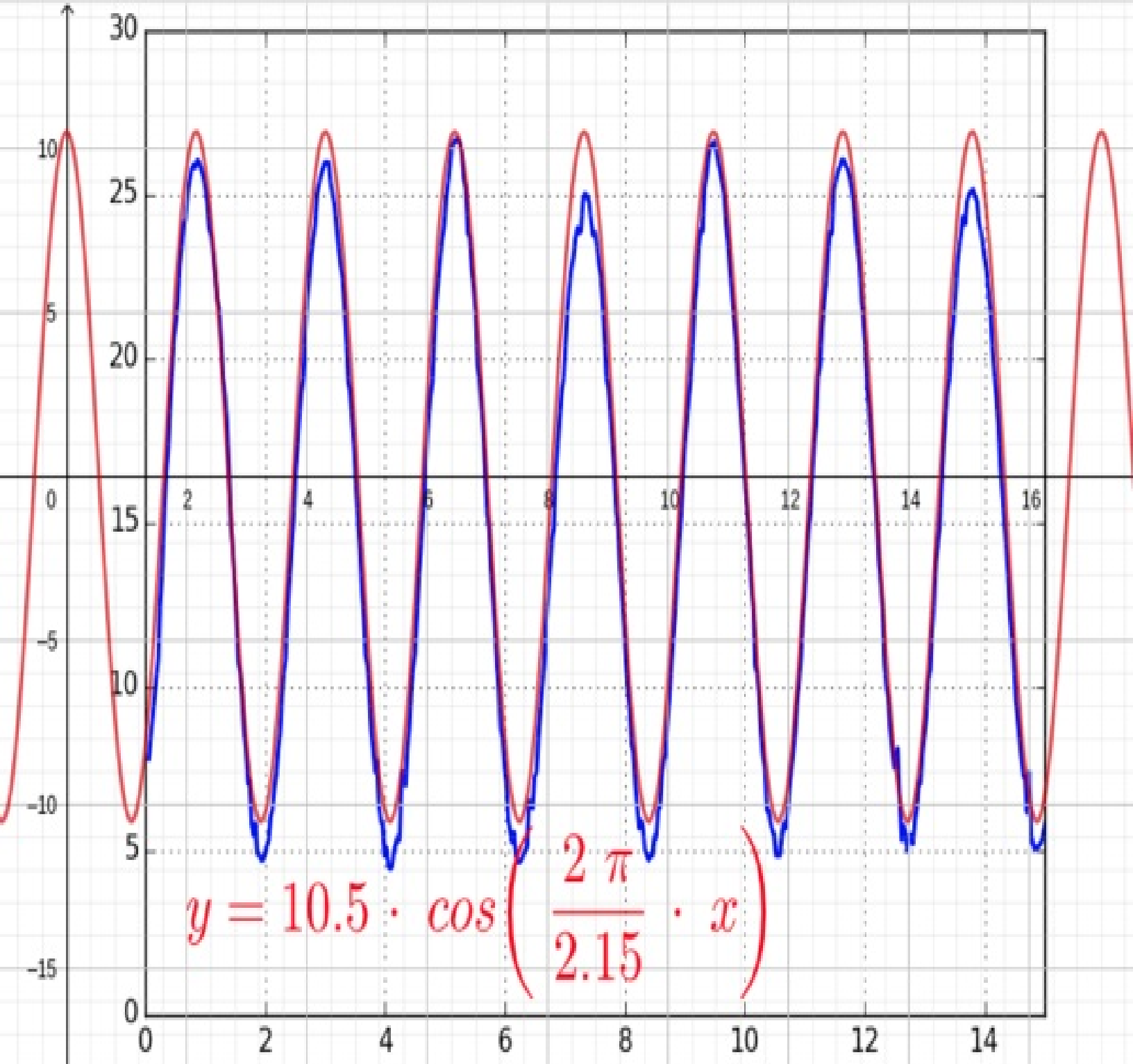


Laboratorio di Fisica *Open*

Liceo “ Renato Donatelli”

Terni

'Il grande libro della natura giace sempre aperto di fronte ai nostri occhi e in esso è scritta la vera filosofia...Ma ci è impossibile leggerlo se prima non abbiamo compreso la lingua e i caratteri in cui è stato scritto....E' scritto nella lingua della matematica e i suoi caratteri sono triangoli, i cerchi e le altre figure geometriche.'

