensore quando applichiamo una forza nota nel valore da noi attribuito a quella forza.
- La riga 5 rinomina la chiamata alla libreria come "scale".
- Dalla riga 7 inizia la funzione iniziale di setup.
- La riga 9 inizializza l'interfaccia definendo i pin alla quale è collegata.
- La riga 12 controlla che l'interfaccia sia pronta per la lettura.
- Alla riga 13, se l'interfaccia è pronta, viene fatta la lettura.

Il valore in uscita dalla nostra interfaccia è attribuito nel codice alla variabile "reading": il valore che questa variabile assume quando lo sforzo applicato è zero lo chiamiamo "shift". Di conseguenza, se la taratura è corretta, il valore di "reading - shift" in tale condizione sarà zero. Dopodiché, usando un pesetto da 100 g vogliamo appunto che il sensore ci dia come risultato il numero 100. Avremo che:

$$100 \cdot \frac{reading - shift}{riscala} = 100 \quad ; \quad riscala = reading - shift \tag{12.1}$$

Quindi, per la taratura:

1. In condizione di assenza di forza o sforzo leggiamo il valore della variabile "reading" e lo salviamo nella variabile "shift".
2. Applicando una massa nota (nel mio caso 100 g) all'estremità della cella leggiamo ora il valore della variabile "reading - shift" e lo salviamo nella variabile "riscala". Il valore in uscita della grandezza (100*(reading-shift)/riscala) ci darà la forza applicata in grammi-peso. Naturalmente, a nostra discrezione, possiamo usare come riferimento un valore in newton o quel che viene meglio.

Osserviamo che con la nostra cella di carico, della portata massima di 1 kg, la taratura prima illustrata permette di misurare i decimi di grammo. Misure più o meno accurate sono legate alla temperatura ambientale e all'accurato posizionamento della cella in orizzontale su un supporto ben stabile.

La scheda HX711 permette di acquisire i dati con un tasso selezionabile di 10 letture o 80 letture al secondo. Purtroppo per attivare le 80 letture al secondo bisogna mandare un segnale ad un determinato pin del circuito integrato. Questa possibilità non è accessibile dall'esterno della scheda per la maggior parte delle implementazioni, oppure bisognerebbe agire fisicamente sulla scheda saldando o desaldando alcune connessioni.

12.4 Studio delle prestazioni

La cella di carico, dopo la taratura, fornisce un'ottima risposta, ma se all'inizio si ha l'impressione di poter ottenere misure estremamente accurate poi ci si accorge che la taratura non sembra permanere nel tempo.

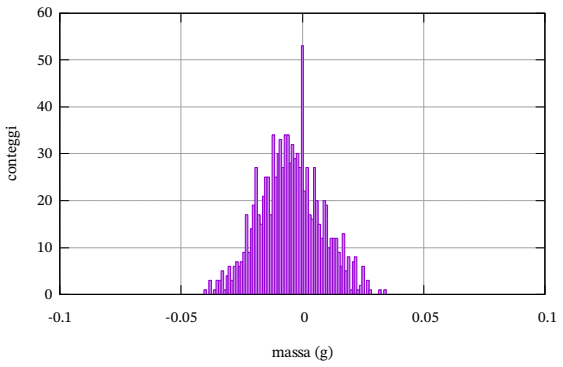
La prima cosa che ho fatto è stato prendere, con la cella non sottoposta a carico, mille misure di seguito e sulla loro media ho determinato lo "shift" prima indicato: l'unità di misura è in grammi. Ho poi fatto mille misure il cui istogramma è il primo che comparire qui di seguito. Possiamo osservare che la dispersione ha l'andamento a gaussiana che ci aspetteremmo. La misura non è centrata sullo zero, ma le misure sono tutte comprese tra più o meno mezzo decimo di grammo.

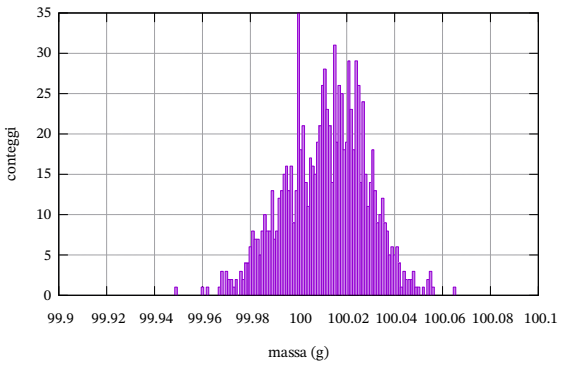
Ho poi applicato una massa da 100 g e completato la taratura come prima illustrato. Ho fatto altre mille misure il cui istogramma compare nel secondo grafico. Il testo molto piccolo è dettato dalla necessità di mostrare le misure in ascissa che anche in questo caso sono risultate piuttosto centrate sul valore atteso e con una dispersione analoga.

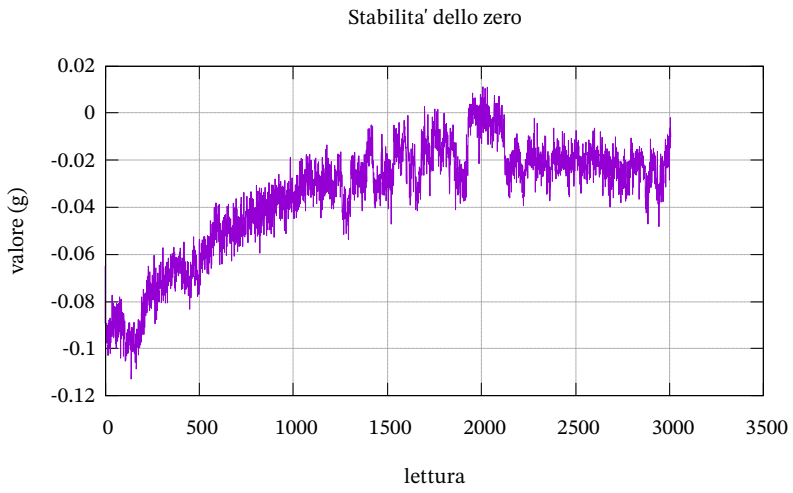
La domanda ora è quanto stabili siano i parametri così ottenuti a distanza di tempo. La risposta è: non quanto si vorrebbe. Lo possiamo osservare con il terzo grafico in cui, partendo da un sensore non usato da alcune ore, ho preso ogni secondo la media su dieci misure di seguito e questo per circa un'ora. Possiamo osservare che il sensore raggiunge una certa stabilità dopo quasi mezz'ora. Non è la condizione ideale per operare in laboratorio, ma è altresì vero che dal valore iniziale si ha uno scostamento che non supera un decimo di grammo.

La dispersione osservata è di circa 0,05 g, e l'accuratezza complessiva è circa 0,1 g.

Il codice per i grafici si trova al capitolo [C].







Successivamente, per testare anche la linearità della cella, ho misurato alcuni pesetti di limitata qualità, sia con un bilancino di precisione con una portata di 100 g che con la cella di carico.

valore nominale (g)	cella (g)	σ (g)	bilancino (g)
0	0,04	0,02	0,00
100	100,00	0,04	99,97
0	0,05	0,01	0,00
50	50,04	0,05	49,98
20	20,11	0,01	19,98
10	10,11	0,02	9,98

Parte III

Esperimenti

13 Studio della caduta di un grave

13.1 Introduzione

Un corpo in caduta libera, trascurando l'attrito dell'aria, è sottoposto alla sola forza di gravità: il suo moto è uniformemente accelerato. In un sistema di riferimento orientato verso il basso, in cui supponiamo che il moto parta dall'altezza $s_0 = 0\,\mathrm{m}$ al tempo $t = 0\,\mathrm{s}$, legge del moto è:

$$S = \frac{1}{2}gt^2 \tag{13.1}$$

L'obbiettivo dell'esperienza è quello di fare cadere un grave da diverse altezze, misurando lo spazio percorso e il tempo impiegato, per verificare che l'accelerazione risultante è costante ed è quella di gravità.

Nella nostra esperienza il grave è inizialmente sospeso e tenuto fermo ad un elettromagnete. Quando si pigia un apposito pulsante l'elettromagnete viene spento, il corpo comincia a cadere e si segna il tempo di partenza. Sotto il grave, ad opportuna distanza, una fotocellula rileva il passaggio del grave in caduta, e viene misurato l'istante in cui avviene il passaggio. Il tempo di caduta è la differenza tra i due istanti misurati.

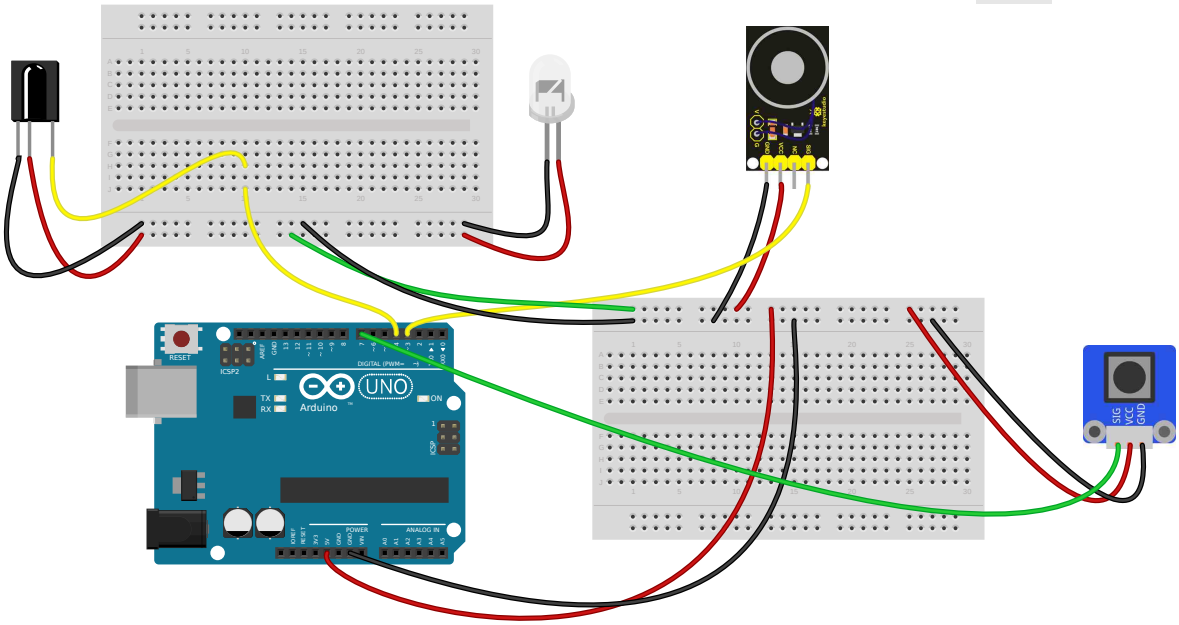
13.2 Materiale utilizzato

- grave (biglia di acciaio)
- Una scheda Arduino (uno)
- Breadboard
- IR break beam sensor - 3mm LED - Adafruit - Portata 25 cm - Tempo di risposta $< 2\,\mathrm{ms}$
- Elettromagnete (KS0320 Keyestudio Electromagnet Module)
- Interruttore a pulsante (sunfounder)
- Cavetteria varia
- Supporto per i sensori
- Metro a nastro - Portata 5 m - accuratezza 1 mm

13.3 Montaggio delle apparecchiature

1. *Collegamento degli apparecchi alla scheda Arduino*
   La nostra scheda Arduino deve essere collegata a quattro congegni: un pulsate, l'elettromagnete e la fotocellula a barriera, che è costituita da un led ad infrarossi e un sensore. Tutti e quattro i dispositivi hanno un ingresso che va messo a terra e uno alla tensione di 5 V. I rispettivi cavi sono evidenziati in nero e rosso nel disegno successivo. Inoltre pulsante, l'elettromagnete e led ad infrarossi hanno una linea su cui mandare o ricevere segnali.

Nella realizzazione con cui abbiamo scritto questo documento, Arduino, pulsante ed elettromagnete erano vicini tra loro, congiunti attraverso una breadboard e poggiati su una tavola. La fotocellula era posizionata sul pavimento e collegata ad una sua propria breadboard: questa serviva da ponte, con dei lunghi fili, all'altra breadboard. Questo è il senso della disposizione che si osserva nel disegno successivo.



In una realizzazione di laboratorio a scuola è probabilmente preferibile collegare su un piano il pulsante e la fotocellula e invece a distanza (su un supporto rialzato) l'elettromagnete.

2. *Codice per Arduino*
   La logica a cui sottende il codice è questa:
   - Si pigia il tasto e si attiva l'elettromagnete e si accende il led di Arduino (riga 29).
   - Si pigia il tasto e si spegne l'elettromagnete, si spegne il led di Arduino e si segna il tempo di inizio della caduta (riga 39).
   - Quando il grave passa attraverso la fotocellula si segna il tempo trascorso dall'inizio della caduta (riga 48).

   Questa sequenza può essere ripetuta quante volte si vuole. Appena accesa la scheda è opportuno pigiare più volte il tasto in modo da portarsi all'inizio della sequenza descritta. Inoltre, dopo che la fotocellula segna il tempo di arrivo, il sistema continua a visualizzare l'istante in cui qualcosa passa attraverso: questo è un limite del codice, ma non pone nessun problema se non estetico.

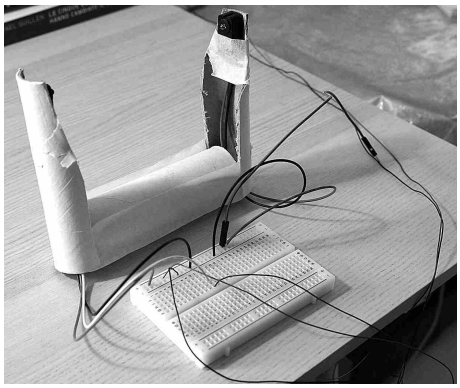
```
1  //*** caduta di un grave ***
   const int ledPin = 13;
3  const int pulsantePin = 7;
   const int barrieraPin = 4;
5  const int SolenoidPin = 3;

7  int misura = 0;
   int stato_pulsante_prec = 0;
9  float tempo_i = 0;
   float tempo_f = 0;
```

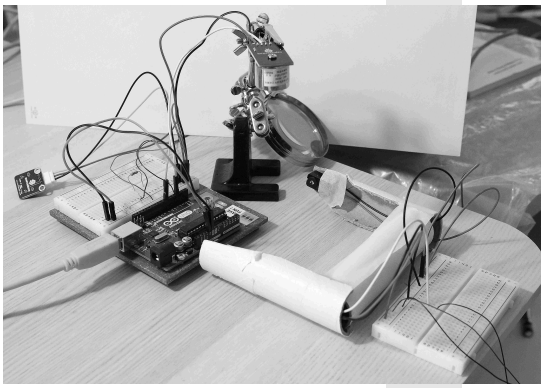
```
11
12 void setup()
13 {
14   pinMode(ledPin, OUTPUT);
15   pinMode(pulsantePin, INPUT);
16   pinMode(SolenoidPin, OUTPUT);
17   pinMode(barrieraPin, INPUT);
18   digitalWrite(barrieraPin, HIGH);
19   Serial.begin (56000);
20 }
21 void loop()
22 {
23   boolean stato_pulsante = digitalRead(pulsantePin);
24   boolean stato_barriera = digitalRead(barrieraPin);
25
26   if (stato_pulsante == LOW) {
27   stato_pulsante_prec = 0;
28   }
29   if (stato_pulsante == HIGH && misura == 1 && stato_pulsante_prec == 0){
30     Se rial.println ("␣");
31     Serial.println ("␣---␣Elettromagnete␣attivo␣---");
32     Serial.println ("␣---␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣---");
33     Serial.println ("␣---␣␣␣␣␣␣␣␣Pronti␣␣␣␣␣␣␣␣␣␣---");
34     digitalWrite(ledPin, HIGH);
35     digitalWrite(SolenoidPin, HIGH);
36     misura = 0;
37     stato_pulsante_prec = 1;
38   }
39   if (stato_pulsante == HIGH && misura == 0 && stato_pulsante_prec == 0){
40     digitalWrite(SolenoidPin, LOW);
41     digitalWrite(ledPin, LOW);
42     tempo_i = micros();
43     misura = 1;
44     stato_pulsante_prec = 1;
45     Serial.println ("␣---␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣---");
46     Serial.println ("␣---␣␣␣inizio␣conteggio␣␣␣␣---");
47   }
48   if (stato_barriera == LOW && misura == 1){
49     tempo_f = micros();
50     Serial.println ("␣---␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣---");
51     Serial.print ("␣␣␣fine␣conteggio:␣");
52     Serial.print ((tempo_f-tempo_i)/1000000,4);
53     Serial.println ("␣s");
54     delay(1000);
55   }
56 }
```

- riga 2-5: indicazione dei pin a cui colleghiamo i vari dispositivi.
- riga 7: la variabile *misura* indica se siamo nel tempo di caduta del grave.
- riga 8: la variabile *stato_pulsante_prec* indica lo stato del pulsante prima dell'ultima lettura.
- riga 14-17: settiamo i vari pin in modalità input o output.
- riga 18: accendiamo la resistenza di pull-up associata al pin del sensore a fotocellula.

- riga 23-24: leggiamo lo stato del pulsante e della fotocellula.
- riga 26: se il pulsante è nello stato ”LOW”, cioè non schiacciato, segno che era nello stato di non schiacciato.
- riga 29: se il pulsante è nello stato ”HIGH”, cioè schiacciato, e *stavo* facendo una misura e lo stato precedente del pulsante era non schiacciato allora ...
- riga 34-35: attivo il led di Arduino e accendo il magnete.
- riga 36: sono nello stato di misura pronta a partire.
- riga 37: segno che lo stato precedente del pulsante era schiacciato.
-
- riga 39: se il pulsante è nello stato ”HIGH”, cioè schiacciato, e *non stavo* facendo una misura e lo stato precedente del pulsante era non schiacciato allora ...
- riga 40-41: spengo il led di Arduino e il magnete.
- riga 42: segno il tempo iniziale in microsecondi.
- riga 43: sono nello stato di misura avviata.
- riga 44: segno che lo stato precedente del pulsante era schiacciato.
-
- riga 48: se la fotocellula è nello stato ”LOW”, cioè c’è un oggetto che intercetta il fascio, e stavo facendo una misura allora ...
- riga 49: leggo il tempo finale in microsecondi.
- riga 52: calcolo e stampo il tempo di caduta in secondi con quattro cifre.
- riga 54: aspetto un secondo prima di fare qualsiasi altra cosa.

3. *Posizionamento del sensore su un supporto*
   Il sensore e il LED ad infrarossi vanno montati su un supporto a U in modo che il grave vi passi attraverso in modo da intercettare il fascio di infrarossi. La scelta delle distanze deve essere tale da far passare il grave in sicurezza, senza sbattere sul dispositivo, ma non troppo distanti per non rendere la rilevazione del passaggio inaffidabile (consigliamo circa 10 cm).
   Un possibile modello semplicissimo e autoesplicativo può essere quello illustrato nella prima foto seguente: è importante tenere il sensore e il LED ben allineati, sia per valutare correttamente il punto di arrivo del grave, sia per rendere efficiente il rilevamento anche con passaggi molto veloci.
   Nell’altra foto un’immagine di tutto il dispositivo nella sua interezza.





13.4 Procedimento

- *Allineamento magnete sensore*
  Schiacciamo più volte il pulsante finché nel monitor seriale di Arduino non compare la scritta che ci dice che siamo pronti e il led di Arduino si illumina. Attacchiamo al magnete il grave e schiacciamo il pulsante per farlo cadere. Osserviamo in che punto della superficie di appoggio è caduto il grave e posizioniamo il sensore in linea.
- *Misura del tempo*
  Ora ripetiamo la sequenza leggendo il tempo di caduta. Ripetiamo anche cinque volte la caduta, sia per assicurarci che il tempo sia uniforme, sia per prendere come misura la media aritmetica dei valori misurati.
- *Misura della lunghezza*
  In una fase iniziale di questa esperienza ho considerato come distanza percorsa dal grave quella che intercorre tra la parte frontale della biglia metallica che ho usato come grave e la linea del fascio della fotocellula. Misurando le distanze in questo modo ho ottenuto valori di g decisamente sottostimati, anche considerando l'errore associato. In realtà bisogna considerare che il sensore ha un tempo di risposta di 1 o 2 ms e non si accorge della presenza del grave nell'istante in cui il grave comincia ad interrompere il fascio, ma solo dopo. Possiamo fare anche la seguente considerazione cinematica.
  Il tempo di caduta, già alla distanza di mezzo metro, è circa tre decimi di secondo.
  La velocità della biglia è quindi:

$$v = gt = 9{,}8\,\mathrm{m/s^2} \cdot 0{,}3\,\mathrm{s} \simeq 3\,\mathrm{m/s} \tag{13.2}$$

  La biglia ha un diametro di 1 cm. Questa distanza, a quella velocità, viene coperta in un tempo di:

$$t = \frac{S}{v} = \frac{0{,}01\,\mathrm{m}}{3\,\mathrm{m/s}} = 0{,}003\,\mathrm{s} \tag{13.3}$$

  Questo tempo è già analogo al tempo di reazione del sensore: il sensore "vede" la biglia quando è passata completamente.
  In conclusione, per lo meno per distanze superiori a mezzo metro, la distanza percorsa va dal magnete (la coda della biglia nell'istante iniziale) al sensore (la coda della biglia quando incomincia ad essere rilevata).

13.5 Dati sperimentali

Abbiamo compiuto tre o più misure per ogni altezza per cinque altezze diverse. Le misure di tempo hanno oscillato di una unità sull'ultima cifra riportata: non ho però svolto una media rigorosa su questi valori per la difficoltà di ripetere misure successive dalla stessa identica altezza.

	Distanza (m)	Tempo (s)
1	1,105	0,476
2	0,885	0,427
3	0,800	0,406
4	0,545	0,335
5	0,255	0,229

13.6 Elaborazione dati sperimentali

Calcoliamo l'accelerazione corrispondente ai dati raccolti e l'errore associato:

$$g = \frac{2S}{t^2} \qquad \Delta g = g\left(\frac{\Delta S}{S} + 2\frac{\Delta t}{t}\right) \tag{13.4}$$

Come valutazione dell'errore sulle lunghezze ho usato 5 mm per la difficoltà di posizionare il metro con maggiore precisione. Per le misure di tempo ho già scritto che le fluttuazioni sulle misure sono dell'ordine del millesimo di secondo. Il tempo di risposta del sensore è indicato come minore di due millesimi. Per cautela ho fatto i calcoli considerando un'accuratezza di 2 ms.

	g (m/s^2)	errore (m/s^2)
1	9,8	0,1
2	9,7	0,2
3	9,8	0,2
4	9,7	0,2
5	9,7	0,4

13.7 Conclusioni

Le accelerazioni riportate sono conformi al valore usuale dell'accelerazione di gravità e, nei limiti degli errori sperimentali, sono costanti al variare dell'altezza di caduta. L'errore riportato è legato soprattutto all'imprecisione sulla misura delle lunghezze: questo errore si può facilmente diminuire con i supporti presenti in un normale laboratorio di fisica.

14 Studio del moto di un pendolo

14.1 Il pendolo

Il pendolo semplice è costituito da un grave, idealmente puntiforme, appeso ad un filo di massa trascurabile e inestensibile. Il filo è appeso ad un vincolo fisso ed è libero di muoversi su di esso senza attrito. Il corpo deve essere piccolo rispetto alla lunghezza del filo, altrimenti le forze che agiscono su di esso non sarebbero assimilabili a quelle che agiscono su un corpo puntiforme. Il filo deve essere di massa trascurabile, altrimenti dovremmo considerare anche la forza peso che agisce su ogni suo elemento; deve essere anche inestensibile in modo da avere una traiettoria su un arco di circonferenza. In queste condizioni ideali il grave è soggetto a due forze o tre forze: se è fermo la sua forza peso e la reazione vincolare del filo che lo sostiene; se in movimento anche la forza centripeta per mantenere una traiettoria circolare.