Galileo Galilei

''Costruire i concetti della matematica, applicarne gli algoritmi, impararne i segreti strategici, farne la propria comunicazione, imparare a scegliere i registri semiotici più opportuni, facendo trasformazioni, è un'attività fortemente creativa che, però, va favorita e non ostacolata da chi la matematica, oltre che praticarla, la insegna. Imparare a intuire i concetti, le regole; imparare a congetturare, cioè a fare ipotesi su quel che sta per succedere nel proseguo dell'azione matematica; imparare a predire quel che avverrà, sono acquisizioni necessarie sia nella ricerca che in aula, se si vuole che l'apprendimento sia efficace e significativo.''

Bruno D'Amore, ''La matematica non serve a nulla''

Ad Alessandra

Indice

Prefazione

Il presente libro nasce con lo scopo di raccogliere le esperienze realizzate dal 2014 con il Laboratorio di Fisica *Open* ed è rivolto a chiunque voglia replicare l'esperienza e valorizzare.

I prerequisiti sono una buona conoscenza del sistema operativo Ubuntu, basi di programmazione e conoscenza delle leggi fisiche.

Il Laboratorio di Fisica digitale è interamente *Open Source* ed è stato realizzato, negli anni, dagli studenti del Liceo 'Renato Donatelli' di Terni.

Si ringrazia la Dirigente Scolastica Professoressa Luciana Leonelli per aver permesso e sostenuto la realizzazione di questo progetto, nato anche grazie ai fondi del Piano Nazionale di Scuola Digitale.

Il progetto del “Laboratorio di Fisica *Open*” (Amb.Uino, Ambiente Arduino) nasce nel 2014. Gli insegnanti del Liceo, dovendo proporre un’attività da svolgere in collaborazione con l’Università di Perugia ed un eventuale associazione culturale del territorio per una terza del Liceo Scientifico Scienze applicate, decisero di realizzare il primo sensore di moto, per gli esperimenti di cinematica.

Il Fisico Marco Ricci, allora ricercatore per l’Università a Terni, suggerì l’uso del software libero Python e della sua libreria grafica, *MatPlotLib*

HackLab, il primo e originale FabLab di Terni, collaborò con indicazioni sull’opportuno sensore da utilizzare.

Così fu creato il primo Laboratorio di Fisica digitale, completamente libero, cioè senza l’uso di materiale protetto da copyright.

Nel frattempo gli studenti del Liceo, anche i più irrequieti ed i più scettici, grazie ad HackLab, imparavano a saldare componenti elettrici,conoscevano python e prendevano confidenza con gli ultrasuoni e ne vedevano le applicazioni nello studio delle leggi fisiche.

I ragazzi nell'anno scolastico 2014-15 hanno realizzato il sensore di moto per gli esperimenti di cinematica, nel 2015-16 il sensore di temperatura per gli esperimenti di termologia, nel 2017-18 il sensore di forza per gli esperimenti di dinamica, nel 2018-19 il sensore per studiare i campi magnetici, nel 2019-20 hanno ripreso il sensore di forza, l'hanno costruito in maniera nuova, alla luce di un nuovo componente elettronico in commercio ed hanno realizzato un esperimento sulla legge di Hooke della molla, nel 2020-21 hanno realizzato in Didattica a Distanza l'esperimento con i led sulla determinazione della costante di Planck.

Il Laboratorio di Fisica *Open*, è, come si diceva, prodotto con Software ed Hardware liberi (sin dal Sistema Operativo usato: Linux Ubuntu) ma ha anche un altro punto di forza dal punto di vista didattico: produce grafici.

Capire una legge fisica è stato sempre un problema per le persone meno intuitive e, con questo sistema, la legge non viene solo enunciata ma anche disegnata; noi vediamo gli assi cartesiani, vediamo rette curve e capiamo che formula c’è dietro ed apprendere diventa decisamente più semplice con le immagini.

Ad esempio nel moto rettilineo uniforme, durante l’esperimento, vediamo il carrello che si sposta 'uniformemente' lungo una retta e contemporaneamente, vediamo formarsi sullo schermo del computer una retta: capire questa volta che sull’asse delle x c’è il tempo e su quello delle y lo spazio, cioè la distanza misurata dal sensore, è facile.

Capire cosa accadrà alla retta se il carrello va più veloce non è un problema.

Questo esperimento è stato portato nelle scuole con ragazzi di 13-14 anni ed gli alunni delle medie non hanno avuto problemi a prevedere le leggi del

moto rettilineo uniforme nel caso di variazione di uno dei parametri (per esempio la maggiore velocità, o il moto di ritorno verso il sensore).

Certo si è trattata di una intuizione e non di un apprendimento formale, ma di certo si è trattato di un apprendimento significativo.

Gli studenti che hanno partecipato al progetto affermano: 'I laboratori realizzati a scuola ci hanno permesso di fare nuove amicizie, di confrontarci, di cogliere l'occasione per conoscere l'ambiente Arduino, poi chi è più motivato, proseguirà il percorso secondo le proprie aspettative.'

I laboratori hanno premesso anche l'integrazione di studenti con problemi relazionali anche gravi, che, trovandosi ad affrontare tematiche completamente nuove, legate alla tecnologia, in un contesto non usuale, si sono entusiasmati, partecipando sempre con una rinnovata correttezza relazionale.

Punti di debolezza del progetto: la necessità di aggiornare conntinuamente il Laboratorio alle nuove versioni dei vari Software utilizzati.

Ultimo punto di forza del progetto: i dati ricavati e le leggi ricavate assumono una buona, se non ottima precisione.

Per visionare tutti i lavori fatti con il Laboratorio di Fisica Open cliccare al Link

Sistema Operativo e Sensore di Moto

Installare il Sistema operativo

Installare Ubuntu seguendo le istruzioni al link.

Installare MatPlotLib e Arduino IDE

Cliccare su attività e digitare Terminale e da terminale inserire i seguenti comandi

Installare PyGame

sudo apt-get install python-pygame

Installare Matplotlib

sudo apt-get build-dep matplotlib

sudo apt-get install python3-pip

sudo pip3 install matplotlib

Se ci dovessero essere dei problemi con i repository, non riesce a caricare i file:

sudo nano /etc/apt/sources.list

permetterà di editare il file delle risorse, cancellare tutti gli # che precedono gli indirizzi con *src-deb* in modo tale da rendere i repository utilizzabili.

A questo punto:

sudo apt-get update

e ripetere i comandi dell'installazione di matplotlib

Installare python serial

Effettuare il download del pacchetto Pyserial-3.4.tar.gz

estrarre il pacchetto su desktop

aprire la cartella di pyserial e aprire il terminale nella cartella con il tasto destro del mouse e poi:

sudo python3 setup.py install

Installare Arduino Ide

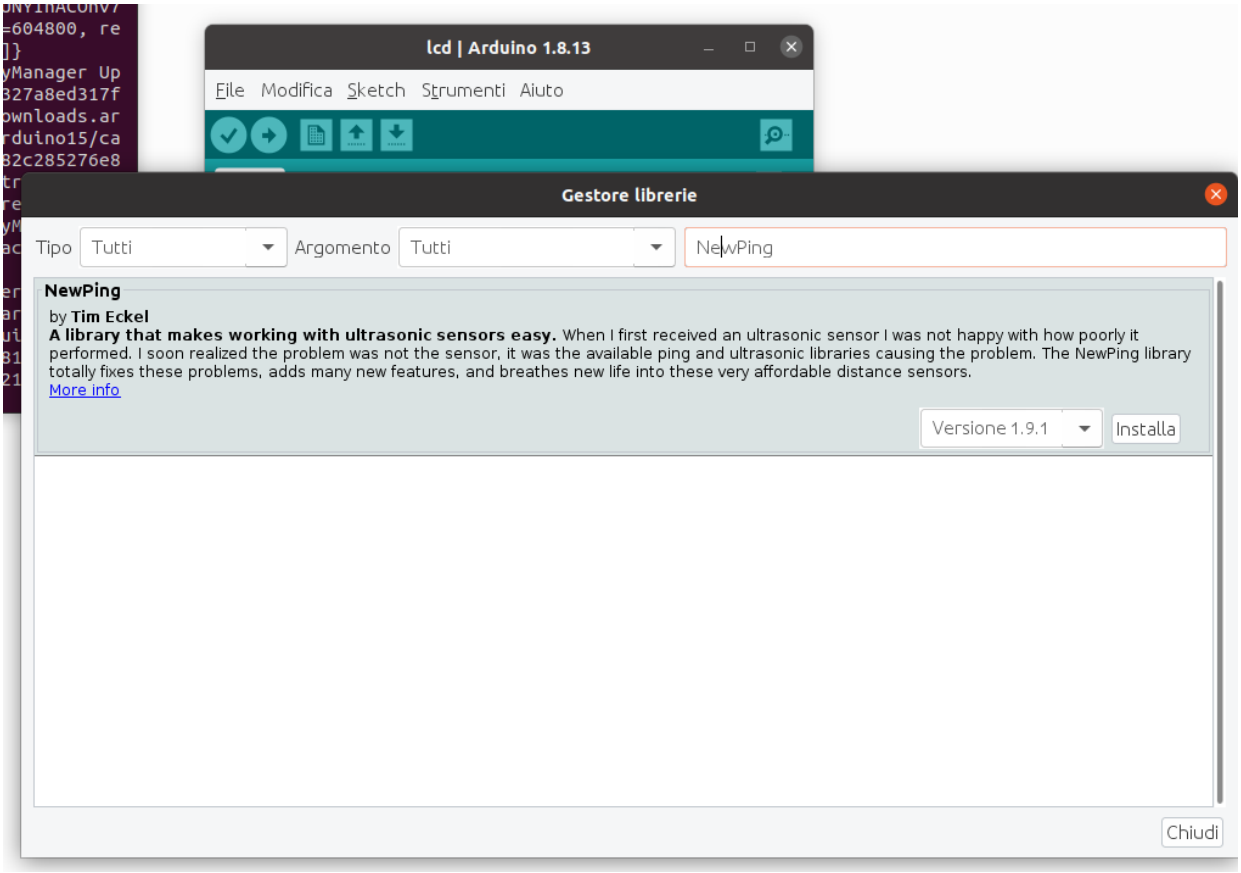
Scaricarlo da https://www.arduino.cc/en/main/software