14.1.1 Isocronismo del pendolo

Se l'angolo di oscillazione è sufficientemente piccolo il periodo di oscillazione può essere ottenuto dalla relazione:

$$T = 2\pi\sqrt{\frac{l}{g}} \tag{14.1}$$

Cosa significa sufficientemente piccolo? Tutto dipende da quale accuratezza vogliamo raggiungere con le nostre misure. Se vogliamo che la formula non dia un risultato che si discosti da quello reale per più del 1% allora l'angolo di oscillazione deve essere inferiore a 23°; se vogliamo un errore inferiore a 0,1% allora l'angolo di oscillazione deve essere inferiore a 7°: più piccolo l'angolo inferiore l'errore. Se vale la relazione precedente, questo periodo non dipende dall'angolo di oscillazione e quindi è costante. Si parla di conseguenza di *isocronismo* del pendolo.

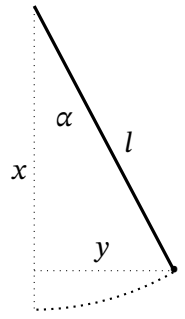
La soluzione esatta del modello del pendolo semplice ci porta alla seguente espressione per il periodo:

$$T = 2\pi\sqrt{\frac{l}{g}}\left[1 + \frac{1}{4}\sin^2\left(\frac{\alpha}{2}\right) + ...\right] \tag{14.2}$$

In questo modello il periodo è dipendente dall'angolo di oscillazione α.
Per poter misurare l'angolo di oscillazione possiamo misurare la lunghezza del segmento y, data dalla distanza del corpo oscillante dall'asse di simmetria della traiettoria del pendolo.
L'angolo α del triangolo rettangolo che ha la lunghezza del pendolo come ipotenusa è:

$$\alpha = \arctan\frac{y}{x} \simeq \arctan\frac{y}{l} \tag{14.3}$$

dove possiamo scrivere l'ultima relazione per una valutazione comunque attendibile dell'angolo.



14.2 Verifica sperimentale dell'isocronismo del pendolo

In questo primo esperimento *misuriamo il tempo di oscillazione del pendolo e la sua stabilità nel tempo*. Se vogliamo studiare sperimentalmente questo fenomeno abbiamo bisogno solo di misurare il periodo impiegato dal pendolo per compiere un'oscillazione completa: non siamo interessati a seguire l'oscillazione nel suo complesso. Misuriamo il tempo impiegato dal grave per passare due volte di seguito in una qualsiasi posizione. I sensori di presenza permettono di determinare con grande precisione l'istante in cui un oggetto passa davanti al sensore: seguiremo questa strada per la nostra misurazione.

14.2.1 Materiale utilizzato

- Pendolo
- Una scheda Arduino (uno)
- Breadboard
- IR break beam sensor - 3mm LED - Adafruit - Portata 25 cm - Tempo di risposta < 2 ms
- Cavetteria varia
- Supporto per i sensori
- Metro a nastro - Portata 5 m - accuratezza 1 mm

14.2.2 Montaggio delle apparecchiature

Per quanto riguarda la scelta del pendolo ho sperimentato in questa esperienza diversi problemi: gli espongo nello note finali. Qui mi limito a dire che il pendolo deve avere un filo inestensibile, di massa trascurabile rispetto al grave ad esso appeso (almeno cento volte più leggero); il grave deve essere simmetrico (cilindrico), preferibilmente opaco, non troppo largo.
Nel mio caso ho usato come filo un filo interdentale. Ad esso ho legato, su dieci centimetri di spago per avere un nodo abbastanza grande alla base, alcuni cilindretti metallici dotati di foro centrale. I cilindri sono stati poi avvolti con del nastro adesivo di carta, per avere maggior compattezza e uniformità ottica.

Per quanto riguarda la realizzazione e messa in opera del sensore di posizione dobbiamo affrontare tre questioni:

1. *Collegamento del sensore ad una scheda Arduino*
   Per questo aspetto rimandiamo al capitolo sui sensori di presenza dove illustriamo come fare i collegamenti.

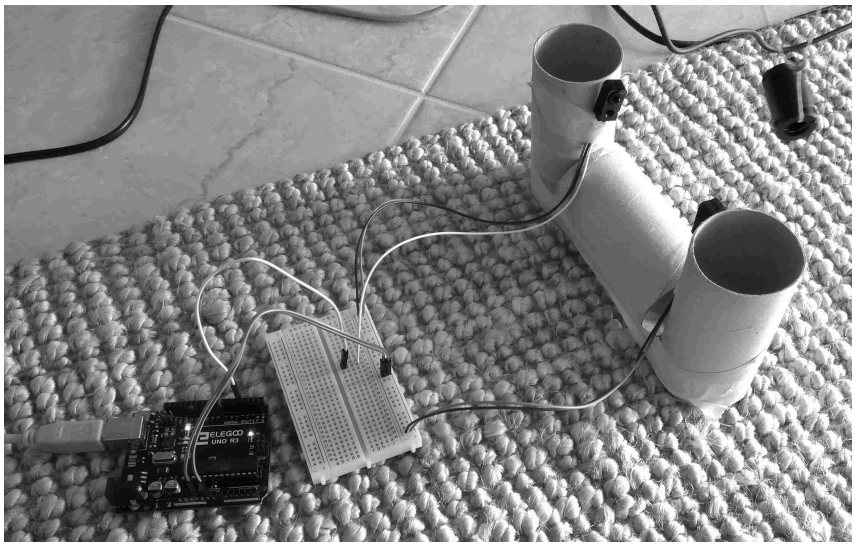
2. *Codice per arduino*
   Il sensore non serve a niente se non ci consente di fare delle misurazioni. Il codice qui presente permette di determinare il periodo di oscillazione del pendolo. Come illustreremo nel procedimento, posizioniamo il sensore nella posizione di equilibrio del pendolo, là dove il grave farà tre passaggi per ogni oscillazione completa. Nel primo passaggio, appena Arduino è avviato, il periodo non è valutato correttamente. Nei successivi passaggi il tempo finale di una oscillazione diventa il tempo iniziale della successiva. Quindi, dei tre passaggi per una oscillazione completa, il terzo è in comune con l'oscillazione successiva.

```
//Misura del periodo di un pendolo
//con un IR break beam
const int sensorPin = 7;
int contatore = 1;
int libero = 0;
int misura = 0;
unsigned long tempo_iniziale = 0;
unsigned long tempo_finale = 0;
void setup()
{
  pinMode(sensorPin, INPUT);
  digitalWrite(sensorPin, HIGH);
  Serial.begin (56000);
}
void loop()
{
  boolean sensorVal = digitalRead(sensorPin);
  if (sensorVal == LOW)
  {
    libero = 1;
  }
  if (sensorVal == HIGH && contatore == 1 && libero == 1) {
    contatore = 2;
    libero = 0;
  }
  if (sensorVal == HIGH && contatore == 2 && libero == 1) {
    tempo_finale = micros();
    Serial.print ("␣-␣II␣-␣Periodo␣oscillazione:␣");
    Serial.print ((tempo_finale-tempo_iniziale)/1000.0,2);
    Serial.println ("␣millisecondi");
    //
    tempo_iniziale = tempo_finale ;
    libero = 0;
    contatore = 1;
    misura = misura + 1;
    Serial.print (misura);
    Serial.print ("␣-␣I");
  }
}
```

- riga 3: stiamo collegando il cavo bianco del sensore al pin 7
- riga 4: la variabile *contatore* vale 1 se il pendolo è passato una prima volta sul sensore.
- riga 11: il pin del sensore è in modalità input
- riga 12: accendiamo la resistenza di pull-up posta nel sensore
- riga 17: leggiamo lo stato del sensore
- riga 18: se il sensore indica ”LOW” il fascio non è interrotto
- riga 19: con il sensore che non segnala nessun passaggio poniamo la variabile ”libero” nello stato 1.
- riga 22: se invece il *sensore segnala un ostacolo* e era nello *stato di ”libero”* e il *contatore di passaggi è a* ***uno***, allora:
- riga 23: ho fatto il secondo passaggio dell'oggetto davanti al sensore

- riga 24: il sensore è ora nello stato occupato
- riga 26: se invece il *sensore segnala un ostacolo* e era nello *stato di "libero"* e il *contatore di passaggi è a* ***due***, allora:
- riga 27: leggo il tempo in microsecondi
- riga 29: calcolo e stampo il periodo in millisecondi
- riga 32: il tempo finale di una oscillazione è il tempo iniziale della successiva
- riga 33: il sensore è nuovamente libero
- riga 34: il contatore vale uno perché è iniziato il primo passaggio della nuova oscillazione
- riga 35: la variabile *misura* conta il numero di oscillazione e viene incrementato di uno

3. *Posizionamento del sensore su un supporto*
   Il sensore e il LED ad infrarossi vanno montati su un supporto a U in modo che la massa appesa al pendolo vi passi attraverso in modo da intercettare il fascio di infrarossi. La scelta delle distanze deve essere tale da far muovere il grave agevolmente, senza sbattere sul dispositivo, ma non troppo distanti per non rendere la rilevazione del passaggio inaffidabile (consigliamo circa 10 cm). Un possibile modello semplicissimo e autoesplicativo può essere quello illustrato nella foto seguente: è importante tenere il sensore e il LED ben allineati.



4. *Posizionamento del supporto*
   Dopo aver realizzato il supporto lo posizioniamo in corrispondenza del punto di oscillazione più basso ovvero quello del grave nella sua posizione di equilibrio stabile, quando rimane fermo.

14.2.3 Procedimento

Quale è la posizione migliore per il sensore? Il pendolo segue una traiettoria in cui alla massima oscillazione corrisponde una velocità nulla e alla posizione di equilibrio (quando il pendolo è a riposo) corrisponde la massima velocità. In questo secondo punto il passaggio avviene nell'intervallo più breve: questo è il punto che scegliamo per la misurazione.

Scegliamo come lunghezza del filo e come posizione del punto di sospensione un valore tale da avere il grave in corrispondenza dell'altezza del fascio di infrarossi del sensore quando il grave è ancora fermo. Oppure, a preferenza, prima costruiamo il filo con il grave e poi collochiamo il sensore all'altezza giusta in corrispondenza del grave.

Mettiamo in oscillazione il pendolo. Il modo che consiglio è allontanarlo dal punto più basso quando è completamente fermo, tenendolo su un piano perpendicolare all'asse del fascio della fotocellula, per poi lasciarlo andare senza spingerlo. Scegliamo una angolo di oscillazione non troppo ampio, ma ampio abbastanza affinché il pendolo possa compiere alcune centinaia di oscillazioni durante le misurazioni.

Per avere una valutazione dell'angolo di oscillazione con un righello misuriamo la distanza del grave quando raggiunge la massima oscillazione dall'asse del pendolo nella sua traiettoria, ovvero la grandezza y indicata nella prima figura del capitolo.

Ora possiamo fare le misurazioni. Prendiamo i primi cinque periodi di oscillazione e gli ultimi cinque quando l'ampiezza di oscillazione è di pochi centimetri. Se il grave è abbastanza piccolo e simmetrico le variazioni del periodo, tra un oscillazione e l'altra, sono contenute entro il millesimo di millimetro. Possiamo ora valutare di quanto è variato il periodo.

Per valutare la lunghezza del pendolo misuriamo la distanza del punto di sospensione del filo dal baricentro del grave utilizzato.

14.2.4 Dati sperimentali

Riporto qui di seguito le prime cinque letture osservate e le ultime cinque dopo 200 oscillazioni complete.

	Periodo iniziale (s)	Periodo finale (s)
1	2,9606	2,9510
2	2,9603	2,9510
3	2,9604	2,9510
4	2,9599	2,9510
5	2,9594	2,9510

L'angolo di oscillazione era circa 12° all'inizio dell'oscillazione e meno di 3° alla fine.

La lunghezza del pendolo usato è $l = 2{,}160 \pm 0{,}005$ m. L'accuratezza di mezzo centimetro è legata alla difficoltà di effettuare da soli la misura, allineando al millimetro il punto iniziale e finale del nastro millimetrato. Il filo era un sottile filo interdentale, di massa trascurabile. Il grave (costituito da dei cilindretti metallici) aveva una massa di circa 42 g.

14.2.5 Elaborazione dati sperimentali

Cominciamo col calcolare il valore atteso per il periodo secondo la formula sia con l'angolo di oscillazione iniziale che con quello finale, usando il valore dell'accelerazione di gravità della località in cui ci troviamo: nel mio caso $g = 9{,}80269\ \mathrm{m/s^2}$.

Otteniamo $T_i = 2{,}958$ s e $T_f = 2{,}951$ s, valori in linea con quelli ottenuti.

Il periodo è soggetto a fluttuazioni. Per quanto riguarda la valutazione dell'errore sul periodo osserviamo che i dati del sensore non riportano l'accuratezza o precisione temporale raggiungibile; si parla solo di tempo di risposta. Questo valore, inferiore a 2 ms, è un errore sistematico per difetto, poiché l'istante di passaggio viene indicato solo dopo l'evento stesso. Tuttavia è anche vero che l'oggetto, nel primo e terzo passaggio, si trova ad attraversare il sensore sempre nella stessa posizione e con lo stesso verso e direzione del moto: possiamo immaginare che il ritardo di risposta sia del tutto simile e invece l'intervallo di tempo tra i due istanti più accurato.

Per tutte queste ragioni valutiamo come errore di misura 1 ms.

Osserviamo che le fluttuazioni sono contenute nell'errore sperimentale che abbiamo valutato. Il periodo non è costante, come atteso dalla teoria, ma la sua variazione è contenuta entro pochi millisecondi tra l'ampiezza più grande e più piccola in cui è avvenuta la misurazione.

14.2.6 Conclusioni

Il periodo del pendolo, con il grado di accuratezza raggiunto, non si mantiene costante nel tempo, con una sostanziale concordanza con il periodo indicato anche dai modelli più accurati.

14.3 Misura dell'accelerazione di gravità

Il periodo di oscillazione è legato all'accelerazione di gravità. Conoscendo la lunghezza l del filo e misurando il periodo di oscillazione possiamo ricavare l'accelerazione di gravità. Ricaviamo il periodo così come indicato nella precedente esperienza in regime di piccola oscillazione, cioè quando la nota formula per il periodo è più corrispondente ai dati reali.

L'accelerazione di gravità nella città in cui mi trovo è tabulata come $g = 9{,}80269\,\mathrm{m/s^2}$.

Dal periodo del pendolo ricaviamo la seguente formula:

$$g = \frac{4\pi^2 l}{T^2} \tag{14.4}$$

L'errore associato a g è:

$$\Delta g = g\left(\frac{\Delta l}{l} + 2\frac{\Delta T}{T}\right) \tag{14.5}$$

Infine, con i dati sperimentali che ho prima riportato, otteniamo $g_s = 9{,}79 \pm 0{,}03\,\mathrm{m}$, in buon accordo con il risultato atteso.

14.4 Note

Questa esperienza si è dimostra molto più sensibile a diversi aspetti sperimentali di quanto avessi immaginato. Qui di seguito espongo alcuni di questi problemi cercando di dare ragione di alcune scelte effettuate.

Fluttuazioni nel periodo

Il grave che ho utilizzato nelle prime prove non era simmetrico: ottenevo delle fluttuazioni di 1 ms tra una oscillazione e la successiva. Poi ho trovato i cilindretti metallici mostrati nella figura precedente: il periodo ha cominciato a manifestare fluttuazioni sui 0,2 ms. Osservando il periodo quando le oscillazioni tendono ad esaurirsi ho trovato che esso tendeva a risalire dopo una fase di minimo. Allora ho fasciato i cilindretti con del nastro adesivo di carta, per renderli più coesi. Non so se per questo effetto o per il fatto che la superficie del grave fosse ora più uniforme e opaca, ma questo ha stabilizzato anche la fase finale delle oscillazioni.

Massa del grave

Si dice sempre che il pendolo ha un periodo di oscillazione che non dipende dalla massa del grave. Non si sottolinea abbastanza il fatto che questo è vero per il pendolo semplice e quindi per un oggetto del tutto ideale; diverso è il caso del pendolo fisico. Nelle prime prove svolte ho usato un filo di corda da pacchi sottile, con un oggetto di qualche decina di grammi appeso. I risultati sperimentali non erano soddisfacenti. Ho poi verificato che la massa del filo era di 4 g, quindi ben un decimo di quella del corpo: una situazione del tutto non ideale. Per questa ragione sono passato al filo interdentale realizzando un rapporto tra massa del filo e quella del grave inferiore a uno su cento.

Lunghezza del pendolo

Se il grave appeso al filo fosse puntiforme non ci sarebbe problema, ma il grave (anche per avere una massa sufficientemente più alta di quella del filo) ha una estensione di alcuni centimetri. Ho scelto di misurare la distanza tra il baricentro del grave e il punto di sospensione del filo.

Luce ambientale e sensore

Il sensore della fotocellula è abbastanza sensibile alla luce ambientale. Se vi arrivano i raggi del Sole rimane abbagliato, ma anche una stanza troppo illuminata lo manda in crisi con risultati sballati e comportamenti imprevedibili. Consiglio di lavorare in una stanza con poca illuminazione (un aula scolastica con le luci spente di solito va benissimo!).

15 Studio delle forze su un pendolo

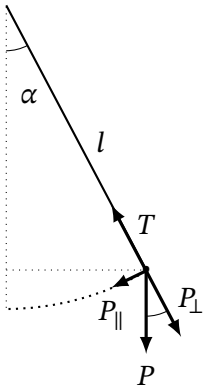
15.1 Il pendolo

Come già esposto nel capitolo precedente il pendolo semplice è costituito da un grave, idealmente puntiforme, appeso ad un filo di massa trascurabile e inestensibile. Il filo è appeso ad un vincolo fisso ed è libero di muoversi su di esso senza attrito. Il corpo deve essere piccolo rispetto alla lunghezza del filo, altrimenti le forze che agiscono su di esso non sarebbero assimilabili a quelle che agiscono su un corpo puntiforme. Il filo deve essere di massa trascurabile, altrimenti dovremmo considerare anche la forza peso che agisce su ogni suo elemento; deve essere anche inestensibile in modo da avere una traiettoria su un arco di circonferenza.

15.2 Le forze che agiscono sul pendolo

In queste condizioni ideali il grave è soggetto a due o tre forze: se è fermo la sua forza peso e la reazione vincolare del filo che lo sostiene; se in movimento anche la forza centripeta per mantenere una traiettoria circolare.

Nella figura affianco vediamo come la forza peso si scomponga in una componente parallela alla traiettoria, che determina il moto, e una perpendicolare, che tiene in tensione il filo. Le forze rappresentate sono le sole presenti solo quando il corpo si trova fermo nel punto di massima altezza. Altrimenti è presente anche una forza centripeta per mantenere la traiettoria circolare.



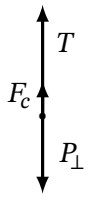
Nel punto di massima altezza la tensione del filo è uguale alla forza peso perpendicolare.

$$\vec{T} + \vec{P}_\perp = 0 \tag{15.1}$$

Tensione e componente del peso sono uguali e contrari.
In modulo possiamo scrivere:

$$T = mg\cos\alpha \tag{15.2}$$

Quando invece il corpo raggiunge la minima altezza abbiamo anche la forza centripeta alla sua massima intensità.



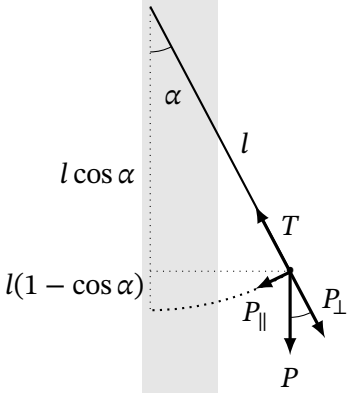
$$\vec{T} + \vec{P}_\perp = \vec{F}_c \tag{15.3}$$

La tensione e la forza centripeta vanno verso l'alto, la componente della forza peso verso il basso.

$$T = mg + m\frac{v^2}{l} \tag{15.4}$$

Nell'espressione della forza centripeta a denominatore compare il raggio della traiettoria: in questo caso è la lunghezza l del filo. La forza centripeta dipende dal quadrato della velocità che a sua volta raggiunge il massimo nel punto più basso della traiettoria.

Per poter determinare la velocità nel punto più basso della traiettoria applichiamo la conservazione dell'energia al moto del pendolo. Dalla figura possiamo osservare che nel punto di massima altezza il corpo si trova ad una altezza uguale a $l(1-\cos\alpha)$ rispetto al punto più in basso e l'energia è solo potenziale gravitazionale. Nel punto di minima altezza l'energia è solo cinetica. Per cui possiamo scrivere:



$$\begin{aligned} U_p &= E_c \\ mgl(1-\cos\alpha) &= \frac{1}{2}mv^2 \\ v^2 &= 2gl(1-\cos\alpha) \end{aligned} \tag{15.5}$$

Sostituiamo questa espressione nell'espressione della tensione del filo nel punto più basso.

$$T = mg + m\frac{v^2}{l} = mg + m\frac{2gl(1-\cos\alpha)}{l} = mg + 2mg(1-\cos\alpha) \tag{15.6}$$

In conclusione la tensione del filo, rispetto al caso del pendolo fermo e soggetto solo alla forza peso, varia continuamente tra un valore massimo:

$$T_{max} = mg + 2mg(1-\cos\alpha) \tag{15.7}$$

e un valore minimo:

$$T_{min} = mg(\cos\alpha) \tag{15.8}$$

La tensione del filo è uguale alla forza esercita dal filo nel punto di sospensione. Quella forza può essere misurata con una cella di carico che è quel che facciamo in questa esperienza.

15.3 Materiale utilizzato

- pendolo
- una scheda Arduino (uno).
- scheda basata sul HX711
- cella di carico da 1 kg
- base di legno come supporto
- un blocco da 100 g per la taratura
- cavetteria varia
- foglio di cartoncino
- metro a nastro

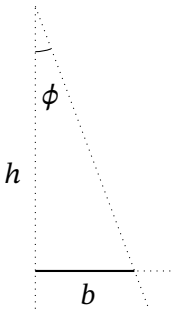
Il materiale utilizzato è quello descritto nel capitolo sulla misura delle forze e sulla cella di carico a cui si aggiunge il pendolo utilizzato nella prima esperienza sul pendolo e inoltre un cartoncino su cui disegnare i valori degli angoli descritti dal pendolo.

15.4 Montaggio delle apparecchiature

Come illustrato nella prima esperienza sul pendolo [14.2.2] quello qui utilizzato è costituito da un filo interdentale a cui ho legato, su dieci centimetri di spago per avere un nodo abbastanza grande alla base, alcuni cilindretti metallici dotati di foro centrale. I cilindri sono stati poi avvolti con del nastro adesivo di carta, per avere maggior compattezza e uniformità ottica.

Inizialmente avevo avvolto il filo interdentale al foro terminale della cella di carico. Successivamente, per le ragioni che illustro nel procedimento, ho preferito far passare una sola volta il filo nel foro, legandolo ad una rondella da appoggiare alla cella. Nella figura mostro la rondella sollevata.

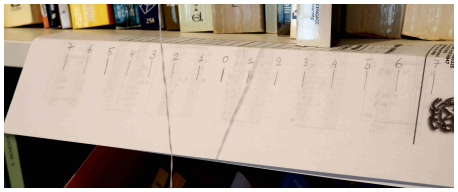
La piastra di legno a cui è fissata la cella è stata posta su un piano di appoggio e fermata con qualche libro posto sopra: può sembrare un fissaggio non sufficientemente stabile, ma certamente lo è maggiormente che non un sostegno posto in cima ad un'asta metallica posta in verticale e poggiata su piano, come è normalmente fatto un pendolo di un laboratorio di fisica. La cella di carico è collegata alla scheda HX711 e ad Arduino come illustrato nel capitolo sulla cella stessa [12.2].