estrarlo nella home, aprire il terminale nella cartella creata e digitare:

sudo ./arduino

Installare il sensore di moto

Per Arduino IDE è stata caricata la libreria NewPing.h e per farlo basta, una volta avviato Arduino IDE, cliccare su strumenti,gestione libreria poi, su filtra, digitare NewPing e cliccare infine su installa (vedi figura).

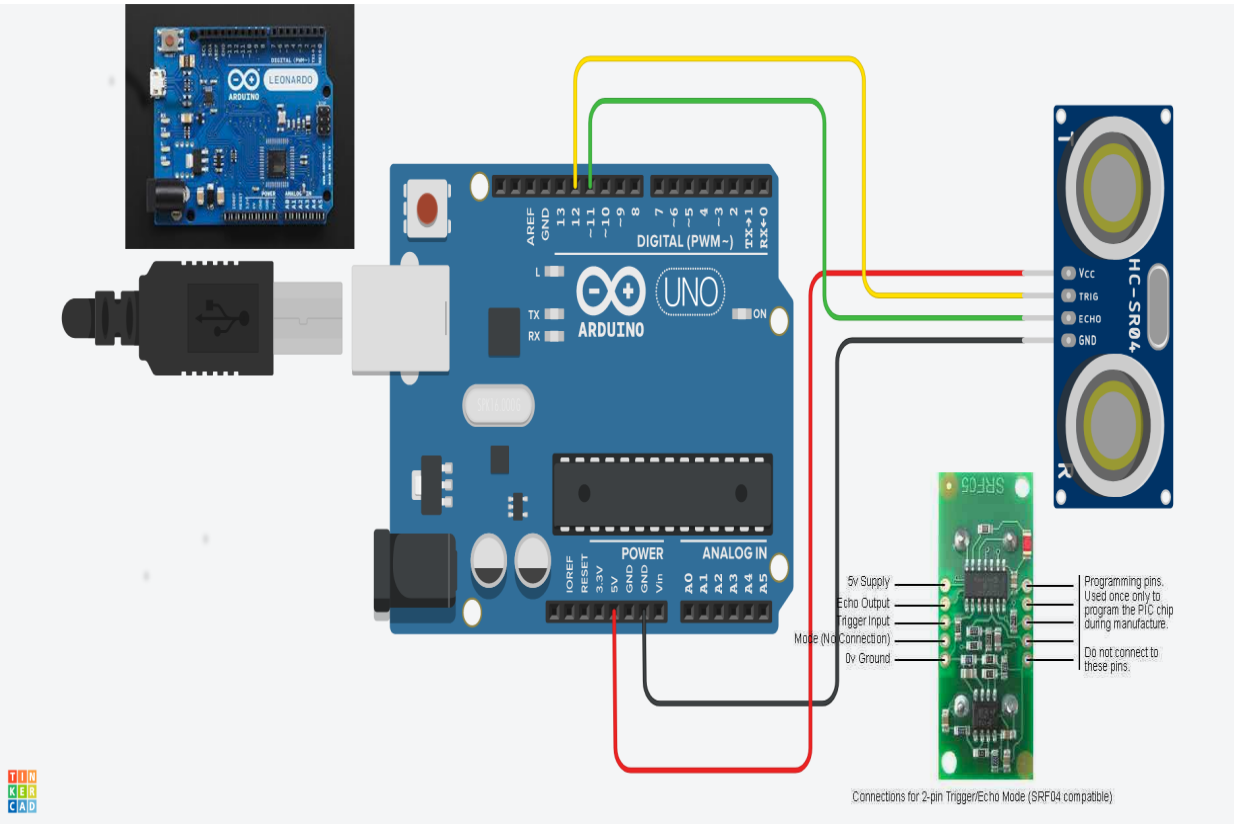


Caricare su Leonardo il seguente sketch:

```
#include <NewPing.h>

#define TRIGGER_PIN 12

#define ECHO_PIN 11

#define MAX_DISTANCE 200

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);


void setup() {

  delay(3000);
```

```
  Serial.begin(9600);
}
  void loop() {
  delay(50);
  unsigned int uS = sonar.ping();
  float s = uS*0.034385/2;
  Serial.println(s);
}
```

Il sensore di moto da utilizzare è **SRF05,** ed è da collegare secondo lo schema indicato. Ricordo che per assemblare i sensore va utilizzato Arduino-Leonardo e il sensore SRF05 e non SRF04.



https://www.robot-italy.com/it/low-cost-ultrasonic-range-finder-1.html

Preparare il file per MatPlotLib

A questo punto il file di Python da utilizzare è il seguente:

```
import matplotlib

matplotlib.use("Qt5Agg")

import numpy as np

import matplotlib.pyplot as plt
```

```
import matplotlib.animation as animation
from time import sleep
import serial
import time
fine="c"
pre="ciao"
while (fine != "f"):
    input("\n\n\n Premi Enter per far partire la registrazione dei dati")
    xdata, ydata = [], []
    ser = serial.Serial('/dev/ttyACM0', 9600)
    while (False):
       pre=ser.readline()
    pre=ser.readline()