Dietro al filo è stata fissata una striscia di cartone in cui è stato disegnato l'angolo sotteso dal filo. Immaginiamo di avere un triangolo rettangolo di altezza h. L'angolo opposto alla base è ϕ. In un triangolo rettangolo un cateto (la base) è uguale all'altro cateto (l'altezza) per la tangente dell'angolo opposto al primo.

$$b = h \tan \phi \tag{15.9}$$

Al variare di ϕ possiamo segnare sulla base un segno in corrispondenza dell'angolo. Se facciamo questo per per valori successivi dell'angolo (1°, 2°, ...) possiamo segnare dove dovrebbe passare il filo.

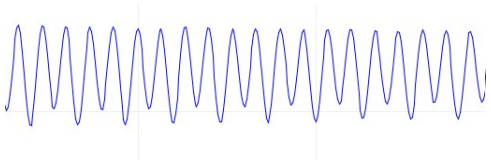


Per quanto riguarda la realizzazione e messa in opera del sensore non ci sono indicazioni particolari se non quelle già scritte sulla cella di carico. Anche il codice che utilizzeremo è lo stesso già riportato nel capitolo sul sensore.

15.5 Procedimento

Prima di collegare il pendolo col suo filo assicuriamoci che la cella sia ben tarata e misuriamo il peso del pendolo senza farlo oscillare. Dopodiché mettiamo in oscillazione il pendolo. Il modo che consiglio è allontanarlo dal punto più basso quando è completamente fermo, tenendolo su un piano perpendicolare alla cella e parallelo alla striscia di cartone con l'indicazione degli angoli, per poi lasciarlo andare senza spingerlo. Scegliamo una angolo di oscillazione non troppo ampio, ma ampio abbastanza affinché il pendolo possa compiere alcune decine di oscillazioni senza variare eccessivamente l'ampiezza di oscillazione.

A meno di non essere particolarmente fortunati il grafico della tensione che otterremo osservando il plotter grafico di Arduino è simile a quello che segue.



In esso possiamo osservare che i massimi sono abbastanza costanti, mentre i minimi si alternano su due valori distinti che si manifestano quando il pendolo oscilla da un parte e dall'altra. La motivazione di questa alternanza credevo fosse associata al modo di legare il filo alla cella di carico, che è il motivo che mi ha fatto poi scegliere il metodo di sospensione del filo prima illustrato. Tuttavia la vera ragione è legata al corretto posizionamento in orizzontale della cella. Infatti se la si regola perfettamente in piano il due minimi spariscono quasi completamente.

Ora che abbiamo fatto, se necessario, questa regolazione facciamo oscillare il pendolo secondo un angolo a nostra scelta e prendiamo le misure relative ad una decina di oscillazioni. Di questi dati prendiamo il massimo e il minimo.

15.6 Dati sperimentali

L'angolo di oscillazione era circa 7°. L'accuratezza della misura della massa ovvero della tensione è circa 0,1 g.

massa del pendolo	Tensione massima	Tensione minima
43,5 g	44,1 g	43,1 g

15.7 Elaborazione dati sperimentali

Confrontiamo i dati sperimentali per la tensione massima e minima con quelli teorici derivanti dall'applicazione delle relazioni prima trovate.

	Tensione massima	Tensione minima
valori sperimentali	44,0 g	43,1 g
valori teorici	44,15 g	43,2 g

15.8 Conclusioni

Il modello teorico proposto e i dati sperimentali sono in buon accordo.

Ci sono ulteriori margini per migliorare l'accuratezza dei dati ottenuti. In particolare:

- La lunghezza del filo non rientra nel valore delle forze in gioco, ma è associata al periodo del pendolo. Un filo più lungo e quindi un periodo più lungo avrebbe consentito di descrivere meglio l'andamento delle forze nel tempo.
- La cella di carico usata ha una portata di 1 Kg. Per la variazione delle forze considerata sarebbe stato meglio usare una cella più sensibile con una portata inferiore. É possibile trovare agevolmente in commercio anche celle da 0,1 Kg, sebbene dal costo superiore.

16 Verifica della legge di Stevino

16.1 Pressione idrostatica

All'interno di un fluido incomprimibile alla profondità h abbiamo una pressione linearmente proporzionale alla profondità e alla pressione dell'eventuale gas presente sopra il fluido: è la legge di Stevino.

$$P = P_0 + dgh \tag{16.1}$$

dove P è la pressione da noi misurata, P_0 è la pressione sul fluido, d la densità del fluido, g l'accelerazione di gravità, h la profondità. Se siamo all'interno di un contenitore aperto P_0 è la pressione atmosferica.

Possiamo misurare direttamente la pressione atmosferica o di un gas con un sensore di pressione per Arduino, come quelli dotati del sensore Bosch BMP180; oppure indirettamente, con un sensore come l'Adafruit MPRLS, mettendo in comunicazione il punto, anche di liquido, in cui vogliamo misurare la pressione con il sensore, attraverso un tubicino chiuso ed impermeabile pieno d'aria in contatto col liquido.

16.2 Descrizione dell'esperienza I

L'obbiettivo dell'esperimento è misurare la pressione all'interno di un contenitore d'acqua, verificando che la pressione misurata segua la legge di Stevino. Non possiamo mettere il sensore direttamente a contatto con l'acqua, ma possiamo usare un tubicino pieno d'aria per mettere in comunicazione il fondo del nostro recipiente con il sensore.

All'interno di un fluido in equilibrio, alla stessa profondità o altezza, la pressione è la stessa: se il sensore e l'estremità del tubicino nel recipiente sono alla stessa altezza la pressione sarà la stessa. In ogni caso, la differenza di pressione dell'aria tra due punti con un dislivello di 30 cm (la profondità massima dell'acqua nella bottiglia che useremo) è di circa 4 Pa, al di sotto della sensibilità effettiva del nostro sensore: anche se l'altezza non è proprio la stessa in questa esperienza non ci sono problemi.

Per misurare la pressione useremo un tubicino. In esso si può manifestare il fenomeno della capillarità: in un tubo sufficientemente sottile e aperto, immerso in un fluido, all'equilibrio il livello del fluido dentro e fuori il tubo può essere diverso. Nel caso dell'acqua che bagna un capillare di vetro il livello del liquido nel capillare tende a salire, tanto più quanto più la sezione interna del capillare è sottile. Esiste un legge che quantifica il fenomeno: la legge di Jurin.

$$h = \frac{2\gamma \cos\theta}{dgr} \tag{16.2}$$

dove h è l'altezza del liquido nel capillare rispetto all'esterno; γ è la tensione superficiale del liquido e d la sua densità; g è l'accelerazione di gravità; θ è l'angolo di raccordo tra la superficie del liquido e la parete del capillare.

Nel nostro caso il capillare è di silicone e non siamo riusciti a trovare in letteratura i parametri specifici per i due materiali in contatto. Dovremo determinare l'h caratteristico per il nostro apparato sperimentalmente.

16.2.1 Apparecchi utilizzati

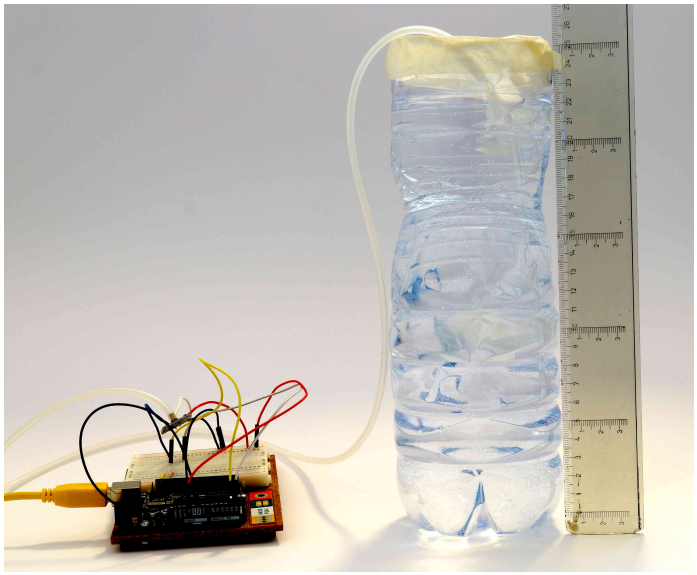
1. scheda Arduino Uno;
2. scheda Adafruit MPRLS;
3. tubicino di silicone, lunghezza 1 m e diametro interno 2 mm;
4. righello millimetrato, lunghezza 50 cm;
5. bottiglia di plastica da due litri;
6. nastro adesivo;
7. cavetteria varia.

16.2.2 Montaggio delle apparecchiature

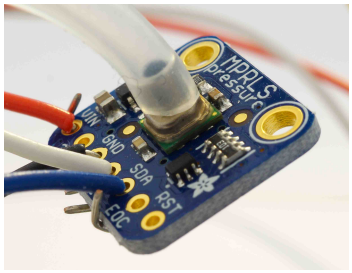
- Prendiamo una bottiglia di plastica da due litri e leviamo l'imboccatura, in modo da avere una accesso più agevole. Facciamo attenzione ai bordi tagliati della bottiglia e passiamoci sopra del nastro adesivo per non tagliarci.
- Colleghiamo il sensore di pressione ad Arduino e questo al computer secondo lo schema indicato nel capitolo "misure di pressione" [66]
- Dal tubo di silicone ne tagliamo un tratto lungo 10 cm. Il tratto rimanente lo colleghiamo al sensore, pronto per essere inserito nella bottiglia.
- Il tratto più corto ci servirà per studiare la capillarità dell'acqua nel tubo: lo teniamo così o lo fissiamo per le estremità sul bordo di un righello, ad esso parallelo, in modo che la parte centrale sia ben visibile.

- Poggeremo la bottiglia su una superficie orizzontale in modo da avere piena visibilità dell'interno della bottiglia e del tubicino. Accanto alla bottiglia in verticale posizioneremo il righello.
- Teniamo a disposizione almeno due litri d'acqua a temperatura ambiente.

Codice per Arduino
Il codice per Arduino è lo stesso utilizzato nel capitolo sulla misura della pressione [66].



(a) apparato montato



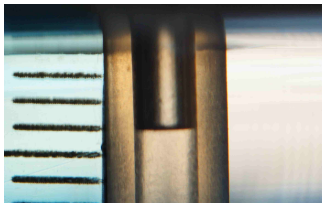
(b) il sensore

16.2.3 Procedimento

1. Immergiamo il tubicino libero da entrambe le estremità in acqua e osserviamo e misuriamo come varia il livello dell'acqua all'interno del capillare. Il dislivello che misuriamo andrà sottratto o aggiunto ai dislivelli che misureremo nei passaggi successivi. Per determinare la variazione di altezza nel tubo per capillarità possiamo anche immergere un righello col tubicino attaccato, in poca acqua, in modo che il livello dell'acqua arrivi al tratto ben visibile del tubo. A quel punto misuriamo di quanto varia il livello dell'acqua nel tubo rispetto all'esterno.
2. Misuriamo la pressione dell'aria prima che l'acqua lambisca il tubicino lungo: è la misura della pressione atmosferica. Aspettiamo almeno cinque minuti dall'avvio di Arduino prima di fare la lettura.
3. Riempiamo di acqua la bottiglia in modo che il tratto superiore sia ben visibile e immergiamo il tubo del sensore per qualche centimetro, facendo in modo che rimanga fermo. Misuriamo col righello l'altezza dell'acqua nel recipiente e il livello dell'acqua nel tubicino. La differenza tra i due valori ci darà la profondità del pelo libero dell'acqua nel tubicino rispetto al pelo libero dell'acqua nella bottiglia. Leggiamo la pressione indicata dal sensore.
4. Immergiamo il tubicino ancora di qualche centimetro e dopo averlo fissato ripetiamo le misure come abbiamo fatto al punto precedente. Ripetiamo questo passaggio per almeno cinque o sei volte.

16.2.4 Dati sperimentali

Chiamiamo x_f la posizione del pelo libero dell'acqua nel recipiente e x_i quella nel tubicino.
La pressione atmosferica è $P_0 = 1017{,}3$ Pa.
L'errore associato alle misure col righello è di 1 mm; per le misure di pressione è 10 Pa.
Come dimostra l'immagine qui accanto il livello dell'acqua nel capillare aperto scende di circa 3 mm: questa misura (che chiamiamo h) andrà sottratta ai dislivelli ottenuti.
L'acqua nei capillari di silicone scende, al contrario di quel che accade nei capillari di vetro.
Osserviamo che l'acqua, nel nostro tubicino, non rimane facilmente sempre alla stessa altezza: a seconda di come spostiamo il capillare il livello può scendere (e rimanere in equilibrio) anche tra i 2 mm e 5 mm.



	x_i (mm)	x_f (mm)	P (hPa)
1	194	221	1019,5
2	178	221	1021,1
3	163	221	1022,6
4	143	221	1024,5
5	98	221	1029,0
6	68	221	1031,7
7	48	221	1033,7

16.2.5 Elaborazione dati sperimentali

Nella seguente tabella riportiamo:

- il dislivello $\Delta x = x_f - x_i$;
- il dislivello corretto con la capillarità $\Delta x_c = \Delta x - h$;
- la pressione idrostatica misurata $P_s = P - P_0$;
- la pressione prevista dalla teoria $P_t = dg\Delta x_c$.

	Δx (mm)	Δx_c (mm)	P_s (hPa)	P_t (hPa)
1	27	24	2,2	2,3
2	43	40	3,8	3,9
3	58	55	5,3	5,4
4	78	75	7,2	7,3
5	123	120	11,7	11,7
6	153	150	14,4	14,7
7	173	170	16,4	16,6

Con i dati a nostra disposizione possiamo provare a calcolare la retta dei minimi quadrati: questo per verificare la linearità dei dati. La retta interpolatrice è $y = mx + q$, dove ad x associamo Δx_c e a y associamo P_s.

m (hPa/mm)	Δm (hPa/mm)	q(hPa)	Δq (hPa)	r
0,0970	0,0005	−0,07	0,06	0,9998

16.2.6 Conclusioni

La concordanza tra misure fatte e dati teorici è molto buona e per quasi tutte le misure è nei limiti degli errori.

Il coefficiente di correlazione della retta interpolatrice ha un valore ottimo. L'intercetta della retta con l'asse y non è nell'origine degli assi come ci dice la teoria. Tuttavia il valore di q (-7 Pa) è inferiore all'errore sulla misura delle pressioni.

La legge prevista dalla teoria è $P_t = dg\Delta x_c = k\Delta x_c$, con $k = dg$: questo k corrisponde al coefficiente angolare della retta interpolatrice. Nel nostro caso, alla temperatura di circa 20 °C, $d = 998\,\mathrm{kg/m^3}$ e $g = 9{,}805\,\mathrm{m/s^2}$, quindi $k = 9{,}79 \times 10^3\,\mathrm{Pa/m} = 0{,}0979\,\mathrm{hPa/mm}$, in buon accordo con i dati sperimentali.

16.2.7 Note

Un valore del dislivello per capillarità di 4 mm invece che i 3 mm utilizzati avrebbe aumentato il grado di concordanza tra dati sperimentali e teorici.

Tra i problemi che permangono ci sono dei possibili errori per parallasse nelle letture delle altezze: il motivo è la forma variegata della bottiglia, tale da non permettere di avvicinare il righello alla bottiglia stessa per tutte le altezze.

Un problema che si è manifestato durante alcune prove è la variazione della pressione atmosferica tra l'inizio e la fine delle misurazioni, con la tendenza ad un maggior discostamento tra valori misurati e valori teorici attesi.

16.3 Descrizione dell'esperienza II

Il problema della capillarità che si è manifestato nella precedente esperienza è stato per me inaspettato e mi ha lasciato insoddisfatto; per cui ho cercato il modo di poter diminuire questo fenomeno sino ad annullarlo completamente, se possibile, per mettere completamente in luce la sola pressione idrostatica, senza altri fenomeni in evidenza. Questa seconda esperienza è un'alternativa leggermente più raffinata alla prima.

Guardiamo ancora la legge di Jurin:

$$h = \frac{2\gamma \cos\theta}{dgr} \tag{16.3}$$

Osserviamo che per ridurre il fenomeno della capillarità possiamo in primo luogo diminuire la tensione superficiale γ: questo può essere realizzato aggiungendo un tensioattivo (del comune detersivo per piatti ad esempio) all'acqua. Con un cucchiaino di detersivo in un litro d'acqua effettivamente l'abbassamento dell'acqua nel tubicino di silicone praticamente si annulla, tuttavia persiste ancora la tendenza dell'acqua a non assumere sempre la stessa altezza all'equilibrio, ciò che rende i risultati sperimentali soggetti ad un andamento aleatorio.

Per ridurre la capillarità possiamo anche allargare il capillare aumentando il suo raggio r. Non possiamo allargare il nostro capillare di silicone, ma possiamo fare in modo che la camera d'aria in cui confluisce il tubicino abbia un diametro più grande. Possiamo ottenere ciò con un bicchiere rovesciato posto sul fondo della bottiglia; il tubicino confluisce in una camera d'aria sulla sommità del bicchiere: in questo modo il pelo libero dell'acqua non ha il raggio del capillare, ma quello del bicchiere.

La legge di Jurin, nel caso dell'acqua che bagna il vetro, si può scrivere così:

$$h = \frac{1{,}48 \times 10^{-5}\,\mathrm{m}^2}{r} \tag{16.4}$$

Da questo ne deriva che la risalita per capillarità nel caso di un raggio di un millimetro vale circa 15 mm, ma nel caso di un raggio di 2 cm (il nostro bicchiere) vale circa 0,7 mm e quindi sotto la sensibilità dei nostri strumenti di misurazione.

16.3.1 Apparecchi utilizzati

1. scheda Arduino Uno;
2. scheda Adafruit MPRLS;
3. tubicino di silicone, lunghezza 1 m e diametro interno 2 mm;
4. righello millimetrato, lunghezza 50 cm;
5. bottiglia di plastica da due litri;
6. bicchiere di vetro, diametro interno 4 cm;
7. nastro adesivo;
8. cavetteria varia.

16.3.2 Montaggio delle apparecchiature

- Prendiamo una bottiglia di plastica da due litri e leviamo l'imboccatura, in modo da avere una accesso più agevole. Facciamo attenzione ai bordi tagliati della bottiglia e passiamoci sopra del nastro adesivo per non tagliarci.
- Colleghiamo il sensore di pressione ad Arduino e questo al computer secondo lo schema indicato nel capitolo "misure di pressione" [66]

- Attacchiamo una estremità del tubo all'interno del bicchiere in modo che l'imboccatura del capillare sia sul fondo del bicchiere (fig. 1).
- Poggeremo la bottiglia su una superficie orizzontale in modo da avere piena visibilità dell'interno della bottiglia e del bicchiere al suo interno. Accanto alla bottiglia in verticale posizioneremo il righello.
- Teniamo a disposizione almeno due litri d'acqua a temperatura ambiente.

Codice per Arduino
Il codice per Arduino è lo stesso utilizzato nel capitolo sulla misura della pressione [66].

(a) fig. 1

(b) fig. 2

(c) fig. 3

16.3.3 Procedimento

1. Misuriamo la pressione dell'aria prima di collegare il tubicino: è la misura della pressione atmosferica. Aspettiamo almeno cinque minuti dall'avvio di Arduino prima di fare la lettura.
2. Versiamo sul fondo della bottiglia un cucchiaino di detersivo: per quanto la capillarità sia molto attenuata questo ci consentirà di avere un tratto più netto del pelo libero dell'acqua e quindi di valutarne più precisamente l'altezza.
3. Immergiamo il bicchiere capovolto dentro la bottiglia (fig. 2), adagiandola sul fondo della bottiglia, facendo attenzione che il tubicino rimanga correttamente attaccato al fondo del bicchiere.
4. Versiamo lentamente l'acqua nella bottiglia finché la camera d'aria nel bicchiere non arriva a circa due centimetri di altezza: il tanto che ci consenta buona visibilità, ma non troppo da far galleggiare il bicchiere (fig. 3).
5. Colleghiamo l'altra estremità del tubo al sensore: a questo punto la camera d'aria nel bicchiere è isolata dall'esterno.
6. Riempiamo di acqua la bottiglia in modo che il tratto superiore sia ben visibile e abbia sommerso di poco il bicchiere. Misuriamo col righello l'altezza dell'acqua nel recipiente e il livello dell'acqua nella camera d'aria. La differenza tra i due valori ci darà la profondità del pelo libero dell'acqua nella camera d'aria rispetto al pelo libero dell'acqua nella bottiglia. Leggiamo la pressione indicata dal sensore.
7. Continuiamo a versare acqua per qualche centimetro e ripetiamo le misure come abbiamo fatto al punto precedente. Ripetiamo questo passaggio per almeno cinque o sei volte.

16.3.4 Dati sperimentali

Chiamiamo x_f la posizione del pelo libero dell'acqua nel recipiente e x_i quella nella camera d'aria nel bicchiere.

La pressione atmosferica è $P_0 = 1012{,}3$ Pa.

L'errore associato alle misure col righello è di 1 mm; per le misure di pressione è 10 Pa.

	x_i (mm)	x_f (mm)	P (hPa)
1	40	77	1015,9
2	40	103	1018,4
3	41	121	1020,0
4	41	134	1021,3
5	41	149	1022,8
6	41	169	1024,7
7	41	189	1026,7

16.3.5 Elaborazione dati sperimentali

Nella seguente tabella riportiamo:

- il dislivello $\Delta x = x_f - x_i$;
- la pressione idrostatica misurata $P_s = P - P_0$;
- la pressione prevista dalla teoria $P_t = dg\Delta x_c$.

	Δx (mm)	P_s (hPa)	P_t (hPa)
1	37	3,6	3,6
2	63	6,1	6,2
3	80	7,7	7,8
4	93	9,0	9,1
5	108	10,5	10,5
6	128	12,4	12,5
7	148	14,4	14,5

Con i dati a nostra disposizione possiamo provare a calcolare la retta dei minimi quadrati per verificare la linearità dei dati. La retta interpolatrice è $y = mx + q$, dove ad x associamo Δx e a y associamo P_s.

m (hPa/mm)	Δm (hPa/mm)	q(hPa)	Δq (hPa)	r
0,0971	0,003	−0,02	0,02	0,99996

16.3.6 Conclusioni

La concordanza tra misure fatte e dati teorici è ottima e per tutte le misure è nei limiti degli errori.

Il coefficiente di correlazione della retta interpolatrice ha un valore ottimo. L'intercetta della retta con l'asse y non è nell'origine degli assi come ci dice la teoria. Tuttavia il valore di q (-2 Pa) è inferiore all'errore sulla misura delle pressioni.

La legge prevista dalla teoria è $P_t = dg\Delta x_c = k\Delta x_c$, con $k = dg$: questo k corrisponde al coefficiente angolare della retta interpolatrice. Nel nostro caso, alla temperatura di circa 28 °C, $d = 996\,\mathrm{kg/m^3}$ e $g = 9{,}805\,\mathrm{m/s^2}$, quindi $k = 9{,}77 \times 10^3$ Pa/m $= 0{,}0977$ hPa/mm, in buon accordo con i dati sperimentali.

16.3.7 Note

In questa esperienza abbiamo versato l'acqua a poco a poco, al contrario della precedente dove è stato il tubicino che è stato portato via via a maggiori profondità. Con questa nuova modalità, inevitabile usando un bicchiere, si può osservare la tendenza del pelo libero dell'acqua nel bicchiere a salire all'aumentare della pressione esterna.