    prec = float(pre)
    y=prec
    fig, ax = plt.subplots()
    line, = ax.plot([], [], lw=2)
    ax.set_ylim(0, 40)
    ax.set_xlim(0, 20)
    ax.grid()
```

```
def data_gen():
    t=0
    y=0
    cnt=0
    p=0
    a = ser.readline()
    while (True)&(t < 20):
        a = ser.readline()
    y = float(a)
    if (y != 0) :
        t = (time.time() - start)
        t = float(t)
        yield t, y
def run(data):
     t,y = data
     xdata.append(t)
     ydata.append(y)
     ax.figure.canvas.draw()
     line.set_data(xdata, ydata)
     return line,
```

```
data_gen.t = 0
while (abs(y-prec) < 2):
    if True:
        a = ser.readline()
        y = float(a)
start = time.time()
ani=animation.FuncAnimation(fig,run,data_gen,blit=True,interval=5,repeat=True)
plt.show()
stop =0
fine=input("Premi f per finire :")
```

Il file può essere scritto con Gedit, l’editor di testo di Ubuntu, e salvato come *grafmar1.py* .

Collegare Leonardo, con il sensore SRF05, al computer aprire il terminale nella cartella che contiene il file *grafmar1.py* e digitare

sudo python3 grafmar1.py

e la finestra grafica di MatPlotLib per la registrazione dei dati si avvierà non appena l’oggetto da studiare per il moto avrà compiuto un movimento.

Dopo l'esperimento è possibile salvare il grafico con il menù della finestra MatPlotLib.

Elaborazione del grafico con GeoGebra

Al link di seguito è possibile seguire un tutorial su come si elaborano i grafici ottenuti dagli esperimenti utilizzando GeoGebra

https://www.youtube.com/watch?v=xkkQtsOgCPM

è sufficiente seguire i primi 4 minuti di spiegazione.

Esempi di esperimenti

Si è pensato di mostrarvi i seguenti due esperimenti:

Moto rettilineo uniforme

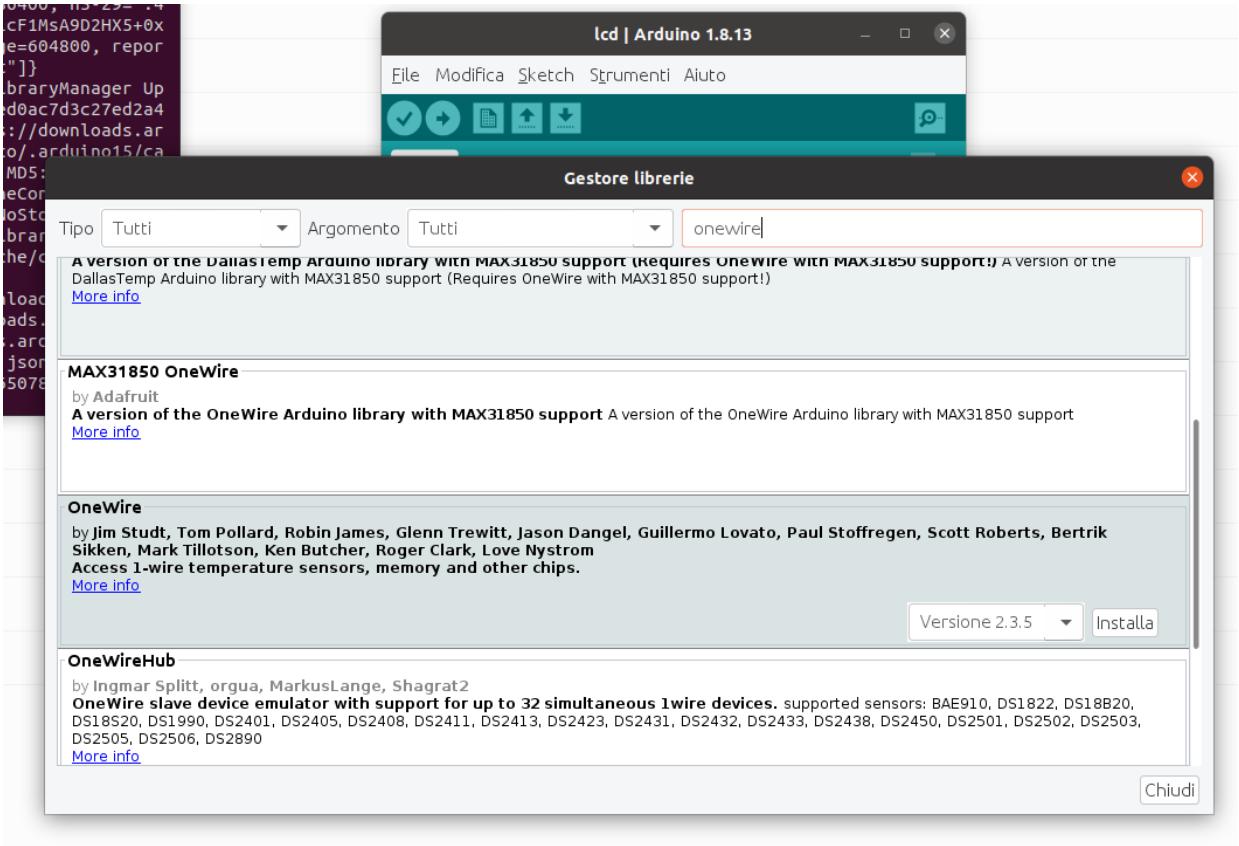
Moto Armonico

Sensore di Temperatura

Preparazione dello strumento di misura

Acquistare un sensore di temperatura DS18B20, una resistenza da 4,7 kΩ ed una breadboard.

Per Arduino IDE caricare la libreria *OneWire.h* e per farlo basta, una volta avviato Arduino IDE, cliccare su strumenti, gestione libreria poi, su filtra, digitare OneWire e cliccare infine su installa (vedi figura).