L'aria nel tubicino non è incomprimibile: se aumentiamo la pressione a parità di sostanza il volume diminuirà. Assumiamo che la temperatura dell'aria non cambi e che segua la legge di Boyle:

$$P_0 V_0 = P_f V_f \tag{16.5}$$

Nella nostra esperienza la pressione dell'aria nel tubicino è variata da $P_0 = 1012{,}3$ Pa a 1026,7 Pa, con un aumento del 1,4%. Il volume deve perciò diminuire di altrettanto.

Se abbiamo solo l'aria in un tubicino di forma cilindrica e lungo inizialmente un metro avremo una variazione di lunghezza di circa 1,4 cm: un effetto piuttosto evidente.

Se l'aria comprende anche la camera d'aria nel bicchiere l'effetto sarà meno visibile. Se il volume della camera d'aria nel bicchiere è quello prevalente e se il raggio interno del bicchiere è venti volte più grande del tubicino, possiamo aspettarci un effetto dieci o venti volte più piccolo, in linea con quanto osservato.

17 Misura della densità dell'aria

17.1 Pressione atmosferica e altitudine

Possiamo misurare direttamente la pressione atmosferica con un sensore di pressione per Arduino, come la scheda BMP388. Dalla misura della pressione atmosferica possiamo ricavare indirettamente la densità dell'aria applicando la legge di Stevino. Supponiamo che la pressione atmosferica vari linearmente con l'altitudine (sicuramente vero per piccoli dislivelli). Se conosciamo la pressione in due località a noi prossime e di altitudine nota possiamo ricavare la densità dell'aria (che supponiamo costante) attraverso la legge di Stevino:

$$P = P_0 - dgh \tag{17.1}$$

dove P è la pressione all'altezza h, P_0 è la pressione della località di riferimento ad altezza 0 m, d la densità dell'aria alla nostra temperatura, g l'accelerazione di gravità, h il dislivello con la località di riferimento; il meno significa che la pressione scende con l'aumentare della altitudine. Se mettiamo in evidenza d possiamo scrivere:

$$d = \frac{P_0 - P}{hg} \tag{17.2}$$

La densità dell'aria varia sensibilmente con la temperatura: per questo motivo per conoscere i valori noti di densità da opportune tabelle di riferimento è necessario conoscere la temperatura.

17.2 Descrizione dell'esperienza

L'esperienza consiste nel misurare la pressione atmosferica (e per controllo anche la temperatura) a due altezze diverse. Dalla altezza e dalla differenza di pressione possiamo ricavare la densità dell'aria. Come illustrazione delle possibilità del metodo ripetiamo l'esperienza a tre possibili dislivelli, con gradi di accuratezza crescenti per il valore di densità trovato. Se la densità è:

$$d = \frac{\Delta P}{hg} \tag{17.3}$$

allora l'errore associato è:

$$\Delta d = d\left(\frac{\Delta(\Delta P)}{\Delta P} + \frac{\Delta h}{h} + \frac{\Delta g}{g}\right) \tag{17.4}$$

17.3 Apparecchi utilizzati

1. scheda Arduino Uno;
2. scheda BMP388;
3. metro a nastro (accuratezza 1 mm) o telemetro laser (accuratezza 10 cm).

17.4 Montaggio delle apparecchiature

Colleghiamo il sensore secondo lo schema del sensore BMP388 illustrato nel capitolo sui sensori di pressione [63]. Siamo immediatamente pronti per le misure.

Codice per Arduino

Il codice per Arduino è lo stesso codice utilizzato per il BMP388.

17.5 Procedimento

Vogliamo fare tre gruppi di misure con altezze diverse: una tra tavolo e pavimento, una tra sommità di un armadio e pavimento e una tra piani diversi di una abitazione. Per ogni gruppo di misure procediamo così.

1. Misuriamo il dislivello con un metro a nastro se vogliamo fare i primi due gruppi di misure o col telemetro laser per misurare la distanza tra i piani di una abitazione. Per quest'ultimo caso procediamo secondo le nostre possibilità: se abbiamo una rampa di scale libera possiamo usare direttamente il telemetro oppure con il metro a nastro misuriamo l'altezza di ogni piano.
2. Posizioniamo il sensore all'altezza base e aspettiamo che la pressione (e possibilmente la temperatura) si stabilizzi: ci vorrà almeno un minuto. Possiamo fare la lettura.
3. Ripetiamo la misura allo stesso modo per l'altezza superiore.

Osserviamo che il sensore non funziona correttamente né alla luce diretta del Sole, né se la temperatura non si è stabilizzata, dato che la misura della pressione dipende anche alla temperatura a cui si trova il sensore. Le misurazioni vanno effettuate preferibilmente al chiuso.

17.6 Dati sperimentali

Chiamiamo h il dislivello misurato: l'errore associato è 2 mm per le misure col metro a nastro e 10 cm per le misure con telemetro laser.

Chiamiamo P_i la misura di pressione al pavimento e P_s quella al livello superiore. L'errore associato alla misura è 1 Pa.

Le misure relative alla palazzina e all'armadio sono state effettuate in due ore diverse della stessa giornata. La temperatura era di circa 29 °C. Per il tavolo e l'armadio la temperatura era circa 27 °C. L'accelerazione di gravità nella città in cui mi trovo è tabulata come $g = 9{,}80269\ \mathrm{m/s^2}$.

Ho ripetuto le misure una sola volta per brevità.

tavolo

	h (mm)	P_i (hPa)	P_s (hPa)
1	742	1015,63	1015,55
2	742	1015,65	1015,57

armadio

	h (mm)	P_i (hPa)	P_s (hPa)
1	2375	1015,60	1015,36
2	2375	1015,76	1015,50

palazzina			
	h (m)	P_i (hPa)	P_s (hPa)
1	11,6	1013,17	1011,83
2	11,6	1013,04	1011,69

17.7 Elaborazione dati sperimentali

Nella seguente tabella riportiamo:

- la variazione di pressione $\Delta P = P_i - P_s$;
- la densità d;
- l'errore associato alla densità Δd;

L'errore associato alla variazione di pressione è $\Delta(\Delta P) = \Delta P_i + \Delta P_s = 2$ Pa.

Le misure di accelerazione di gravità le consideriamo praticamente senza errore.

tavolo			
	ΔP (Pa)	d (kg/m^3)	Δd (kg/m^3)
1	8	1,1	0,3
2	8	1,1	0,3

armadio			
	ΔP (Pa)	d (kg/m^3)	Δd (kg/m^3)
1	24	1,03	0,08
2	26	1,12	0,08

palazzina			
	ΔP (Pa)	d (kg/m^3)	Δd (kg/m^3)
1	134	1,18	0,03
2	135	1,19	0,03

17.8 Conclusioni

I valori di densità dell'aria che abbiamo trovato in letteratura sono 1,18 kg/m^3 a 25 °C e 1,16 kg/m^3 a 30 °C. Di conseguenza, nei limiti degli errori sperimentali, la concordanza tra misure fatte e dati teorici è molto buona.

17.9 Note

Le misure che mi hanno dato più problemi sono quelle relative all'armadio dal momento che la temperatura tra pavimento e soffitto variava anche di due gradi. Osserviamo che prima di effettuare almeno la prima misura è opportuno fare stabilizzare il sensore per almeno cinque minuti.

18 Legge di Boyle

18.1 Introduzione

La legge di Boyle riguarda i gas perfetti e afferma che, per una data quantità di gas mantenuto a temperatura costante in un recipiente chiuso, la pressione e il volume sono inversamente proporzionali:

$$PV = cost \tag{18.1}$$

Se usiamo la legge dei gas perfetti possiamo anche scrivere:

$$PV = nRT \tag{18.2}$$

dove n è il numero di moli del gas e R la costante dei gas perfetti. Pressione, volume e temperatura vanno indicate preferibilmente nelle unità di misura del sistema internazionale; altrimenti la costante R deve essere adeguata di conseguenza.

18.2 Descrizione dell'esperienza

In questa esperienza faremo variare il volume d'aria contenuto in una siringa ermeticamente chiusa, misurando la pressione conseguente col sensore Adafruit MPLRS: dovremo verificare che il prodotto del volume d'aria per la pressione rimangono costanti.

L'esperienza è divisa in due parti: la prima è quella della compressione e la seconda dell'espansione. La fase di compressione ha come limite il valore massimo di pressione misurabile dal sensore cioè circa 1,7 bar. La seconda è limitata dalla capacità di tenuta della siringa e del tubicino che la collega al sensore; qui invece non abbiamo problemi col sensore che ha una portata minima quasi nulla. Siccome una quantità d'aria non trascurabile è contenuta nel tubicino, abbiamo anche il problema di non espandere troppo il pistone per non introdurre un eccessivo errore sistematico sul volume di partenza. Con questa ultima osservazione è meglio utilizzare una siringa con un cilindro di grosse dimensione, ad esempio 50 ml.

Nell'esperienza bisognerebbe assicurarsi che la temperatura rimanga costante. Noi non abbiamo un termometro in contatto con l'aria della siringa, ma se evitiamo di stringere la siringa tra le mani allora la temperatura (secondo esperienze analoghe con apparecchi didattici) varia solo di qualche grado e la possiamo considerare, nei limiti degli errori sperimentali, constante.

18.3 Apparecchi utilizzati

- siringa da 20 ml con venti tacche.
- tubo di gomma di raccordo, lunghezza circa 10 cm
- scheda Arduino Uno (si può usare un R3 o R4)
- sensore Adafruit MPLRS

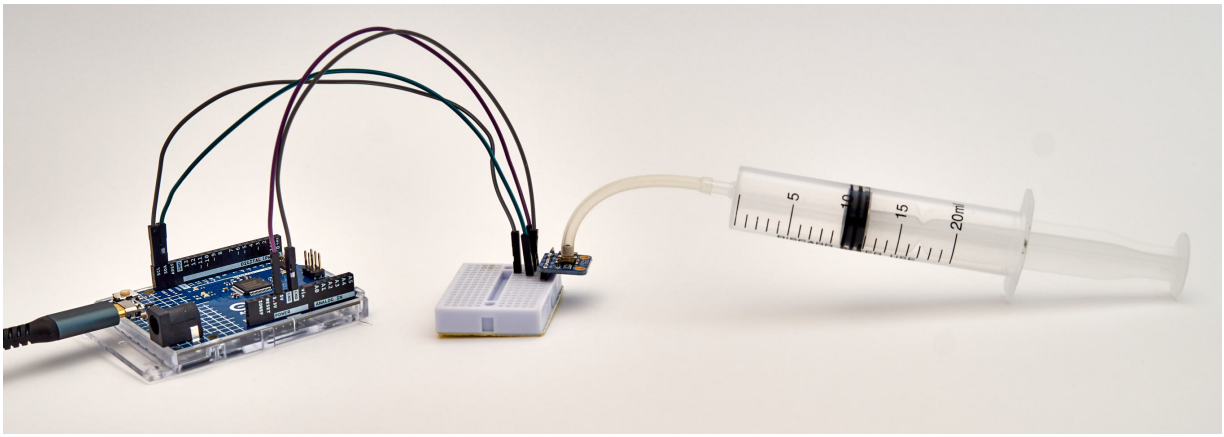
- breadboard
- quattro cavi Dupont maschio-maschio da circa 20 cm

Il sensore di pressione ha una precisione dell'ordine di 50 Pa. Non sappiamo quale sia l'accuratezza effettiva, ma ciò che ci interessa è soprattutto che la pressione indicata sia direttamente proporzionale a quella vera e che sia stabile nel tempo e questo è garantito.

La siringa utilizzata è da 20 ml perché è quella che avevo disponibile in un kit di esperienze di statica dei fluidi. Se possibile consiglio di utilizzarne una da 50 ml, come qui sopra ho già indicato. Ho provato preliminarmente ad usare una siringa da 5 ml con un tubicino da diversi centimetri di lunghezza e i risultati sono stati deludenti, soprattutto nella fase di espansione.

18.4 Montaggio apparecchi

Inseriamo il sensore su una breaboard, come visibile al centro dell'immagine seguente. Colleghiamo il sensore ad Arduino con i quattro cavi Dupont, con lo schema rappresentato al paragrafo 7.4. Il tubicino di gomma va inserito prima nel beccuccio della siringa (nel nostro caso esso era più piccolo del beccuccio, ma ciò ne ha facilitato la presa) e infine, dopo aver scelto da quale volume partire con l'esperienza, colleghiamo l'altra estremità al sensore.



18.5 Codice per l'utilizzo

Per l'uso del sensore potrebbe andare bene il codice già visto nel paragrafo 7.4; tuttavia qui ne proponiamo una variante che prende più letture successive e ne calcola il valor medio e la deviazione standard. Ai fini del successivo calcolo dell'errore questa maggior precisione è superflua, ma in questa maniera siamo più certi della stabilità delle letture che abbiamo effettuato.
Il valor medio della pressione p può essere ottenuto così:

$$\overline{p} = \frac{\sum_{i=1}^{n} p_i}{n} \tag{18.3}$$

Invece la deviazione standard può essere valutata con:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (\bar{p} - p_i)^2}{n}} \tag{18.4}$$

Per utilizzare queste relazioni si dovrebbero memorizzare i valori p_i di ogni singola lettura. Possiamo evitare tutto ciò accumulando la somma parziale delle pressioni p_i e del loro quadrato ${p_i}^2$ e usando la seguente relazione per valutare la deviazione standard:

$$\sigma = \frac{1}{n} \sqrt{n \sum_{i=1}^{n} (p_i)^2 - \left(\sum_{i=1}^{n} p_i \right)^2} \tag{18.5}$$

```
#include <Wire.h>
#include "Adafruit_MPRLS.h"

int N = 20;
float pressione = 0;
float pressione_sigma = 0;

#define RESET_PIN  -1
#define EOC_PIN    -1
Adafruit_MPRLS mpr = Adafruit_MPRLS(RESET_PIN, EOC_PIN);

void setup() {
  Serial.begin(115200);
  Serial.println("MPRLS Simple Test");
  if (! mpr.begin()) {
    Serial.println("Failed to communicate with MPRLS sensor, check wiring?");
    while (1) {
      delay(10);
    }
  }
  Serial.println("Found MPRLS sensor");
}

void loop() {
  float p = 0;
  float p_sum = 0;
```

```
27    float p_sum_q = 0;
      int i;
29    for(i=1;i<=N;i++){
        p = mpr.readPressure();
31      p_sum = p_sum + p;
        p_sum_q = p_sum_q + p*p;
33      delay(250);
      }
35    pressione = p_sum/N;
      pressione_sigma = sqrt(abs(N*p_sum_q-p_sum*p_sum))/N;
37
      Serial.print("Pressione = ");
39    Serial.print(pressione,1);
      Serial.print(" +- ");
41    Serial.print(pressione_sigma,1);
      Serial.println(" Pa");
43
}
```

- Le righe 1 e 2 richiamano le librerie necessarie per utilizzare il protocollo I2C e per interfacciarsi con il sensore.
- Le righe 4 - 6 definiscono come variabili globali le variabili che conterranno il valor medio e la deviazione standard della pressione, nonché il numero di misure su cui fare la media.
- Le righe da 8 a 22 sono analoghe al codice che abbiamo già descritto.
- Le righe 25 - 27 mettono a zero le variabili che conterranno la singola misura, la somma delle misure e la somma dei loro quadrati.
- Le righe da 29 a 34 contengono il loop principale per fare le media su N misure.
- Le righe 35 - 36 danno il risultato finale.
- Le righe 38 - 42 stampano i risultati.

18.6 Procedimento

Colleghiamo Arduino al computer con il sensore non collegato al tubicino e lasciamo stabilizzare il sistema per qualche minuto.

Per la fase di compressione colleghiamo il tubicino alla siringa, portiamo lo stantuffo nella posizione di massimo volume, per metterci nelle condizioni di minimo errore relativo riferito al volume. Colleghiamo il tubicino al sensore e prendiamo la misura iniziale: sarà leggermente diversa da quella atmosferica, ma di poco. Avendo cura di non scaldare troppo la siringa, maneggiandola per l'estremità, spingiamo lo stantuffo nella posizione delle tacche successive, aspettiamo qualche secondo che si stabilizzi la pressione e la deviazione standard associata sia minima, e prendiamo la misura. Precediamo così sin quasi al limite della portata del sensore. Infine riportiamo il pistone nella posizione di partenza per valutare se ci siano state eventuali perdite d'aria o se le condizioni iniziali siano significativamente variate.

Per la fase di espansione colleghiamo il tubicino alla siringa, portiamo lo stantuffo nella posizione centrale, per metterci nelle condizioni che permettano un ulteriore espansione, ma non introducano un eccessivo errore nella valutazione del volume iniziale. Poi procediamo come nella fase precedente fin quasi a raddoppiare il volume iniziale e infine riportiamo il pistone nella posizione di partenza.

18.7 Dati sperimentali

Il valori di volume e temperatura sono nelle unità di misure indicate dallo strumento.

Compressione

Volume (ml)	Pressione (hPa)
20	1011,0
19	1067,4
18	1122,0
17	1190,5
16	1256,8
15	1342,7
14	1432,9
13	1530,5
20	1016,5

Espansione

Volume (ml)	Pressione (hPa)
10	1019,9
11	928,0
12	856,8
13	789,7
14	736,7
15	690,1
16	648,6
17	613,5
18	581,7
10	1012,9

L'errore associato al volume corrisponde a una tacca della siringa ovvero 1 ml; quello associato alla pressione è circa 0,5 hPa ovvero il massimo valore ottenuto dalle deviazioni standard nell'esperimento: per semplicità li prendiamo tutti uguali.

18.8 Elaborazione dati sperimentali

L'errore associato al prodotto PV è dato da:

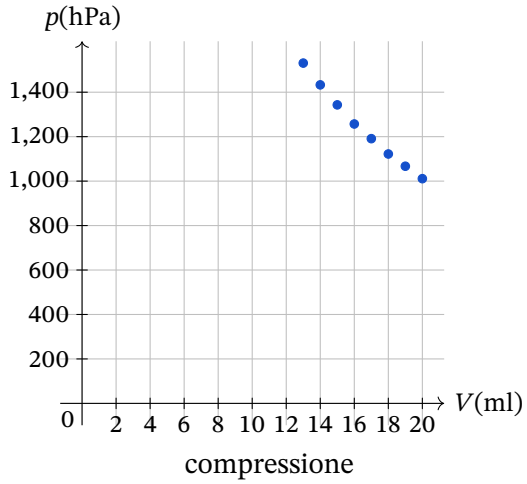
$$\Delta(PV) = PV\left(\frac{\Delta P}{P} + \frac{\Delta V}{V}\right) \tag{18.6}$$

Compressione

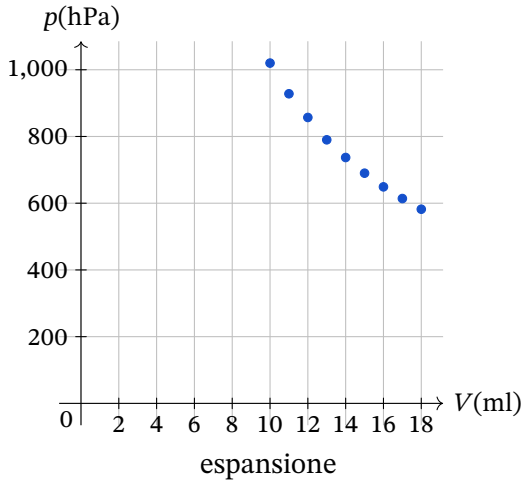
V (10^{-6} m^3)	P (hPa)	PV (Pa m^3)	$\Delta(PV)$ (Pa m^3)
20	1011,0	2,02	0,10
19	1067,4	2,03	0,11
18	1122,0	2,02	0,11
17	1190,5	2,02	0,12
16	1256,8	2,01	0,13
15	1342,7	2,01	0,14
14	1432,9	2,01	0,14
13	1530,5	1,99	0,15
20	1016,5	2,03	0,10

Espansione

V (10^{-6} m^3)	P (hPa)	PV (Pa m^3)	$\Delta(PV)$ (Pa m^3)
10	1019,9	1,02	0,10
11	928,0	1,02	0,09
12	856,8	1,03	0,09
13	789,7	1,03	0,08
14	736,7	1,03	0,07
15	690,1	1,04	0,07
16	648,6	1,04	0,07
17	613,5	1,04	0,06
18	581,7	1,05	0,06
10	1012,9	1,01	0,10



compressione



espansione

Nei limiti dell'errore sperimentale il prodotto della pressione per il volume si è mantenuto costante e la legge è stata verifica.

Il prodotto pressione-volume tende a diminuire nella fase di compressione ed aumentare in quella di espansione: non sappiamo dare un'interpretazione univoca di questo fenomeno.

L'ultima misura di controllo fornisce risultati accettabili. Bisogna osservare che è molto difficile riportare il pistone nell'esatta posizione di partenza.

18.9 Note

Facciamo attenzione a maneggiare l'apparato sperimentale il meno possibile per evitare di riscaldarlo con le mani.

19 Calorimetro delle mescolanze

19.1 Calorimetro

Il calorimetro è un apparecchio solitamente realizzato con un contenitore termicamente isolato all'interno del quale si pone un liquido di massa e calore specifico noti e un termometro per misurarne la temperatura. Nel liquido viene poi immerso un oggetto di massa e temperatura iniziale nota, ma di cui non si conosce il calore specifico. Dalla misura della temperatura del liquido dopo il raggiungimento dell'equilibrio termico possiamo determinare il calore specifico ignoto.

Per realizzare questa esperienza possiamo procere in due maniere diverse: nella prima avremo bisogno di poche misure, con una limitata cura dei dettagli sperimentali; nella seconda cercheremo di eliminare alcuni errori sistematici.

19.1.1 I metodo

La legge fondamentale della calorimetria quantifica la quantità di calore Q assorbito o ceduto da un corpo di calore specifico c e massa m per passare dalla temperatura iniziale T_i a quella finale T_f:

$$Q = mc\Delta T = mc(T_f - T_i) \tag{19.1}$$

Il liquido che utilizziamo nel calorimetro è l'acqua. Il corpo X che immergiamo in acqua è più caldo e trasferisce all'acqua del calore. Il calore perso dal corpo è idealmente uguale a quello guadagnato dall'acqua; la somma algebrica del calore fornito dal corpo caldo (negativo) e il calore assorbito dall'acqua (positivo) è uguale a zero (l'energia totale si conserva se non ci sono dispersioni). Indichiamo con T_e la temperatura di equilibrio del sistema.

$$\begin{aligned} Q_x + Q_a &= 0 \\ m_X c_X (T_e - T_{iX}) + m_a c_a (T_e - T_{ia}) &= 0 \\ m_X c_X (T_{iX} - T_e) &= m_a c_a (T_e - T_{ia}) \\ c_X &= \frac{m_a c_a (T_e - T_{ia})}{m_X (T_{iX} - T_e)} \end{aligned} \tag{19.2}$$

L'errore associato è:

$$\Delta c_X = c_X \left(\frac{\Delta m_a}{m_a} + \frac{\Delta c_a}{c_a} + \frac{\Delta(T_e - T_{ia})}{(T_e - T_{ia})} + \frac{\Delta m_X}{m_X} + \frac{\Delta(T_{iX} - T_e)}{(T_{iX} - T_e)} \right) \tag{19.3}$$

19.1.2 Il metodo

Quando scaldiamo il liquido presente nel calorimetro finiamo inevitabilmente per scaldare anche il calorimetro stesso: la tenuta termica dello strumento non è totale. Possiamo tener conto di questo aspetto calcolando m_c, l'*equivalente in acqua della massa del calorimetro*.