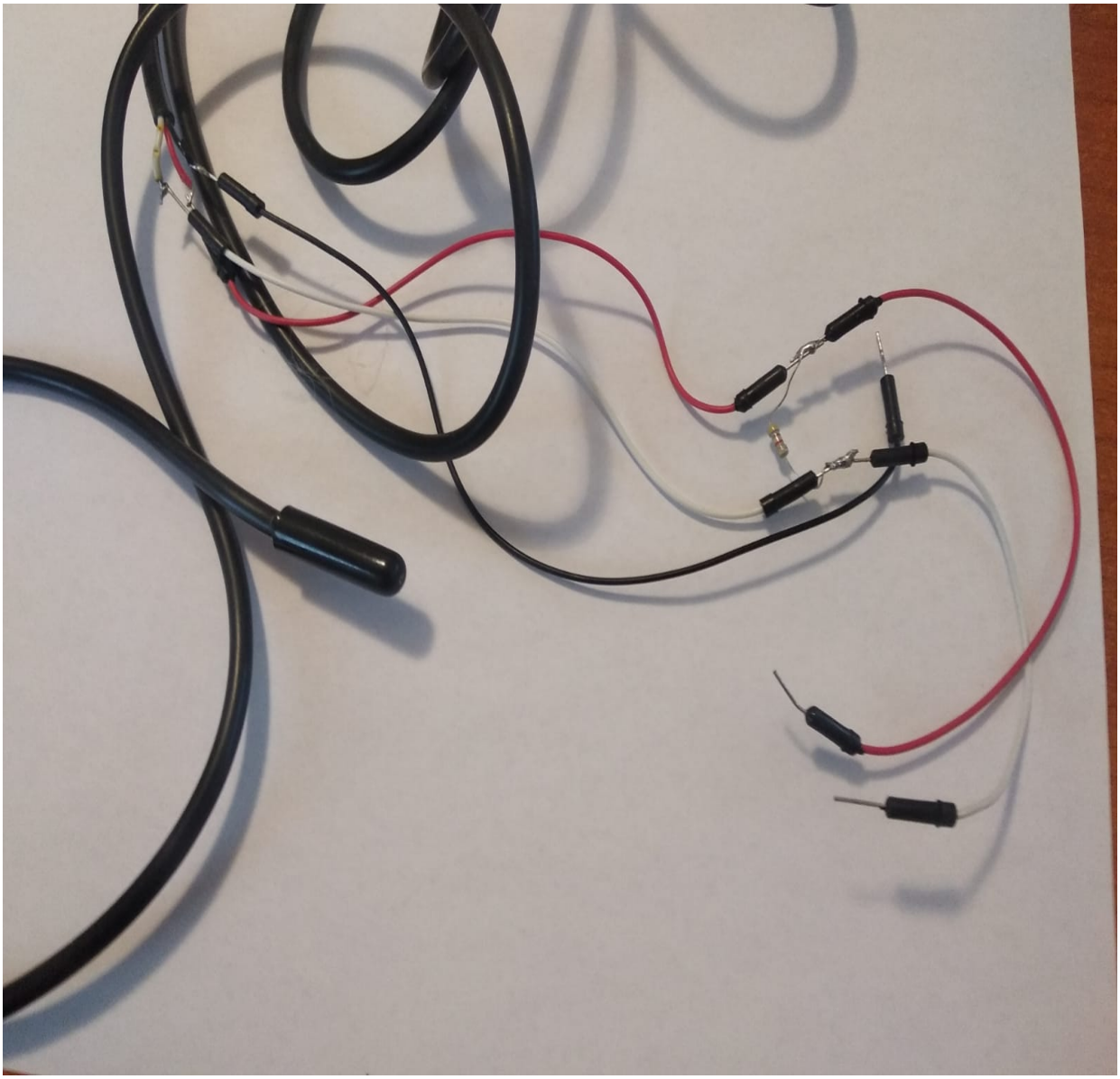


Modificare uno sketch di esempio di applicazione della libreria in modo tale che possa essere utile agli esperimenti di Fisica.

Per Python modificare il programma di rilevazione dati e rappresentazione grafica in maniera tale che raccogliesse dati per 10 minuti.

Per scaricare sketch e programma Python clicca qui.

Saldare i fili con il resistore (vedi figura), il filo rosso deve essere collegato a 5V, il filo nero al GRND e il filo bianco al PIN 10.



Esempi di Esperimenti

Passaggio di stato filmato

Passaggio di stato relazione

Effetto Joule (Esperimento ispirato dagli appunti lasciati dal Professor Francesco Celi al Liceo)

Sensore di forza

Preparazione dello strumento di misura

Prima di tutto si acquista una cella di carico con fondo scala di 1 kg e relativa scheda di amplificazione ; si saldano i componenti e si collega la scheda ad Arduino secondo lo schema indicato in figura.