Abbiamo una massa di acqua fredda m_1 alla temperatura T_{i1} e versiamo in essa una massa di acqua calda m_2 alla temperatura T_{i2}. Allora la somma algebrica del calore Q_1 assorbito dall'acqua fredda e il calore Q_c assorbito dal calorimetro più il calore Q_2 ceduto dall'acqua calda deve essere nulla.

$$\begin{aligned} Q_c + Q_1 + Q_2 &= 0 \\ m_c c(T_e - T_{i1}) + m_1 c(T_e - T_{i1}) + m_2 c(T_e - T_{i2}) &= 0 \\ m_c c(T_{i1} - T_e) &= m_1 c(T_e - T_{i1}) + m_2 c(T_e - T_{i2}) \\ m_c &= \frac{m_1(T_e - T_{i1}) + m_2(T_e - T_{i2})}{(T_{i1} - T_e)} \\ m_c &= -m_1 + \frac{m_2(T_e - T_{i2})}{(T_{i1} - T_e)} \end{aligned} \tag{19.4}$$

L'errore associato è:

$$\Delta m_c = \Delta m_1 + \frac{m_2(T_e - T_{i2})}{(T_{i1} - T_e)} \left(\frac{\Delta m_2}{m_2} + \frac{\Delta(T_e - T_{i2})}{(T_e - T_{i2})} + \frac{\Delta(T_{i1} - T_e)}{(T_{i1} - T_e))} \right) \tag{19.5}$$

Tenendo conto dell'equivalente in acqua della massa del calorimetro possiamo riesprimere la precedente relazione che ci ha portato a trovare il calore specifico incognito.

$$\begin{aligned} Q_x + Q_a + Q_c &= 0 \\ m_X c_X(T_e - T_{iX}) + m_a c_a(T_e - T_{ia}) + m_c c(T_e - T_{i1}) &= 0 \\ m_X c_X(T_{iX} - T_e) &= (m_a + m_c)\, c_a(T_e - T_{ia}) \\ c_X &= \frac{(m_a + m_c)\, c_a(T_e - T_{ia})}{m_X(T_{iX} - T_e)} \end{aligned} \tag{19.6}$$

L'errore associato è:

$$\Delta c_X = c_X \left(\frac{\Delta(m_a + m_c)}{(m_a + m_c)} + \frac{\Delta c_a}{c_a} + \frac{\Delta(T_e - T_{ia})}{(T_e - T_{ia})} + \frac{\Delta m_X}{m_X} + \frac{\Delta(T_{iX} - T_e)}{(T_{iX} - T_e)} \right) \tag{19.7}$$

19.2 Descrizione dell'esperienza

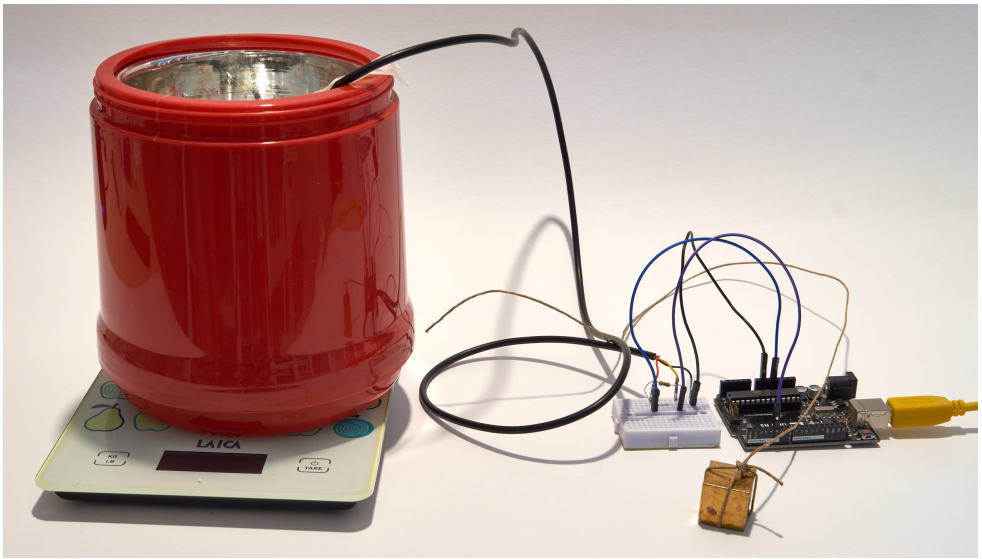
L'esperienza, nella forma casalinga, si svolge usando un thermos o un porta vivande termico come calorimetro, ovvero come contenitore termicamente isolato. Come oggetto caldo da inserire nel calorimetro ho usato un blocchetto di rame di un kit economico di cubi di diversi materiali. Per scaldare il corpo usiamo l'acqua bollente. Come sensore di temperatura da inserire nel calorimetro usiamo il sensore DS18B20.

19.3 Apparecchi utilizzati

1. scheda Arduino Uno (R3 o R4)
2. sensore DS18B20
3. breadboard
4. resistenza da 4,7 kΩ
5. Cavi di collegamento
6. bilancia elettronica: portata 5 kg, sensibilità 1 g
7. blocco metallico di rame
8. tegame per fare bollire l'acqua
9. portavivande in plastica con all'interno un vaso di Dewar: volume interno quasi un litro.

19.4 Montaggio apparecchiature

Il sensore con Arduino è lo stesso utilizzato e descritto nel capitolo 9.2.3. In questo caso non cambiamo neanche il codice prima descritto. Il porta vivande qui utilizzato è posto e lasciato sopra la bilancia. Il sensore di temperatura è tenuto fermo sul bordo dell'apertura del portavivande con una striscia di nastro adesivo in modo che rimanga penzoloni sull'acqua contenuta del recipiente senza toccarne il fondo. Il pesetto è stato avvolto con uno spago per poterlo inserire nell'acqua bollente e nel portavivande.



19.5 Procedimento

Determinazione del calore specifico del rame, senza correzione

- Misuriamo la massa del blocco di rame.
- Poggiamo il calorimetro vuoto sulla bilancia e misuriamone il peso.
- Mettiamo nel calorimetro abbastanza acqua da riempirlo per circa due terzi del volume: se la bilancia consente la tara azzeriamola prima di versare l'acqua. Altrimenti determiniamo la massa dell'acqua per sottrazione.
- Immergiamo il sensore nell'acqua senza che tocchi il fondo del recipiente e teniamolo fermo con del nastro adesivo.
- Lasciamo che il sensore si stabilizzi e misuriamo la temperatura dell'acqua fredda.
- Prepariamo il blocco di rame di cui vogliamo misurare il calore specifico legandolo con un spago in modo da poterlo maneggiare senza toccarlo.
- Mettiamo a bollire l'acqua nel tegame o in un becher resistente al calore e immergiamoci il blocco di rame. Estraiamolo solo quando l'ebollizione è tumultuosa, segno evidente che siamo realmente prossimi ai 100 °C.
- Estraiamo il blocco dall'acqua bollente e immergiamolo velocemente nell'acqua del calorimetro, muovendolo un po' senza appoggiarlo subito sul fondo. *Fare attenzione al vaso Dewar perché è molto delicato*: non lasciare cadere il blocco metallico al suo interno, pena la sua fragorosa rottura.
- Monitoriamo la temperatura dell'acqua del calorimetro determinando la temperatura massima a cui arriva (bastano realmente pochi secondi).

Determinazione dell'equivalente in acqua del calorimetro

- In questo caso riempiamo il calorimetro solo per metà.
- Procediamo allo stesso modo del caso precedente fino al riscaldamento dell'acqua calda. Ora, per i motivi che di seguito sono indicati, immergiamo il sensore nell'acqua calda, dopo aver misurato la temperatura stabilizzata dell'acqua fredda.

  Uno dei problemi che si presenta anche nel caso precedente è il raffreddamento del corpo caldo fino all'inserimento nel calorimetro. Con l'acqua a 100 °C questo fenomeno è molto consistente. Ragion per cui abbiamo appurato che è meglio riscaldare l'acqua a non più di ottanta gradi. Superata questa temperatura spegniamo il fornello.
- Misuriamo la temperatura dell'acqua calda per circa un minuto, osservando di quanto scende la temperatura in circa venti secondi.
- Leggiamo la temperatura dell'acqua calda al momento di togliere da esso il sensore, per riimmergerlo nell'acqua fredda.
- Entro venti secondi immergiamo abbastanza acqua calda nel calorimetro in modo da portarlo a due terzi della sua capacità, leggendo sulla bilancia quanta acqua stiamo versando.
- Come temperatura dell'acqua calda versata usiamo l'ultima lettura fatta diminuendola per quel tanto che si sarà raffreddata nei venti secondi.
- Infine monitoriamo la temperatura dell'acqua del calorimetro determinando la temperatura massima a cui arriva.

19.6 Dati sperimentali

Il valori di volume e temperatura sono nelle unità di misure indicate dallo strumento.

rame

m_1 (kg)	T_{i1} (°C)	m_2 (kg)	T_{i2} (°C)	T_e (°C)
0,666	27,07	0,066	100,0	27,65

$$\begin{aligned} \Delta m_1 &= 0{,}001\,\text{kg} \\ \Delta m_2 &= 0{,}001\,\text{kg} \\ \Delta T &= 0{,}5\,^\circ\text{C} \\ \Delta(T_e - T_{ia}) &= 0{,}06\,^\circ\text{C} \\ \Delta(T_{iX} - T_e) &= 0{,}5\,^\circ\text{C} \end{aligned} \tag{19.8}$$

Gli errori associati ai salti di temperatura tra corpo caldo e acqua sono stati considerati con l'accuratezza indicata dalle specifiche dello strumento. Gli errori associati ai salti di temperatura tra acqua fredda e riscadalta sono stati considerati con la sensibilità dello strumento perché relativi a piccole variazioni di temperatura.

equivalente in acqua

m_1 (kg)	T_{i1} (°C)	m_2 (kg)	T_{i2} (°C)	T_e (°C)
0,459	22,94	0,183	77,1	37,43

$$\begin{aligned} \Delta m_1 &= 0{,}001\,\text{kg} \\ \Delta m_2 &= 0{,}001\,\text{kg} \\ \Delta T &= 0{,}5\,^\circ\text{C} \\ \Delta(T_e - T_{i2}) &= 0{,}5\,^\circ\text{C} \\ \Delta(T_{i1} - T_e) &= 0{,}06\,^\circ\text{C} \end{aligned} \tag{19.9}$$

19.7 Elaborazione dati sperimentali

rame

c (j/kg °C)	Δc (j/kg °C)	c^* (j/kg °C)	Δc^* (j/kg °C)
340	50	370	60

equivalente in acqua

m_c (kg)	Δm_c (kg)
0,042	0,013

In letteratura troviamo che il rame ha un calore specifico $c = 385\,\text{j/kg}\,^\circ\text{C}$, per cui *i valori trovati*, nei limiti degli errori sperimentali, *sono del tutto corrispondenti*. Tuttavia grande è l'errore associato ad entrambe le misure.

19.8 Note finali

- I valori ottenuti sono caratterizzati da un errore notevole dovuto alla piccola dimensione del blocco di metallo utilizzato e il conseguente piccolo innalzamento di temperatura. Un oggetto dalla massa di almeno un terzo di quella dell'acqua nel quale verrà immerso garantirebbe risultati più accurati.
- La temperatura dell'oggetto caldo non è certamente quella indicata. Infatti il tragitto dal contenitore dell'acqua bollente al calorimetro abbassa di un valore indefinito, ma dell'ordine di almeno uno o due gradi, la sua temperatura. Se avessimo considerato una temperatura iniziale di 98 °C allora il calore specifico corretto con l'equivalente in acqua del calorimetro sarebbe stato un ottimo $c^* = 380\,\mathrm{j/kg\,°C}$.
- Per determinare l'equivalente in acqua del calorimetro ho inizialmente utilizzato acqua bollente. I valori trovati erano piuttosto alti. Il problema era sia il raffreddamento dell'acqua fino al suo versamento nel calorimetro, ma anche le perdite per irraggiamento ed evaporazione. Questa la ragione per cui poi ho elaborato la proposta di determinazione che qui ho presentato.
- Il portavivande come calorimetro ha il difetto di essere totalmente aperto. D'altra parte usare un tradizionale thermos pone dei problemi all'inserimento in esso dell'oggetto da analizzare.

20 Studio della velocità del suono in aria

20.1 Velocità di un onda

Un'onda periodica che si propaga in un mezzo si sposta con una velocità v che è legata alla lunghezza d'onda λ e al periodo T:

$$v = \frac{\lambda}{T} \tag{20.1}$$

La velocità del suono in aria in condizioni normali è circa 343 m/s. Questa velocità dipende da diverse grandezze la più importante delle quali è la temperatura; intervengono in maniera significativa anche la densità e l'umidità.

La velocità del suono in aria può essere determinata con la seguente relazione:

$$v = 20{,}05\sqrt{\theta + 273{,}15}\ \text{m/s} \tag{20.2}$$

dove θ è il valore numerico della temperatura in celsius.

Questa relazione può essere ricavata considerando l'aria come una miscela di gas perfetti biatomici. La relazione vale per l'aria secca. La presenza di umidità determina un aumento della velocità che alcuni autori quantificano fino ad un 0,35%.

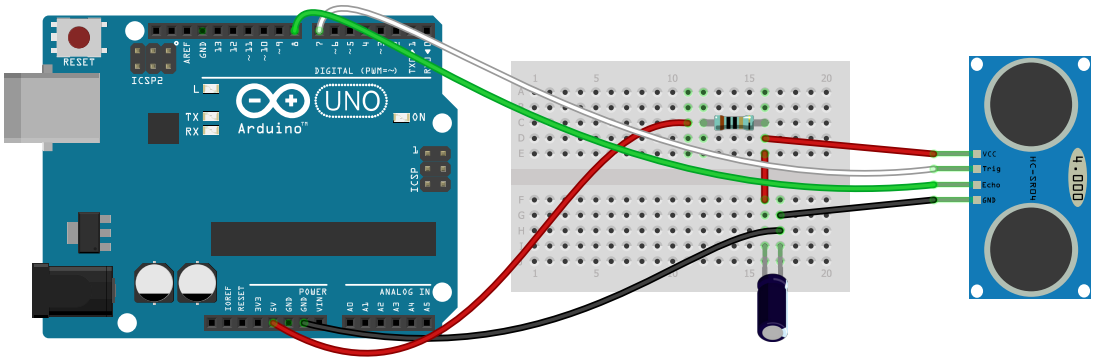
20.2 Descrizione dell'esperienza

I sensori di distanza ad ultrasuoni sfruttano la conoscenza di questa velocità di propagazione per misurare distanze attraverso la misura diretta del tempo di propagazione di un segnale sonoro per andare dal sensore all'oggetto voluto e ritornare indietro. La scheda HC-SR04 misura direttamente questo tempo di propagazione del suono. Da questa grandezza, se misuriamo la distanza tra la scheda e un oggetto opportuno, possiamo determinare la velocità del suono nell'aria.

20.3 Apparecchi utilizzati

1. Arduino uno;
2. scheda HC-SR04;
3. metro a nastro: portata 5 m; accuratezza 1 mm.
4. breadboard;
5. resistenza 220 Ω;
6. capacità 100 μF;
7. cavetteria varia.

20.4 Montaggio delle apparecchiature



Secondo specifiche la scheda HC-SR04 opera ad una frequenza di 40 kHz inviando 8 impulsi sonori strettamente ravvicinati. Il suono riflesso è ascoltato se proveniente da una zona larga circa 15°. Il tempo misurato è dato in microsecondi. Le specifiche parlano della capacità di misurare distanze comprese tra 2 cm e 400 cm, con una incertezza di 0,5 cm. In pratica non sappiamo con quale precisione si possa misurare il tempo di propagazione del segnale. Nello svolgimento dell'esperienza cercheremo di valutare questa precisione.

La scheda può essere connessa direttamente ad Arduino. In pratica si può osservare la tendenza a sensibili fluttuazioni nei valori ottenibili: ciò ne rende difficile l'uso e la corretta interpretazione dei dati. Consultando apparecchi ad ultrasuoni di altre marche ho trovato il suggerimento di usare un condensatore tra la messa a terra e la linea di alimentazione, per rendere più uniforme la tensione in ingresso. Questo suggerimento, applicato alla nostra scheda, mi ha consentito di avere valori molto più stabili. Nel manuale in cui ho individuato questo suggerimento si dava indicazione di usare una resistenza di 100 Ω o di 10 Ω, a seconda dell'apparecchio. Nel nostro caso disponevo solo di una resistenza di 220 Ω e quella ho utilizzato. Ho poi sperimentato che la scheda funziona correttamente anche senza la resistenza, ma senza ulteriori miglioramenti o peggioramenti.

Per quanto riguarda il montaggio dei componenti ricordiamoci che i comuni condensatori, come quello da noi usato e rappresentato, sono condensatori elettrolitici. Questi condensatori sono polarizzati: accettano solo una determinata polarità ai loro capi. Nel condensatore è indicato con un segno meno il piedino che va posto a potenziale minore, nel nostro caso la messa a terra (0 V).

Codice per Arduino

Il codice qui di seguito illustrato legge N volte di seguito il tempo t_i che l'onda sonora impiega per andare e tornare. In fine si calcola il valor medio delle misure ottenute e la deviazione standard. Il valor medio può essere ottenuto così:

$$\bar{t} = \frac{\sum_{i=1}^{n} t_i}{n} \tag{20.3}$$

Invece la deviazione standard può essere valutata con:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (\bar{t} - t_i)^2}{n}} \tag{20.4}$$

Per utilizzare queste relazioni si dovrebbero memorizzare i valori t_i. Possiamo evitare tutto ciò accumulando la somma parziale dei tempi t_i e dei tempi al quadrato ${t_i}^2$ e usando la seguente relazione

per valutare la deviazione standard:

$$\sigma = \frac{1}{n}\sqrt{n\sum_{i=1}^{n}(t_i)^2 - \left(\sum_{i=1}^{n} t_i\right)^2} \tag{20.5}$$

```
1  int trigPin = 7;
2  int echoPin = 8;
3  int i = 0;
4  long t_sum = 0;
5  long t_sum_q = 0;
6  long t = 0;
7  float sigma = 0.0;
8  int N = 50;
9
10 void setup()
11 {
12   Serial.begin(9600);
13   pinMode(trigPin, OUTPUT);
14   pinMode(echoPin, INPUT);
15   digitalWrite(trigPin, LOW);
16 }
17
18 void loop()
19 {
20   t = 0;
21   t_sum = 0;
22   t_sum_q = 0;
23   for(i=1;i<=N;i++){
24     digitalWrite(trigPin, LOW);
25     delayMicroseconds(10);
26     digitalWrite(trigPin, HIGH);
27     delayMicroseconds(10);
28     digitalWrite(trigPin, LOW);
29     long tUs = pulseIn(echoPin, HIGH);
30     t = tUs ; // s
31     t_sum = t_sum + t;
32     t_sum_q = t_sum_q + t*t;
33     Serial.print(i);
34     Serial.print("␣␣␣");
35     Serial.println(tUs);
36     delay(100);
37   }
38   t = t_sum/N;
39   sigma = sqrt(abs(N*t_sum_q-t_sum*t_sum))/N;
40   Serial.print("-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣␣␣␣␣␣␣␣");
41   Serial.print(t/2);
42   Serial.print("␣");
43   Serial.println(sigma/2,1);
44  }
```

- La riga 1 definisce il nome della variabile che contiene il pin a cui inviare il segnale per fare partire l'emissione del suono.
- La riga 2 definisce il nome della variabile che contiene il pin a cui ascoltare il suono riflesso.

- Le righe 3-8 inizializzano le varie variabili utilizzate nel codice. La variabile tUs è utilizzata per raccogliere il tempo, in microsecondi, indicato dal sensore. Volendo si può usare anche una variabile float.
- Le righe 10-16 inizializzano il sensore secondo la logica descritta in dettaglio nel capitolo sulla misurazione di distanze, a cui rimandiamo.
- La riga 18 da inizio al loop principale del programma.
- Le righe 19-21 pongono uguale a zero le variabili t (tempo), t_sum (somma parziale dei tempi) e t_sum_q (somma parziale dei tempi al quadrato).
- Righe 23-37: Abbiamo un loop che viene ripetuto per N volte.
- Righe 24-29: viene letto il tempo tUs dal sensore.
- Righe 30-32: vengono aggiornate le variabili t, t_sum e t_sum_q.
- Righe 33-35: viene stampato il numero i della misurazione e il valore del tempo.
- Riga 36: si attende un decimo di secondo prima di fare un'altra misura.
- Riga 38: viene attribuito a t il valor medio delle misure ottenute.
- Riga 39: viene calcolata la deviazione standard.
- Riga 41: viene stampato la metà del valor medio, ovvero il tempo per percorrere la distanza dal sensore dall'ostacolo.
- Riga 43: anche la deviazione viene divisa a metà e viene mostrata solo una cifra decimale.

Il sensore va tenuto fermo e ben allineato all'ostacolo riflettente con una terza mano. Sensore e Arduino vanno posti ad almeno mezzo metro da terra in modo da avere sufficiente campo libero davanti e sotto il sensore. Anche qualche filo penzolante sotto il sensore può inficiare le misure.

20.5 Procedimento

Poniamo il sensore a distanze crescenti da una parete rigida, piana e sufficientemente estesa rispetto alla distanza del sensore. In pratica ci conviene porci di fronte ad una parete libera. Osserviamo sul plotter seriale di Arduino se la curva dei valori che otteniamo è sufficientemente piana. Nella mia esperienza si riesce ad ottenere una retta con pochi scostamenti per ogni ciclo di loop. Lasciamo che vengano mostrate diverse misure, finché non siamo sicuri che i valori medi ottenuti variano solo di poche unità l'uno dall'altro. Infine misuriamo la distanza tra la parete e i cilindri metallici del sensore.

20.6 Dati sperimentali

	Distanza (m)	$\bar{t}$ (μs)	σ (μs)
1	0,255	732	2
2	0,375	1074	1
3	0,508	1457	2
4	0,630	1798	2
5	0,737	2104	2

L'errore che ho considerato per le distanze è 2 mm.

20.7 Elaborazione dati sperimentali

La temperatura della stanza in cui si è svolta l'esperienza era $t = 22\,°\mathrm{C}$ e l'umidità relativa 58%. La velocità del suono per l'aria secca in queste condizioni, secondo la relazione scritta all'inizio di questo capitolo, è $v = 344{,}5\,\mathrm{m/s}$. Se vogliamo tener conto anche dell'umidità arriviamo a circa $v = 346\,\mathrm{m/s}$.

Questi sono invece i nostri dati sperimentali, usando la seguente relazione per calcolare l'errore associato alla velocità.

$$\Delta v = v\left(\frac{\Delta s}{s} + \frac{\Delta t}{t}\right) \tag{20.6}$$

	Velocità (m/s)	Δv (m/s)
1	348	4
2	349	2
3	349	2
4	350	2
5	350	1

20.8 Conclusioni

Le misure ottenute sono allineate ai dati teorici e ai dati sperimentali comunemente noti. Anche considerando gli errori sperimentali ottenuti è presente una sistematica sovrastima della velocità di cui non sono riuscito a dare conto.

20.9 Note

Esistono in commercio varie schede tutte con la stessa sigla, ma di fattura e generazione differente. Negli innumerevoli tentativi di ottenere misure più uniformi e concordi con i valori di velocità tabulati in letteratura ho potuto provare anche a cambiare sensore. Ho a mia disposizione alcuni sensori (come quello che ho usato per ottenere i precedenti risultati) di prima generazione. Ho inoltre un altro sensore di seconda generazione che ha le resistenze disposte diversamente. Questo mi ha fornito risultati un po' più bassi, ma anche molto più fluttuanti al variare della distanza. Ho deciso quindi di non usarlo per questa prova.

20.10 Integrazione: esperienza con il US-015

21 Studio della velocità della luce

21.1 Velocità di un'onda e della luce

Un'onda periodica che si propaga in un mezzo si sposta con una velocità v che è legata alla lunghezza d'onda λ e al periodo T:

$$v = \frac{\lambda}{T} \tag{21.1}$$

La velocità della luce nel vuoto, e con ottima approssimazione anche in aria, è costante e vale esattamente 299792457 m/s.

21.2 Descrizione dell'esperienza

I sensori di distanza che fanno uso della luce sfruttano la conoscenza di questa velocità di propagazione per misurare distanze attraverso la misura diretta del tempo di propagazione che un segnale luminoso impiega per andare dal sensore all'oggetto voluto e ritornare indietro. La scheda VL53L1X [5.2] misura direttamente questo tempo di propagazione. Da questa grandezza, se misuriamo la distanza tra la scheda e un oggetto opportuno, possiamo determinare la velocità della luce nell'aria.

Per l'utilizzo in fisica è importante osservare che, al contrario del sensore sonar prima illustrato, qui (per quanto mi è dato di sapere) non è possibile conoscere direttamente il tempo di volo del raggio di luce, ma solo la distanza ad esso associata. Tuttavia questa distanza è ricavata dalla definizione di velocità della luce nel vuoto (o nell'aria) $\mathrm{c} = \frac{d}{t}$ e quindi conoscere la distanza equivale a conoscere il tempo di volo $t = \frac{d}{\mathrm{c}}$.

Quindi nella nostra esperienza ricaviamo il tempo di volo t' sperimentale dalla distanza indicata dal sensore. Ripetiamo questa misurazione del tempo di volo venti volte di seguito facendo la media dei risultati ottenuti e associando alla deviazione standard delle misure l'errore associato commesso.

Misuriamo la distanza reale d' con un metro a nastro e ricaviamo la velocità della luce c' corrispondente.

$$\mathrm{c}' = \frac{d'}{t'} \tag{21.2}$$

Ripetiamo questo processo per cinque o sei distanze differenti e con il metodo dei minimi quadrati ricaviamo la retta interpolatrice dei dati ottenuti. Se la variabile indipendente è il tempo e quella dipendente la distanza allora la pendenza della retta che otteniamo rappresenta la velocità della luce media ottenuta.

$$d' = \bar{c} \cdot t' + d_0 \tag{21.3}$$

21.3 Apparecchi utilizzati

1. Arduino uno;
2. scheda VL53L1X;
3. metro a nastro: portata 5 m; accuratezza 1 mm.

21.4 Montaggio delle apparecchiature

Per quanto riguarda il montaggio seguiamo le indicazioni già date nella sezione [5.2]. Il sensore va tenuto fermo e ben allineato all'ostacolo riflettente con una terza mano. Sensore e Arduino vanno posti ad almeno mezzo metro da terra in modo da avere sufficiente campo libero davanti e sotto il sensore. Anche qualche filo penzolante sotto il sensore può inficiare le misure.

Codice per Arduino

Il codice qui di seguito illustrato legge N volte di seguito la distanza misurata dal sensore d_i. In fine si calcola il valor medio delle misure ottenute e la deviazione standard. Il codice riportato è uguale a quello della sezione [5.2.4] con integrate delle righe aggiuntive per la ripetizione della misura della distanza e per il calcolo qui di seguito indicato.

Il valor medio può essere ottenuto così:

$$\overline{d} = \frac{\sum_{i=1}^{n} d_i}{n} \tag{21.4}$$

Invece la deviazione standard può essere valutata con:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (\overline{d} - d_i)^2}{n}} \tag{21.5}$$

Per utilizzare queste relazioni si dovrebbero memorizzare i valori d_i. Possiamo evitare tutto ciò accumulando la somma parziale dei tempi d_i e dei tempi al quadrato d_i^2 e usando la seguente relazione per valutare la deviazione standard:

$$\sigma = \frac{1}{n} \sqrt{n \sum_{i=1}^{n} (d_i)^2 - \left(\sum_{i=1}^{n} d_i \right)^2} \tag{21.6}$$

```
#include <Wire.h>
#include <VL53L1X.h>

VL53L1X sensor;
int i = 0;
long d_sum = 0;
long d_sum_q = 0;
long d = 0;
float sigma = 0.0;
int N = 20;

void setup()
{
```

```
14    Serial.begin(115200);
      Wire.begin();
16    Wire.setClock(400000); // use 400 kHz I2C
      sensor.setTimeout(500);
18    if (!sensor.init())
      {
20      Serial.println("Failed␣to␣detect␣and␣initialize␣sensor!");
        while (1);
22    }
      sensor.setDistanceMode(VL53L1X::Long);
24    sensor.setMeasurementTimingBudget(250000);
      }
26
  void loop()
28 {
      d = 0;
30    d_sum = 0;
      d_sum_q = 0;
32    for(i=1;i<=N;i++){
        d = sensor.readSingle(true);
34      d_sum = d_sum + d;
        d_sum_q = d_sum_q + d*d;
36    }
      d = d_sum/N;
38    sigma = sqrt(abs(N*d_sum_q-d_sum*d_sum))/N;
      Serial.print("-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣␣␣␣␣␣␣␣␣");
40    Serial.print(d);
      Serial.print("␣␣");
42    Serial.println(sigma,1);
      Serial.print(d_sum);
44    Serial.println();
  }
```

- Le righe 3-10 inizializzano le varie variabili utilizzate nel codice. La variabile ”d” corrisponde al d_i prima indicato ed è utilizzata per raccogliere la distanza, in millimetri, indicata dal sensore.
- Le righe 10-25 inizializzano il sensore secondo la logica descritta in dettaglio nel capitolo sulla misurazione di distanze, a cui rimandiamo. Il timingBudget indicato è del tutto adeguato per ottenere le misure più accurate possibili.
- La riga 27 dà inizio al loop principale del programma.
- Le righe 29-31 pongono uguale a zero le variabili d (distanza), d_sum (somma parziale delle distanze) e d_sum_q (somma parziale delle distanze al quadrato).
- Righe 32-36: Abbiamo un loop che viene ripetuto per N volte.
- Riga 33: viene letta la distanza dal sensore.
- Righe 34-35: vengono aggiornate le variabili d, d_sum e d_sum_q.
- Riga 37: viene attribuito a d il valor medio delle misure ottenute.
- Riga 38: viene calcolata la deviazione standard.
- Riga 40: viene stampato il valor medio.
- Riga 42: la deviazione viene mostrata con solo una cifra decimale.

21.5 Procedimento

Poniamo il sensore a distanze crescenti da una parete rigida, piana, di colore chiaro e sufficientemente estesa rispetto alla distanza del sensore. In pratica conviene mettersi di fronte ad una parete libera. Lasciamo che vengano mostrate diverse misure, finché non siamo sicuri che i valori medi ottenuti variano solo di poche unità l'uno dall'altro. Infine misuriamo la distanza tra la parete e la superficie del sensore, che andrà posta parallela alla superficie riflettente. Per quanto il sensore sia sensibile all'infrarosso vicino è meglio procedere in un ambiente poco illuminato.

21.6 Dati sperimentali

	d'(m)	$\overline{d}$ (m)	σ (m)
1	0,833	0,830	0,0007
2	0,970	0,962	0,0007
3	1,074	1,068	0,0005
4	1,208	1,199	0,0007
5	1,412	1,404	0,0008
6	1,558	1,550	0,0011

L'errore che ho considerato per le distanze è 1 mm.

21.7 Elaborazione dati sperimentali

Nella seguente tabella riportiamo:

1. Lo scostamento Δd tra le distanze $\overline{d}$ indicate dal sensore e quelle d' misurate col metro.
2. Il tempo di volo associato alla distanza $\overline{d}$.
3. La velocità c$'$.

	Δd	t' (s)	c$'$ (m/s)
1	0,003	$2{,}768 \times 10^{-9}$	$3{,}009 \times 10^{8}$
2	0,008	$3{,}209 \times 10^{-9}$	$3{,}023 \times 10^{8}$
3	0,006	$3{,}562 \times 10^{-9}$	$3{,}015 \times 10^{8}$
4	0,009	$3{,}999 \times 10^{-9}$	$3{,}020 \times 10^{8}$
5	0,008	$4{,}683 \times 10^{-9}$	$3{,}015 \times 10^{8}$
6	0,008	$5{,}170 \times 10^{-9}$	$3{,}013 \times 10^{8}$

Dai precedenti dati si osserva che la velocità della luce ricavata è sistematicamente sovrastimata, se pur di meno dell'un per cento del valore reale. Si osserva inoltre uno scarto sistematico in eccesso per la distanza rilevata; a parte la misura più piccola le altre hanno uno scarto che tende ad essere costante. Per questo motivo sottopongo a regressione lineare gli ultimi cinque dati: come variabile indipendente prendo il tempo e come variabile dipendente la distanza reale. In questo modo la retta interpolante dovrebbe avere come coefficiente angolare la velocità della luce indirettamente misurata.

Secondo quanto indicato nel capitolo [A] otteniamo la retta di equazione:

$$d = 3{,}00(2) \times 10^8 \text{ m/s} \cdot t + 0{,}00(6) \text{ m} \tag{21.7}$$

Il valore $x_0 = 0{,}00(6)$ m indica un errore sistematico di sovrastima delle distanze e il coefficiente di correlazione è $r = 0{,}99998$.

Possiamo infine scrivere che:

$$\mathrm{c}' = (3{,}002 \pm 0{,}008) \times 10^8 \text{ m/s} \tag{21.8}$$

Contro il valore corretto (approssimato alla quarta cifra significativa):

$$\mathrm{c} = (2{,}998 \pm 0{,}001) \times 10^8 \text{ m/s} \tag{21.9}$$

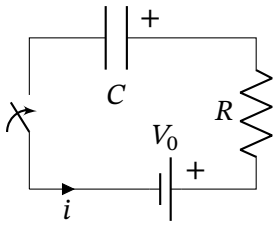
21.8 Conclusioni

Le misure ottenute sono del tutto allineate ai dati teorici e sperimentali comunemente noti, con uno scostamento dal valore atteso di meno dell'un per cento.

22 Il circuito RC

22.1 Introduzione

Il circuito RC è un circuito elementare costituito da tre elementi posti in serie: un generatore di f.e.m (costante), una resistenza e un condensatore.



Se il condensatore è scarico, quando si chiude il circuito in questo circolerà una corrente il cui andamento tende esponenzialmente a zero. Se invece il condensatore è carico e nel circuito non si inserisce il generatore, allora circolerà una corrente il cui andamento tende comunque a zero, ma di verso opposto a quello precedente.

Se applichiamo la seconda legge di Kirchhoff all'unica maglia del circuito possiamo scrivere:

$$RI(t) + \frac{Q(t)}{C} - V_0 = 0 \tag{22.1}$$

dove V_0 è la forza elettromotrice del generatore, RI è la caduta di tensione ai capi della resistenza e $\frac{Q(t)}{C}$ è la (contro)forza elettromotrice che si genera ai capi del condensatore di capacità C quando, all'istante t, la carica accumulata su una faccia delle sue armature è $Q(t)$.
Alla chiusura del circuito, ai capi del condensatore sarà presente una tensione:

$$V(t) = V_0(1 - e^{-\frac{t}{RC}}) \tag{22.2}$$

e nel circuito circolerà una corrente:

$$I(t) = I_0 e^{-\frac{t}{RC}} \tag{22.3}$$

Se invece il condensatore è carico con una tensione iniziale V_0 e chiudiamo il circuito escludendo il generatore di f.e.m, ai capi del condensatore sarà presente una tensione:

$$V(t) = V_0 e^{-\frac{t}{RC}} \tag{22.4}$$

e nel circuito circolerà una corrente:

$$I(t) = -I_0 e^{-\frac{t}{RC}} \tag{22.5}$$

22.2 Descrizione dell'esperienza

Procediamo costruendo l'esperienza per gradi, in modo da avere la sicurezza che tutti i componenti del circuito che utilizzeremo funzionino correttamente.

Misuriamo con un multimetro la resistenza R del resistore utilizzato, sia per la difficoltà nel leggere il valore della resistenza facendo riferimento al codice a colori in essa indicato, sia perché il valore nominale della resistenza ha un accuratezza massima solamente del 5%. Col multimetro possiamo conoscere il suo valore con una accuratezza dell'1%.

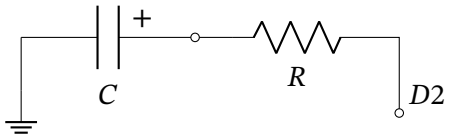
Per quanto riguarda il condensatore leggiamo direttamente il suo valore dalla sua etichetta. A questo punto, dopo il montaggio degli apparecchi, siamo pronti a leggere i dati dal monitor seriale dell'IDE di Arduino. Copieremo quei dati in un foglio elettronico e per poterli elaborare.

22.3 Materiale utilizzato

1. scheda Arduino Uno
2. breadboard
3. condensatore da 1000 μF
4. resistenza da 1000 ohm
5. cavetteria varia
6. multimetro

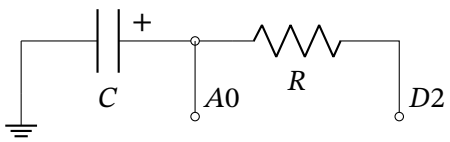
22.4 Montaggio delle apparecchiature

Lo schema di circuito sopra esposto può essere ricondotto al seguente.

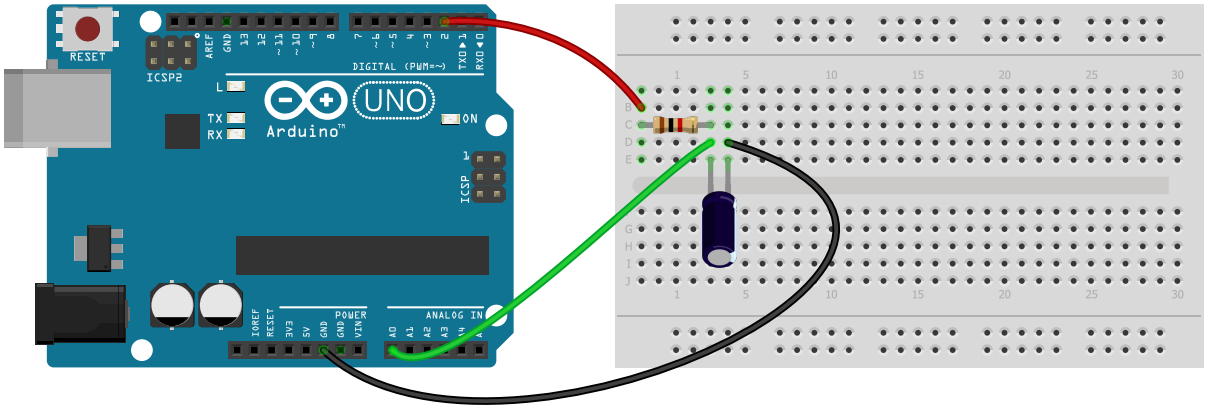


In questo non abbiamo più l'interruttore per aprire o chiudere il circuito. Non abbiamo neanche una batteria, ma i due capi del circuito sono uno a terra (quello a sinistra) e quindi ad un potenziale di riferimento di 0 V; l'altro, chiamato D2, verrà posto a 5 V collegandolo al pin D2 di Arduino.

Un circuito siffatto non serve ancora a niente se non facciamo delle misurazioni. Se vogliamo misurare la d.d.p. ai capi del condensatore doppiamo applicare ai suoi capi un voltmetro. In pratica possiamo compiere questa operazione con Arduino, collegando il capo positivo del condensatore ad un ingresso analogico. L'ingresso misurerà la tensione presente su quel pin rispetto allo zero di riferimento della scheda a cui è collegato il capo negativo del condensatore. Per ulteriore approfondimenti sulle misure di tensione si rimanda al paragrafo 11.1. Il circuito diventa quindi:



Uno schema di collegamento può essere quindi questo:



Il condensatore normalmente è di tipo elettrolitico, come disegnato nella figura. Questo genere di condensatore ha una polarizzazione preferenziale, ovvero un terminale (di solito segnato con una striscia colorata) va collegato al potenziale minore e l'altro al potenziale maggiore. Nel nostro caso il terminale negativo va collegato al pin del GND e l'altro è in comune con la resistenza.

22.5 Codice

Il seguente codice é diviso in tre parti.

1. Nella prima parte, tra le righe 19 e 30, accendiamo il pin dei 5 V e per per un certo numero di "passi" leggiamo al pin A0 (dotato di convertitore analogico digitale a 10 bit) la tensione ai capi del condensatore. *Siamo nella fase di carica*.
2. Nella seconda parte, tra le righe 31 e 41, spegniamo il pin dei 5 V e per per un certo numero di "passi" leggiamo al pin A0 la tensione ai capi del condensatore. *Siamo nella fase di scarica*.
3. Infine, righe 43 - 46, aspettiamo che il condensatore si sia pressoché completamente scaricato.

Per decidere quale intervallo di tempo utilizzare per i campionamenti calcoliamo la costante di tempo τ e fermiamo ogni "passo" dei due loop principali per un tempo pari a un decimo di τ.

```
1  unsigned long tempo_iniziale;
   unsigned long tempo;
3  float tensione;
   float tens_rif = 4.94;
5  float resistenza = 1189.0;
   float capacita = 0.001;
7  float tau;
   float tempo_osservazione;
9  int passi = 100;
   //
11 void setup() {
     Serial.begin(56000);
13   pinMode(2,OUTPUT);
     digitalWrite(2, LOW);
15   delay(1000);
     tau = resistenza * capacita * 1000; // in millisecondi
```

```
17    tempo_osservazione = tau /10;
  }
19  void loop() {
      digitalWrite(2,HIGH);
21    tempo_iniziale = micros();
      //
23    for (int i = 0; i < passi; i ++) {
        tempo = micros();
25      tensione = analogRead(A0)*tens_rif/1024.;
        Serial.print(tensione);
27      Serial.print("␣");
        Serial.println((tempo - tempo_iniziale)/1000000.0);
29      delay(int(tempo_osservazione));
      }
31    digitalWrite(2, LOW);
      tempo_iniziale = micros();
33    //
      for (int i = 0; i < passi; i ++) {
35      tempo = micros();
        tensione = analogRead(A0)*tens_rif/1024.;
37      Serial.print(tensione);
        Serial.print("␣");
39      Serial.println((tempo - tempo_iniziale)/1000000.0);
        delay(int(tempo_osservazione));
41    }
      //
43    while (fabs(analogRead(A0)*tens_rif/1024.) > 0.001) {
        Serial.println("␣-----␣mi␣sto␣scaricando␣------");
45    }
      Serial.println("*******************************");
47  }
```

1. riga 13: il pin 2 viene settato in uscita
2. riga 14: il pin 2 viene messo nello stato ”LOW”, tensione zero.
3. riga 15: aspetto un secondo
4. riga 16 calcolo il tau (in millisecondi)
5. riga 20: sezione relativa alla carica; “accendo” la tensione in uscita dal pin 2
6. riga 23: faccio un loop con un numero di step pari a ”passi”
7. riga 24: leggo il tempo in microsecondi
8. riga 25: leggo la tensione misurata sul pin A0 e la trasformo in un numero che rappresenta i volt misurati.
9. riga 26: stampo la tensione
10. riga 28: stampo il tempo trascorso dall’inizio della carica e trasformato in secondi
11. riga 29: mi fermo prima di procedere al nuovo step
12. riga 31: ”spengo” l’uscita al pin 2 e inizio il processo di scarica con gli stessi criteri uscita in quello di carica.
13. riga 43: quando la tensione misurata scende sotto i 0,001 V mi fermo

22.6 Procedimento

Una volta avviato Arduino e la fase di carica e scarica, osserviamola nel plotter seriale dell'IDE; se soddisfatti dell'andamento chiudiamo il plotter seriale e apriamo il monitor seriale. In questo disabilitiamo lo scorrimento automatico e selezioniamo una coppia di dati completa per carica e scarica. Incolliamo questi dati in un foglio elettronico. Facciamo attenzione che il separatore dei decimali è (all'inglese) il punto e le due coppie di dati sono separate da uno spazio bianco. Con queste due accortezze il foglio elettronico riconoscerà correttamente i dati. Una volta salvati i dati possiamo affiancarli alla curva teorica della tensione associata agli stessi istanti di tempo raccolti e con l'opportuna costante di tempo τ.

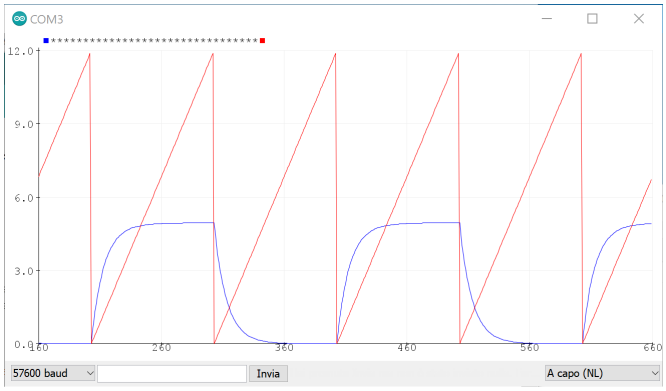
22.7 Dati sperimentali

La resistenza ha un valore nominale $R_n = 1\,\text{k}\Omega$ e un valore misurato col voltmetro $R = 1189\,\Omega$. La capacità ha un valore nominale $C = 1 \times 10^3\,\mu\text{F}$. Il convertitore analogico digitale che permette di ricavare la tensione ai capi del condensatore ha una risoluzione di 10 bit. Questo significa che su un fondo scala di 5 V l'accuratezza è al massimo di 0,005 V e in pratica circa 0,01 V: per questa ragione indichiamo i valori di tensione con sole due cifre decimali. Misurando la differenza di potenziale tra pin 2 e GND con un tester la tensione effettiva nella mia scheda è stata 4,94 V.

Questi i dati raccolti nei primi 10 step della fase di carica e di scarica:

	Tempo (s)	Tensione carica (V)	Tensione scarica (V)
1	0,00	0,00	4,93
2	0,12	0,49	4,45
3	0,24	0,91	4,02
4	0,36	1,30	3,65
5	0,48	1,64	3,30
6	0,59	1,94	3,00
7	0,71	2,23	2,72
8	0,83	2,48	2,47
9	0,95	2,71	2,24
10	1,07	2,91	2,03

Quello che segue è l'output che possiamo osservare nel plotter seriale dell'IDE di Arduino. In ascissa l'unità di misura è lo step di misurazione (da quando si avvia la raccolta dati); in ordinata abbiamo la curva con i dati della tensione, dal caratteristico andamento esponenziale, e delle rette inclinate relative ai dati del tempo (che cresce linearmente). Si può osservare che il tempo per la fase di carica e scarica è di circa 12 s; la tensione varia tra 0 V e 5 V.



22.8 Elaborazione dati sperimentali

In questa tabella abbiamo aggiunto il valore che risulterebbe dalla teoria, lo scostamento in percentuale tra il valore misurato e quello teorico $\left(100 \cdot \frac{V_t - V_m}{V_m}\right)$ e la differenza tra i due valori ($\Delta V = V_t - V_m$).

	Fase di carica				
	tempo (s)	tensione mi. (V)	tensione te. (V)	scostamento (%)	ΔV (V)
1	0,00	0,00	0,00	0,0	0,00 V
2	0,12	0,49	0,47	−3,3	−0,02 V
3	0,24	0,91	0,90	−0,8	−0,01 V
4	0,36	1,30	1,29	−0,7	−0,01 V
5	0,48	1,64	1,64	0,1	0,00 V
6	0,59	1,94	1,93	−0,4	−0,01 V
7	0,71	2,23	2,22	−0,4	−0,01 V
8	0,83	2,48	2,48	0,1	0,00 V
9	0,95	2,71	2,72	0,3	0,01 V
10	1,07	2,91	2,93	0,7	0,02 V

	Fase di scarica				
	tempo (s)	tensione mi. (V)	tensione te. (V)	scostamento (%)	ΔV (V)
1	0,00	4,93	4,93	0,0	0,00 V
2	0,12	4,45	4,46	0,2	0,01 V
3	0,24	4,02	4,03	0,2	0,01 V
4	0,36	3,65	3,64	−0,2	−0,01 V
5	0,48	3,30	3,29	−0,2	−0,01 V
6	0,59	3,00	3,00	0,1	0,00 V
7	0,71	2,72	2,71	−0,2	−0,01 V
8	0,83	2,47	2,45	−0,7	−0,02 V
9	0,95	2,24	2,22	−1,0	−0,02 V
10	1,07	2,03	2,00	−1,3	−0,03 V

22.9 Conclusioni

I dati raccolti sono in buon accordo con quanto previsto dalla teoria, con dei valori tendenzialmente in eccesso nella carica e in difetto nella scarica. Le differenze assolute tra valori teorici e misurati non superano mai 0,05 V. Costituisce un punto di criticità la non buona conoscenza della capacità reale del condensatore.

23 Studio del campo magnetico di un solenoide

23.1 Introduzione

Intorno a un filo percorso da corrente si forma un campo magnetico: l'intensità del campo è proporzionale alla corrente. Per produrre un campo abbastanza intenso, ma con la minima corrente, possiamo usare un filo avvolto in spire come quello detto *solenoide*. Nel solenoide ideale, di lunghezza infinita, il campo è uniforme all'interno del solenoide e nullo all'esterno. Se il solenoide ha N spire per una lunghezza l e siamo nel vuoto allora il campo all'interno vale:

$$B = \mu_0 \frac{N}{l} i \tag{23.1}$$

dove i è l'intensità di corrente che circola nel solenoide e μ_0 è la permeabilità magnetica del vuoto. Dalla formula qui scritta si può osservare che **il campo magnetico presente all'interno del solenoide è direttamente proporzionale alla corrente che lo attraversa**. In questa esperienza vogliamo studiare, mediante un sensore di campo magnetico, la validità di questa affermazione.

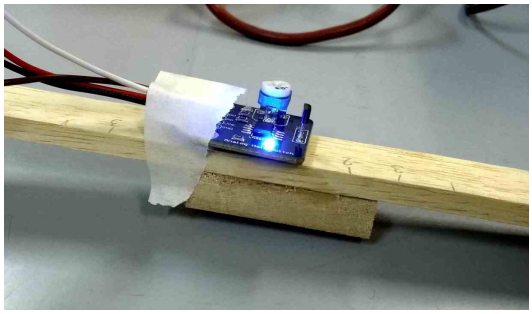
23.2 Apparecchi utilizzati

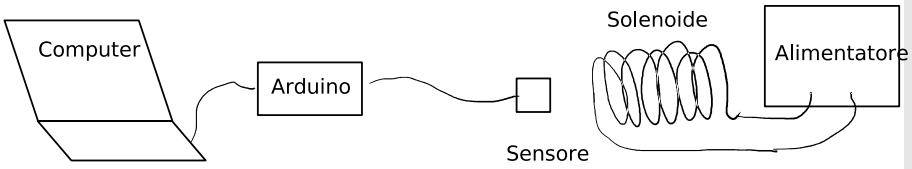
1. solenoide con 600 spire, lungo 6,5 cm;
2. righello millimetrato;
3. alimentatore in corrente continua da 300 VA in grado di erogare almeno 1 A;
4. sensore di campo magnetico Sunfounder ad effetto Hall per Arduino;
5. scheda Arduino Uno;
6. computer con il monitor seriale dell'IDE di Arduino;
7. cavi per il collegamento di Arduino e del sensore e del solenoide;
8. amperometro integrato nell'alimentatore:
   accuratezza 0,01 A o ±1% a fondo scala, portata 10 A;
9. asta di legno lunga circa 30 cm;
10. nastro adesivo;
11. terza mano o supporto per l'asta di legno.

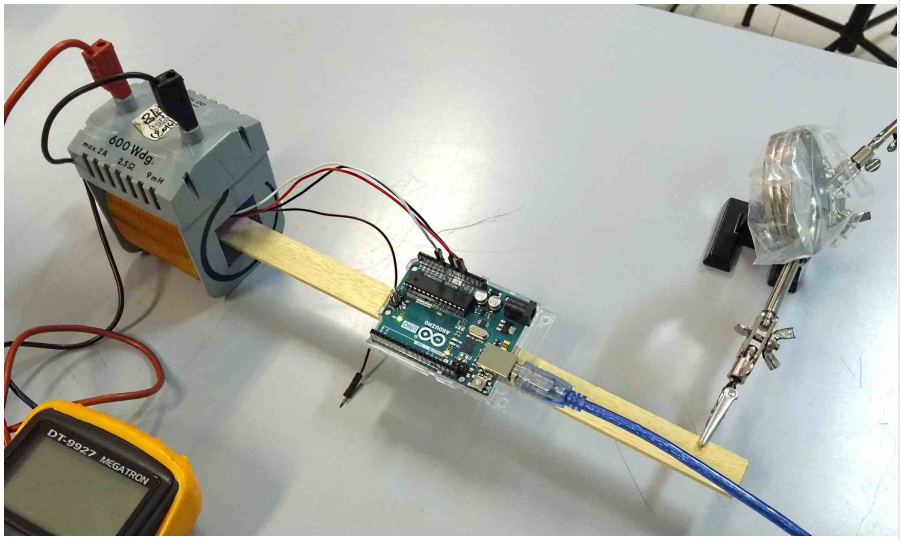
23.3 Montaggio apparecchi

Il sensore di campo magnetico utilizzato misura il campo magnetico in asse con la piccola scheda su cui è montato: lo poniamo all'interno del solenoide in asse con solenoide stesso. Tuttavia, per avere un posizionamento in asse e possibilmente al centro del solenoide, lo poggiamo su un'asta di legno (e quindi diamagnetica) su cui lo fermiamo con una striscia di nastro adesivo.

L'essere al centro del solenoide non è essenziale ai fini della proporzionalità che vogliamo osservare. Tuttavia è importante tenere il sensore fermo e in asse per tutta l'esperienza dal momento che esso misura il campo magnetico orientato solo perpendicolarmente al suo piano.







Dopo aver collegato il sensore di campo magnetico e la scheda Arduino al computer avviamo l'IDE di Arduino e apriamo il monitor seriale. In esso leggeremo i valori del campo magnetico misurato indirettamente, tramite una porta analogica di Arduino: leggeremo quindi un numero che, per un campo magnetico nullo è circa 500.

23.3.1 Codice per l'utilizzo

Questa è una versione semplificata del codice che abbiamo presentato per il sensore Sunfounder. Non utilizziamo il pin legato al led del sensore e stampiamo il valore letto dalla porta analogica di Arduino in modo che con un campo nullo dia un valore nullo.

```
1  int sensorPin = A0;
   int sensorValue = 0;
3  void setup()
   {
5    Serial.begin(9600);
   }
7  void loop()
   {
9    sensorValue = analogRead(sensorPin);
     Serial.print("Valore del sensore   ");
11   Serial.print(sensorValue-511);
     Serial.print("   ");
13   Serial.println(sensorValue);
     delay(200);
15 }
```

- La riga 2 seleziona il pin di input per il potenziometro.
- La riga 3 nomina la variabile che contiene il valore letto dalla porta A0.
- La riga 5 seleziona la velocità di comunicazione tra Arduino e l'esterno.
- La riga 8 legge il valore di A0.
- La riga 11 stampa sulla porta seriale il valore del segnale inviato dal sensore riscalato. Il valore qui sottratto (511) è specifico dell'esemplare di sensore in mio possesso. Ognuno deve inserire il valore specifico per il proprio apparecchio
- La riga 14 mette Arduino in pausa per 200 ms.

23.4 Procedimento

Iniziamo col leggere il campo magnetico fuori dal solenoide, lontano da fonti di campo: il sensore deve segnare zero o un valore fluttuante intorno allo zero. Se così non fosse modifichiamo il programma installato su Arduino o al limite riscaliamo tutte le misure successive tenendo conto di questo valore base indicato dal sensore, valore che può essere positivo o negativo.

Successivamente poniamo il sensore dentro il solenoide quando questo non è ancora collegato all'alimentatore: il sensore deve indicare ancora zero. Se così non fosse controlliamo la qualità del collegamento elettrico del sensore: ci sono sicuramente dei contatti laschi o traballanti. Qualche colpetto ai fili nei punti di contatto o dei piccoli movimenti alla scheda Arduino o al sensore normalmente risolvono il problema.

Volendo possiamo orientare il solenoide perpendicolarmente con il campo magnetico terrestre in modo che il sensore non venga influenzato nelle misurazioni. Tuttavia, come abbiamo già detto, la sensibilità del sensore è inferiore al valore del campo magnetico terrestre.

Adesso accendiamo l'alimentatore, assicurandoci che tutti i suoi potenziometri siamo a zero: l'alimentatore appena acceso non deve erogare alcuna corrente.

Dopodiché portiamo la corrente a 0,1 A e rimisuriamo il campo magnetico. Proseguiamo fino a raggiungere 0,8 A a passi di 0,1 A. Potrebbe succedere che la misura del campo magnetico indicata dal sensore non sia del tutto stabile: è normale. In quel caso prendiamo la misura che compare più spesso: in ogni caso le fluttuazioni sono dell'ordine della sensibilità dello strumento.

Una volta finito con queste coppie di valori azzeriamo nuovamente la corrente, spegniamo l'alimentatore per sicurezza e invertiamo i cavi ai capi del solenoide. Cominciamo nuovamente a prendere una nuova serie di coppie di valori corrente / campo magnetico fino ad arrivare a 0,8 A. In questa nuova serie la corrente circola in verso opposto rispetto alla serie precedente; il campo magnetico prodotto dal solenoide ha anch'esso verso opposto; il sensore dovrebbe segnare valori simili ai precedenti, ma di segno opposto.

Infine determiniamo la retta dei minimi quadrati con i valori ottenuti e verifichiamo che sono linearmente correlati.

23.5 Dati sperimentali

$N = 600$ $l = 65\,\text{mm}$ $\Delta l = 1\,\text{mm}$ $\Delta i = 0{,}01\,\text{A}$

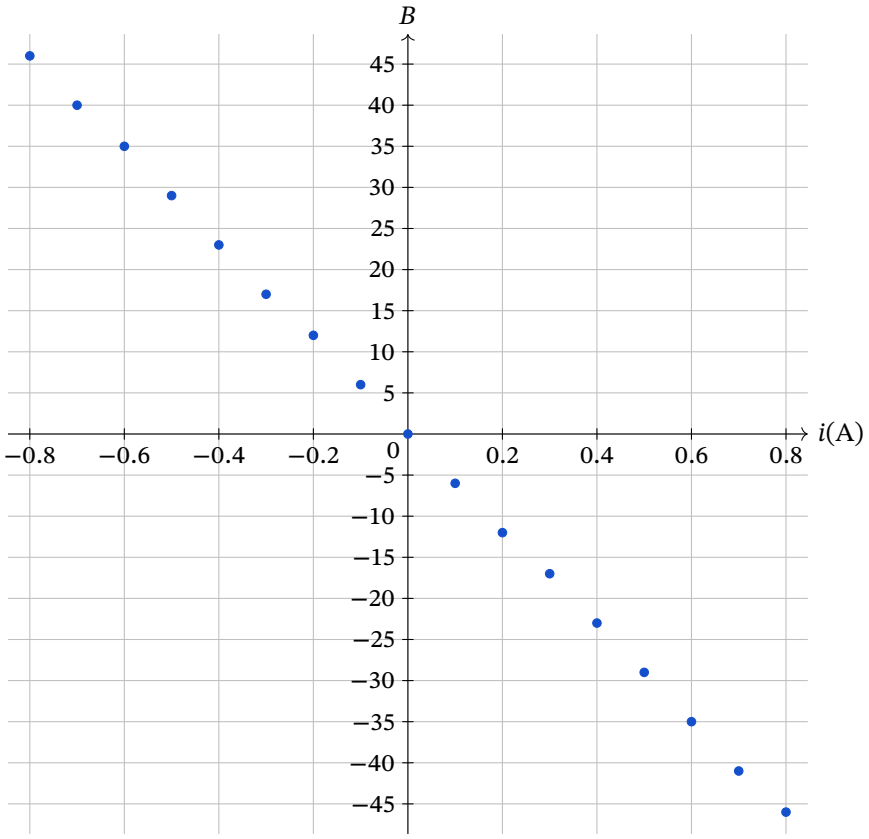
I Serie	
i (A)	B
0,00	0
0,10	−6
0,20	−12
0,30	−17
0,40	−23
0,50	−29
0,60	−35
0,70	−41
0,80	−46

II Serie	
i (A)	B
0,00	0
−0,10	6
−0,20	12
−0,30	17
−0,40	23
−0,50	29
−0,60	35
−0,70	40
−0,80	46

I valori indicati per il campo magnetico non hanno unità di misura: sono i valori letti direttamente dalla porta analogica di Arduino.

Nella seconda serie l'intensità di corrente è negativa perché abbiamo invertito il suo verso rispetto alla prima serie. Il campo magnetico è negativo se la parte anteriore del sensore è posta in prossimità di un polo nord magnetico, positivo altrimenti.

23.6 Elaborazione dati sperimentali



Il grafico ottenuto è una retta passante per l'origine: ciò conferma l'ipotesi fatta. Determiniamo anche la retta dei minimi quadrati e il coefficiente di correlazione associato ai dati sperimentali per confermare quantitativamente l'ipotesi della diretta proporzionalità.

La retta interpolatrice è $y = mx + q$, dove ad x associamo l'intensità di corrente i e a y associamo il campo magnetico misurato B.

m (1/A)	Δm (1/A)	q	Δq	r
−57,8	0,2	−0,06	0,08	0,9998

L'indice ci correlazione dato indica che, con grande confidenza, il campo magnetico è linearmente dipendente dalla corrente applicata. Il valore molto piccolo del parametro q indica che anche la retta interpolatrice passa per lo zero.

Quindi campo magnetico e corrente sono direttamente proporzionali.

23.7 Note

Inizialmente avevo provato a determinare anche il valore in tesla del campo magnetico, ma i risultati sperimentali non erano in linea con quanto atteso. Approfondendo la validità della relazione che ci indica l'intensità del campo magnetico in un solenoide ideale e acquistando anche un sensore che mi desse direttamente i valori del campo, ho potuto appurare quanto il modello del solenoide ideale si discosti dalla realtà. A quel punto ecco le ragioni per studiare solo la linearità del campo rispetto all'intensità di corrente.

24 Studio del campo magnetico di un magnete

24.1 I magneti permanenti

Un magnete permanente è un oggetto di materiale ferromagnetico che, precedentemente sottoposto ad un campo magnetico di sufficiente intensità, tende a rimanere nello stato di magnetizzazione assunto. La forma del magnete può essere qualsiasi. I magneti più frequentemente presenti nei laboratori scolastici hanno la forma o di sbarra piatta o di ferro di cavallo.

La descrizione quantitativa del campo intorno ad un magnete è sempre molto complessa. Come riferimento possiamo considerare il campo lungo l'asse di una spira circolare piana percorsa da corrente, campo al centro della spira o nei suoi pressi:

$$B(x) = \frac{\mu_0 i R^2}{2(R^2 + x^2)^{3/2}} \tag{24.1}$$

dove R è il raggio della spira, i la corrente che l'attraversa, x la distanza dal piano della spira.

Se siamo a distanze superiori alla misura del raggio possiamo approssimare la formula precedente:

$$B(x) = \frac{\mu_0 i R^2}{2x^3} = k\frac{1}{x^3} \tag{24.2}$$

I magneti che useremo sono di forma cilindrica, uno di lunghezza inferiore al diametro e un altro molto più lungo. Il comportamento del campo di questi due magneti è simile a quello generato da spira piana o da un solenoide. Non conosciamo l'espressione funzionale precisa del campo magnetico intorno a questi due oggetti, ma ipotizzeremo che il campo abbia l'andamento descritto dall'ultima relazione.

24.2 Descrizione dell'esperienza

In questa esperienza vogliamo studiare l'andamento del campo magnetico intorno a due magneti di forma diversa in funzione della distanza. Cercheremo di determinare anche una legge che descriva l'andamento osservato. Useremo un sensore di campo magnetico ad effetto Hall per misurare l'intensità del campo magnetico e un righello per misurare la distanza tra magnete e sensore.

Sul modello del campo lungo l'asse di una spira percorsa da corrente ipotizziamo che il campo magnetico segua una legge di potenza del tipo:

$$B(x) = \frac{k}{x^\alpha} \tag{24.3}$$

Quello che vogliamo determinare con i dati sperimentali è il miglior valore di α e ci aspettiamo che sia prossimo a 3. Per rispondere a questo obbiettivo usiamo il metodo dei minimi quadrati. Con questo metodo possiamo trovare il miglior valore dei parametri di una retta che si adatti ai dati sperimentali. Nel nostro caso i dati non sono sicuramente descritti da una retta, ma ad essa possiamo

ricondurci. Infatti proviamo a calcolare il logaritmo naturale dei due membri dell'ultima relazione scritta:

$$\begin{aligned} \ln(B(x)) &= \ln\left(\frac{k}{x^{\alpha}}\right) \\ \ln(B(x)) &= \ln k - \ln x^{\alpha} \\ \ln(B(x)) &= -\alpha \cdot \ln x + \ln k \end{aligned} \tag{24.4}$$

L'ultima relazione è assimilabile ad una retta di equazione:

$$y = mx + q \tag{24.5}$$

dove $\ln(B(x))$ é y; $-\alpha$ è m cioè il coefficiente angolare della retta; $\ln x$ è x; $\ln k$ è q e quindi $k = e^{q}$.

Il metodo dei minimi quadrati ci consentirà di determinare i parametri di questa retta e da essi quelli della nostra funzione di potenza. Osservando che m risulterà negativo possiamo scrivere che il nostro modello è:

$$B(x) = e^{q} x^{m} \tag{24.6}$$

Per verificare il grado di concordanza tra modello e dati sperimentali possiamo usare il coefficiente di correlazione r: questo è un parametro che si ricava dal metodo e il cui valore è prossimo ad uno quando si verifica una buona concordanza.

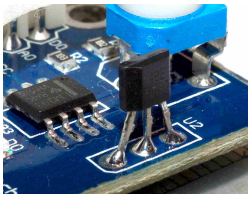
24.3 Apparecchi utilizzati

1. Arduino uno;
2. sensore di campo magnetico basato su integrato SS49E;
3. righello millimetrato (20 o 30 cm);
4. nastro adesivo;
5. due magneti al neodimio cilindrici:
   - magnete *corto*; $l = 15\,\text{mm}$, $h = 8\,\text{mm}$;
   - magnete *lungo*; $l = 10\,\text{mm}$, $h = 40\,\text{mm}$;

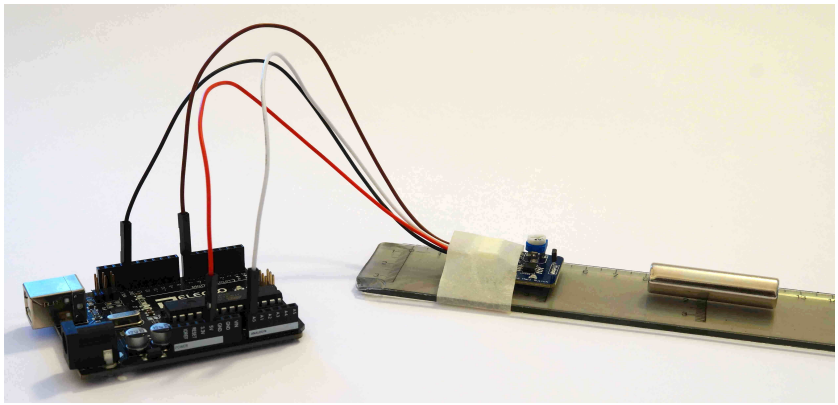
24.4 Montaggio delle apparecchiature

Per quanto riguarda il montaggio di Arduino con il suo sensore useremo lo schema descritto nel capitolo sulla misura di campi magnetici. Posizioniamo il sensore su l'estremità del righello con il sensore verso l'altra estremità e fissiamo il sensore con del nastro adesivo. Sempre sull'asta posizioniamo il magnete con il suo nord o sud magnetico verso il sensore e teniamolo fisso col nastro adesivo: ogni volta che faremo una nuova misurazione metteremo il magnete ad una nuova distanza dal sensore.

(a) magneti



(b) sensore



Codice per Arduino
Il codice per Arduino è lo stesso utilizzato nel capitolo sulla misura dei campi magnetici.

24.5 Procedimento

La sensibilità minima di Arduino non dovrebbe permettere di rilevare il campo magnetico terrestre. Ciononostante posizioniamo l'asta in direzione est-ovest, in modo che il nord-sud del magnete sia perpendicolare al campo terreste e che quest'ultimo non influenzi le misure.

Il valore del campo magnetico dei due poli dovrebbe essere idealmente lo stesso a parità di distanza dal magnete. Lo verifichiamo misurando il campo dei due poli ogni volta per ogni distanza. Questo metodo ha anche lo scopo di minimizzare l'eventuale isteresi del sensore con la continua inversione del campo applicato. Questa isteresi non è indicata tra le specifiche del circuito integrato, ma è sempre presente in sensori simili e viene valutata dello stesso ordine di grandezza della sensibilità fornita con Arduino.

24.6 Dati sperimentali

Riportiamo i dati relativi al campo magnetico senza unità di misura per semplicità, con solo le indicazioni numeriche che vengono da Arduino.

magnete corto

	Distanza (mm)	*B* Nord	*B* Sud
1	102	2	−1
2	92	3	−1
3	82	4	−2
4	72	5	−4
5	62	7	−5
6	52	10	−9
7	42	18	−17
8	32	39	−37
9	22	100	−101
10	12	330	−344

magnete lungo

	Distanza (mm)	*B* (Nord)	*B* (Sud)
1	122	2	−1
2	112	3	−1
3	102	3	−2
4	92	4	−2
5	82	5	−3
6	72	6	−4
7	62	9	−8
8	52	12	−11
9	42	20	−19
10	32	37	−36
11	22	78	−79
12	12	232	−234

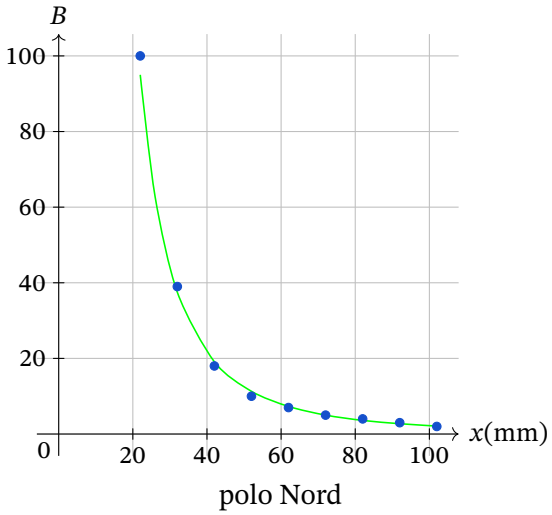
Abbiamo preso i dati dalla distanza in cui il campo magnetico risultava diverso dal valore di fondo. Per le distanze maggiori nei dati che provengono dal sensore si può osservare qualche fluttuazione intorno all'unità. Per le distanze minori l'incertezza è legata alla rapida variazione del campo per minime variazioni della distanza. L'errore associato alla distanza è 1 mm; l'errore associato al campo magnetico è 1 per valore inferiori a 100 e di diverse unità per valori superiori.

24.7 Elaborazione dati sperimentali

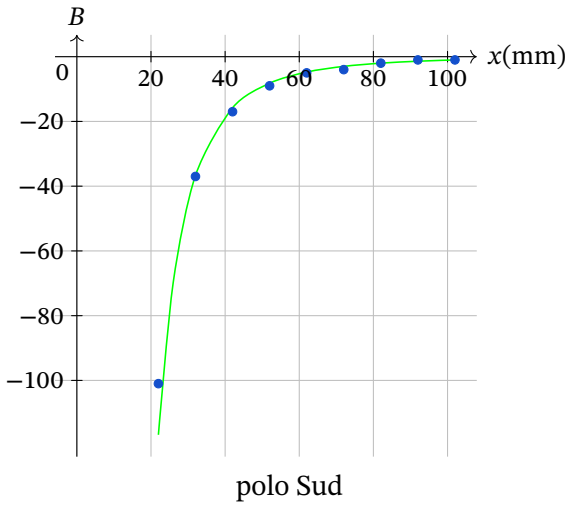
Osserviamo subito che il valore ottenuto per la distanza più piccola, nel caso del magnete corto, va oltre la regione di linearità dello strumento e non l'abbiamo preso in considerazione nei calcoli successivi. Calcoliamo la retta dei minimi quadrati con il metodo illustrato in appendice[175] e rappresentiamo in un grafico i dati sperimentali e la curva interpolante. Non abbiamo tenuto conto degli errori associati alle misure nel calcolo con i minimi quadrati perché la sua valutazione andava al di là del livello di dettaglio utilizzabile in questo tipo di esperienza.

magnete corto

	m	Δm	q	Δq	r
polo Nord	−2,48	0,05	12,2	0,2	0,997
polo Sud	−3,08	0,13	−14,3	0,5	0,988



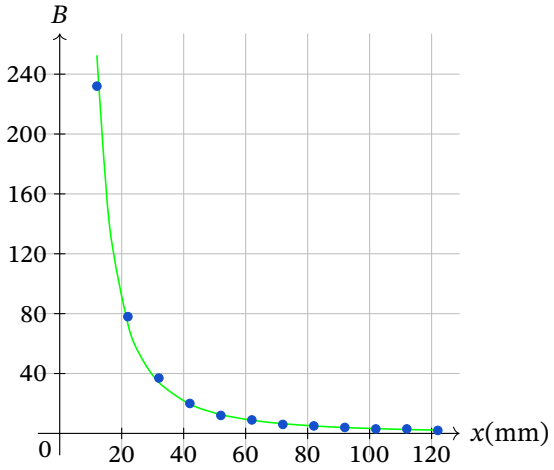
polo Nord



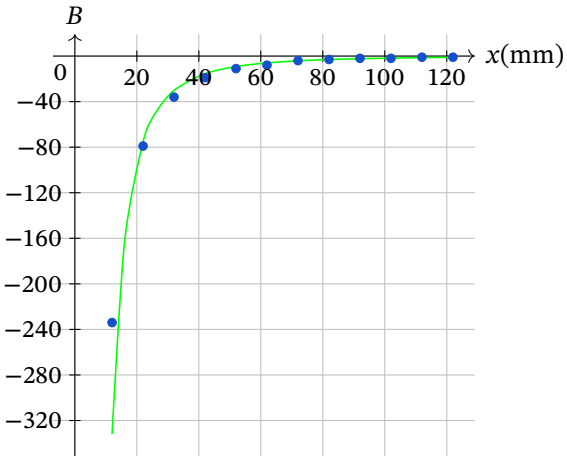
polo Sud

magnete lungo

	m	Δm	q	Δq	r
polo Nord	−2,04	0,03	10,6	0,13	0,997
polo Sud	−2,44	0,09	−11,8	0,4	0,989



polo Nord



polo Sud

24.8 Conclusioni

1. Le misure ottenute ci hanno dato una legge di potenza simile a quella prevista dal modello teorico. Il magnete corto ha fornito risultati più allineati al modello: quel magnete è più simile ad una spira piana di quanto non lo sia il magnete lungo.
2. Le misure relative al polo Nord e Sud dei magneti sono molto simili: a parità di distanza le misure differiscono solitamente di una sola unità ovvero la sensibilità dello strumento. Tuttavia si osserva che col polo Sud si parte da valori che differiscono per difetto e si termina con valori che differiscono per eccesso: non sappiano se sia un problema strumentale o cos'altro.
3. La potenza che compare con i poli Nord è sensibilmente più bassa di quella dei poli Sud, pur essendo le serie di dati simili: ciò é associabile all'ultima osservazione del punto precedente.
4. L'indice di correlazione per tutte le serie di dati è molto buono.

24.9 Note

Per chi volesse usare un calibro a corsoio per misurare le distanze osserviamo che normalmente si tratta di oggetti di materiale ferromagnetico. Non appena posti in prossimità del magnete si magnetizzano e rimangono solidalmente attaccati al magnete, rendendo la misura molto difficoltosa.

Parte IV

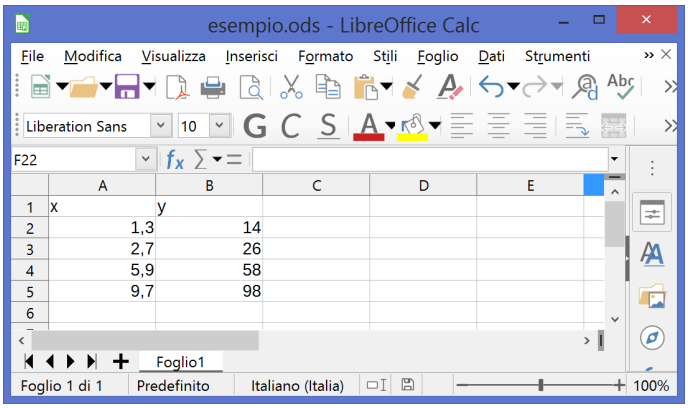
Appendici

A Retta dei minimi quadrati con Libreoffice

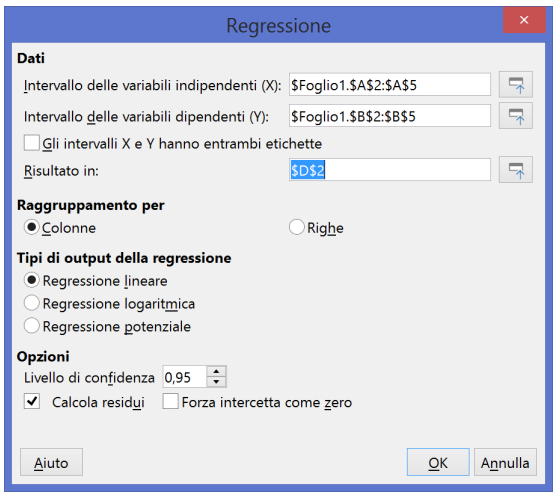
La retta dei minimi quadrati è un procedimento con cui si adatta una coppia di dati ad un modello descritto da una relazione lineare $y = mx + q$. Naturalmente questo adattamento ha senso se la relazione lineare è veramente descrittiva dei dati a disposizione. Il metodo consente anche di determinare il grado di concordanza tra dati e modello e l'errore associabile ai parametri della retta trovata. In questo manuale non illustriamo come calcolare la retta dai dati grezzi con i calcoli dettagliati, ma solo come ottenere i parametri usando un foglio elettronico. Useremo il programma Libreoffice perché gratuito, opensource e perfettamente adatto alle nostre esigenze.

Supponiamo di avere i seguenti dati: inseriamoli in due colonne vicine in libreoffice calc.

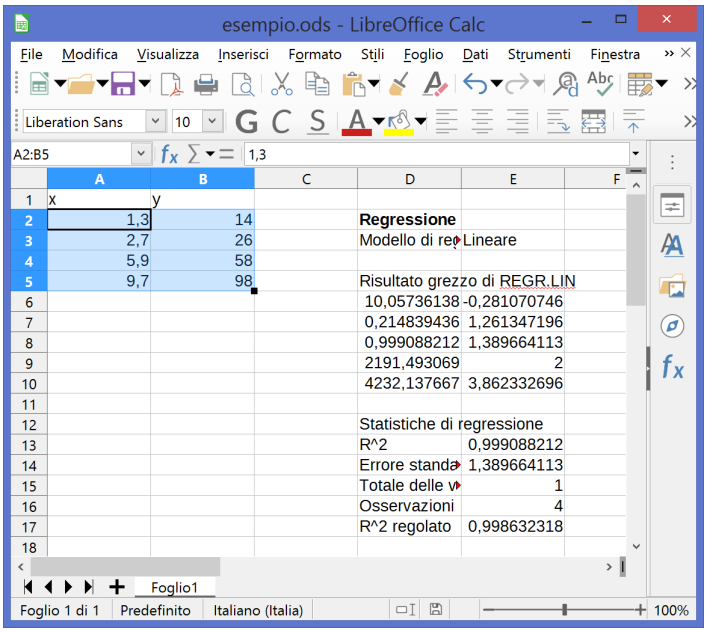
	x	y
1	1,3	14
2	2,7	26
3	5,9	58
4	9,7	98



Adesso selezioniamo le due colonne e da menu scegliamo nell'ordine le voci Dati -> Statistiche -> Regressione: comparirà la seguente finestra.



Il campo ”Risultato in:” risulterà vuoto: io ho inserito le coordinate della cella in cui voglio che compaiano i risultati del calcolo della retta di regressione. Dopo aver concluso pigiamo il tasto OK e compariranno i seguenti risultati:



- Cella D6 : coefficiente angolare m della retta di regressione
- Cella E6 : parametro q della retta di regressione
- Cella D7 : errore associato a m
- Cella E7 : errore associato a q
- Cella D8 : coefficiente di correlazione, riportato anche nella cella E13

Gli altri dati presenti sono innumerevoli e vanno al di là delle esigenze di questo manuale.

A.1 Legge di potenza

Il metodo dei minimi quadrati si può usare anche per relazioni non lineari come ad esempio la legge di potenza:

$$y = kx^{\alpha} \tag{A.1}$$

Come già illustrato nel capitolo sullo studio del campo magnetico possiamo fare il logaritmo naturale dei due membri della precedente relazione trasformandola in una relazione lineare.

$$\begin{aligned} \ln y &= \ln(kx^{\alpha}) \\ \ln y &= \ln k + \ln(x^{\alpha}) \\ \ln y &= \alpha \cdot \ln x + \ln k \end{aligned} \tag{A.2}$$

dove $\ln y$ corrisponde a y, α a m cioè il coefficiente angolare della retta e $\ln x$ a x; k lo possiamo ricavare facendo l’esponenziale di $\ln k$.

Nel nostro foglio elettronico inseriremo altre due colonne in cui otterremo il logaritmo naturale scrivendo nelle celle ”=ln(indirizzo della cella di partenza)”. Queste due nuove colonne saranno la base per il calcolo di regressione come illustrato nel paragrafo precedente.

B Arduino indipendente

In tutte le esperienze di questo documento abbiamo usato Arduino nella maniera più semplice, con la scheda sempre connessa ad un PC, sia nella fase di sviluppo del codice che in quella della raccolta dati. C'è stata un'unica eccezione nell'esperienza sulla misura della densità dell'aria, dove portarsi appresso almeno un portatile per tutte le misure era un po' troppo scomodo. Comodità a parte illustriamo qui di seguito il metodo che abbiamo usato per rendere Arduino, nella fase finale di raccolta dati, indipendente ed autonomo da qualsiasi altro computer.

Ciò di cui abbiamo bisogno è una fonte di alimentazione e uno schermo in cui visualizzare i dati.

B.1 Alimentazione

Esistono varie possibilità per alimentare Arduino. La più classica è quella di collegare con un porta pile un gruppo di pile al connettore di alimentazione. Tuttavia le pile non costano pochissimo, in particolare quelle ricaricabili, e non ci sono particolari garanzie per la stabilità dell'alimentazione.

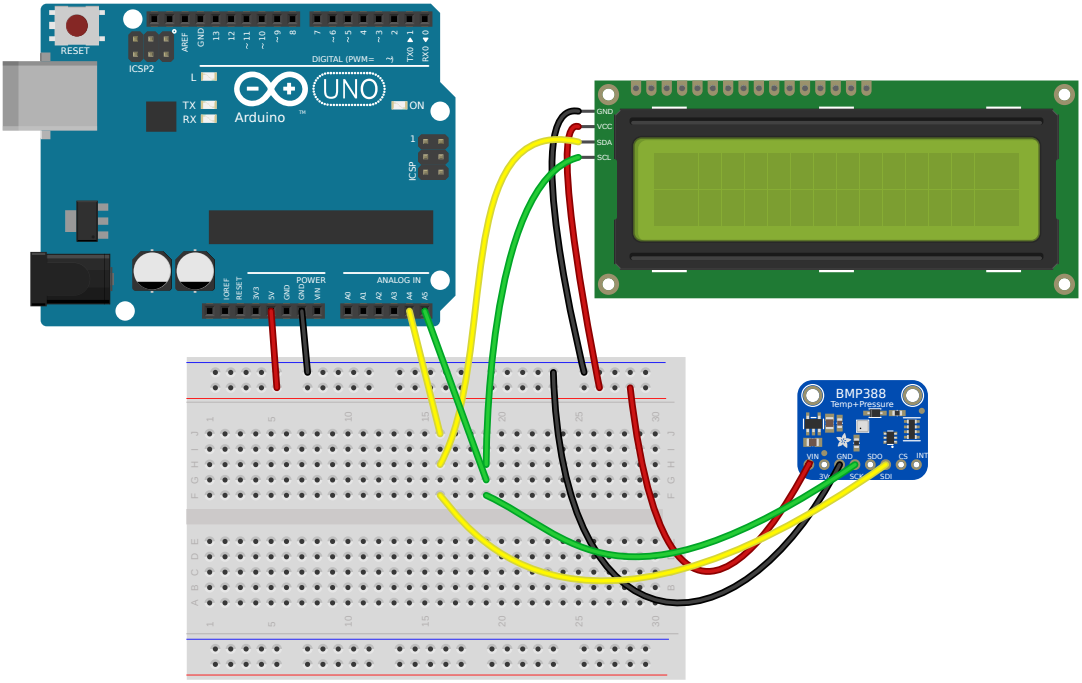
Il metodo che ho usato e consiglio è quello di acquistare un power bank per cellulari, uno di piccola capacità (il mio è da 6000 mA), e collegarlo direttamente alla presa usb di Arduino. Il prezzo dei modelli base è paragonabile all'acquisto di batterie ricaricabili, ma con molta maggiore praticità anche nella fase di ricarica, dato che tutti abbiamo a disposizione un caricabatteria per cellulari, ma pochi hanno già un caricabatteria per batterie in formato stilo o ministilo. La tensione in uscita da questi dispositivi è sempre stabilizzata. La corrente erogabile (con un minimo nominale di 1 A) dovrebbe essere sempre adeguata anche con i dispositivi di minor qualità.

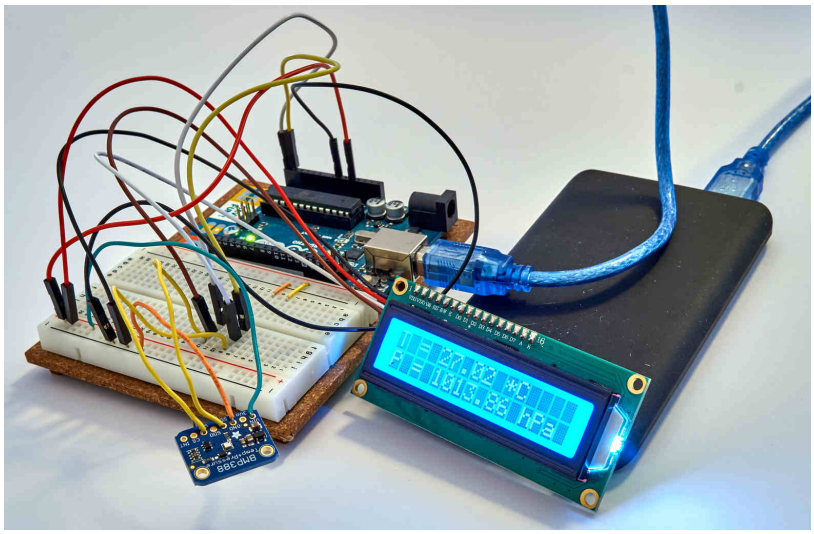
B.2 Esempio di applicazione

Come esempio di applicazione colleghiamo anche il display LCD al sistema [63] creato per l'esperienza sulla misura della densità dell'aria.

B.2.1 Montaggio delle apparecchiature

Il sensore e il display hanno entrambi quatto pin e usano la comunicazione I2C: quindi devono essere collegati entrambi ai pin A4 e A5 di Arduino Uno e condivideranno l'alimentazione. I dispositivi con interfaccia I2C sono predisposti normalmente per comunicare su due indirizzi predefiniti: se uno dei due è occupato selezionano automaticamente l'altro.





Codice per Arduino

Il seguente codice mette insieme quanto precedentemente usato con qualche linea in più per avere in output i dati anche sul display LCD.

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BMP3XX.h"
#include <LiquidCrystal_I2C.h>

Adafruit_BMP3XX bmp; // I2C

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 16, 2);

void setup() {
  Serial.begin(115200);
  while (!Serial);
  Serial.println("BMP388 test");

  if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP3 sensor, check wiring!");
    while (1);
  }

  // Set up oversampling and filter initialization
  bmp.setTemperatureOversampling(BMP3_OVERSAMPLING_2X);
  bmp.setPressureOversampling(BMP3_OVERSAMPLING_16X);
  bmp.setIIRFilterCoeff(BMP3_IIR_FILTER_COEFF_7);
  //bmp.setOutputDataRate(BMP3_ODR_50_HZ);

  lcd.init();
  lcd.backlight();
}

void loop() {
  if (! bmp.performReading()) {
    Serial.println("Failed to perform reading :(");
    return;
  }
  Serial.print("Temperature = ");
  Serial.print(bmp.temperature);
  Serial.println(" *C");

  Serial.print("Pressure = ");
  Serial.print(bmp.pressure / 100.0);
  Serial.println(" hPa");

  Serial.print("Approx. Altitude = ");
  Serial.print(bmp.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");

  Serial.println();

  lcd.clear();
  lcd.setCursor(0, 0);
```

```
    lcd.print("T = ");
53  lcd.print(bmp.temperature);
    lcd.print(" °C");
55  lcd.setCursor(0, 1);
    lcd.print("P = ");
57  lcd.print(bmp.pressure / 100.0);
    lcd.print(" hPa");
59  delay(2000);
}
```

C Codici per gnuplot

Per poter fare gli istogrammi con i dati raccolti nello studio delle performance di diversi sensori è stato usato il programma gnuplot col seguente codice.

```
set term qt size 900,600 # 900 pixels per 600 pixels
set terminal qt font "STIX Two Text,14"
set grid
set xlabel "distanza (mm)"
set ylabel "conteggi"
set yrange [0:]
set xrange [390:410]
binwidth=1
set boxwidth binwidth absolute
bin(x,width)=width*floor(x/width)
set style fill solid 0.5 # fill style
plot "data40.dat" using (bin($1,binwidth)) smooth freq with boxes notitle
```

D Alimentazione

Alcuni sensori per Arduino necessitano di correnti di alimentazione che possono mettere in crisi le possibilità di Arduino. Il problema si presenta anche con schede della stessa tipologia, ma più sofisticate, dove la corrente massima erogabile è comunque simile. Per ovviare a questi limiti si può alimentare la scheda in maniera alternativa alla presa USB. Oppure si possono alimentare direttamente i sensori o altri apparecchi da collegare poi ad Arduino.

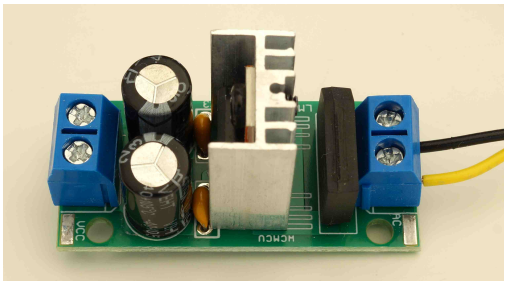
D.1 Alimentazione dei sensori con alimentazione esterna

Oltre a fornire direttamente corrente ad Arduino possiamo fornirlo ai soli sensori o apparecchi collegati. Per fornire 5 V o 3,3 V stabili possiamo usare un generatore anche in corrente alternata o poco stabile, interfacciandolo con un regolatore di tensione o un convertitore DC - DC.

Un *regolatore di tensione* è un dispositivo che viene alimentato con una tensione superiore a quella che si vuole ottenere; la potenza in eccesso viene tuttavia dissipata come calore. Pur garantendo una uscita particolarmente stabile questi dispositivi non sono opportuni se la tensione con cui gli alimentiamo è significativamente diversa da quella che vogliamo ottenere.

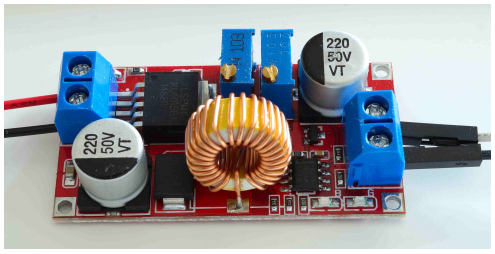
Tra i circuiti in grado di fornirci 5 V è particolarmente diffuso il regolatore di tensione LM 7805. Come si vede nella figura è dotato di tre terminali: uno va collegato a massa (GND), uno alla tensione di ingresso e uno a quella di uscita; si vede inoltre che è realizzato su una placca metallica che funge da dissipatore.

Le dimensioni di questa placca non sono tuttavia adeguate per le potenze che è costretto a dissipare anche in condizioni normali. É possibile utilizzarlo da solo, ma ritengo sia preferibile acquistarlo già completo di un dissipatore integrativo e con alcuni condensatori di supporto, come nel dispositivo seguente, dove sono presenti anche degli utili connettori per cavi spellati e un raddrizzatore di corrente per utilizzarlo anche con corrente alternata.



Per poterlo alimentare con un vecchio trasformatore con uscita a 9 V o 12 V esistono degli adattatori appositi come quello qui di seguito rappresentato.

Un *convertitore DC - DC* è un dispositivo che viene alimentato con una tensione superiore o inferiore a quella che si vuole ottenere. Al contrario del regolatore ha una efficienza molto elevata, fino al 95%. In apparecchi come quello indicato in figura sono presenti delle viti di regolazione (sopra i due componenti blu) per regolare la tensione e la corrente massima in uscita. Il loro funzionamento tuttavia non permette regolazioni finissime e l'affidabilità generale del dispositivo di regolazione non è particolarmente buona. Se dobbiamo lavorare con un progetto con Arduino consiglierei di utilizzare dispositivi con uscita fissa.



E Costi dei sensori e dei componenti

Qui di seguito un elenco di materiali proposti in questo documento, con i relativi costi.

1. Arduino uno: 24,20€
2. Arduino compatibile (Elegoo): 12,00€
3. breadboard: 4,50€
4. HC-SR04: 2,00/5,00€
5. HY-SRF05: 2,70€
6. TMP36: 2,29€
7. BMP180: 2,23€
8. BMP388: 13,82€
9. Adafruit MPRLS: 14,95$
10. SS49E: 3,15€
11. DS18B20: 2,10 – 5,00€
12. IR obstacle sensor - Sunfounder: 9,00€
13. IR break beam sensor - Adafruit: 3,59€
14. KS0320 Keyestudio Electromagnet Module: 11,99€
15. Display LCD 1602: 5,00€
16. VL53L1X CQrobot: 21,00€
17. AZDelivery KY-024: 2,50€
18. Adafruit Tlv493d: 6,00€

Indice analitico