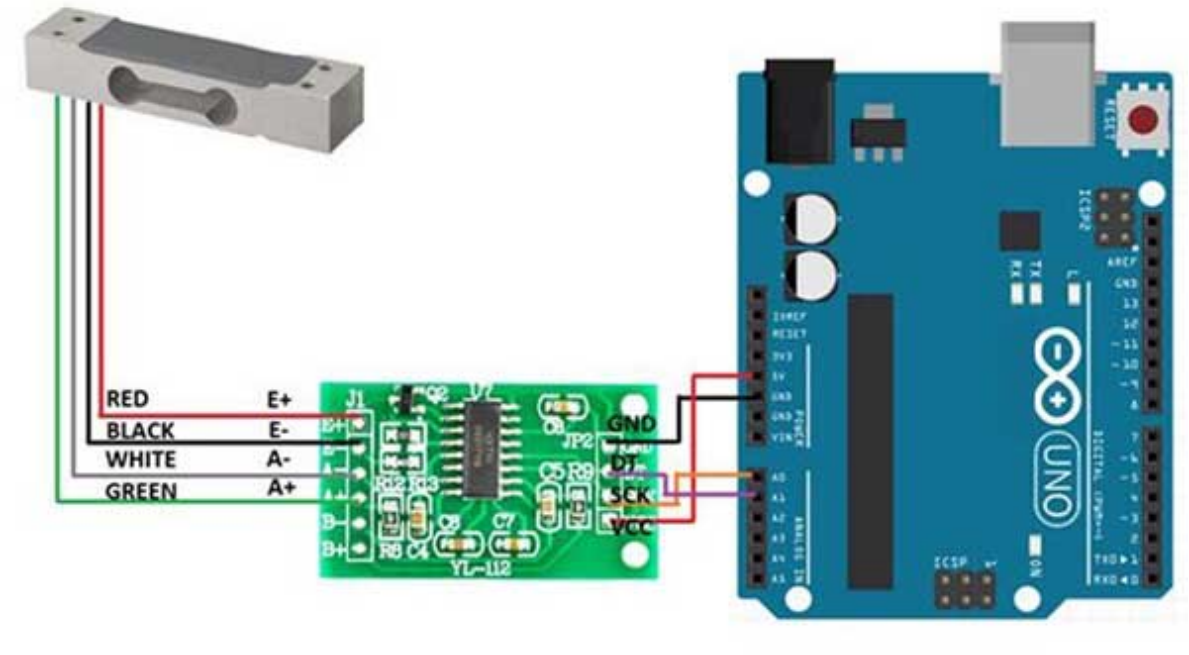


Figura da *Futura Elettronica*

Scaricare da sito di Futura Elettronica il file sotto Esempi *Hx711.zip* (si tratta della libreria da caricare per far funzionare il dinamometro).

Prima di caricare la libreria bisogna apportare alcune modifiche, quindi si decompatta il file e si modifica il file *hx711.cpp* alla riga: *delayMicroseconds(100)* lo si modifica con *delayMicroseconds(10)*

E nel file *hx711.h*

long averageValue(byte times = 32) lo si modifica con *long averageValue(byte times = 1).*

La modifica della libreria nasce dalla necessità di ricevere i dati con maggiore frequenza utilizzando la stessa procedura che è scritta nello sketch di esempio il *SerialScale.ino,* allegato nella libreria.

Anche se non si sa esattamente cosa è stato fatto, va bene lo stesso perché nella logica dell'informatica: 'se funziona va bene così com'è'.

Una volta fatte le modifiche ai due file si ricostruisce il file *.zip*, si avvia Arduino (ricordo a chi usa Ubuntu di avviare Arduino da Super User altrimenti non vede la porta della scheda), si carica la libreria *Hx711.zip*, si apre lo sketch *SerialScale.ino,* che troverete tra gli esempi e si carica sulla scheda.

Taratura dello strumento: appendere alla cella di carico due pesi di massa nota (y_1,y_2), rilevare le misure ottenute (x_1,x_2) e poi mediante l'equazione $(y-y_1)/(y_2-y_1)=(x-x_1)/(x_2-x_1)$ si ottiene la legge lineare $y=m\cdot x + q$che permette di trasformare il dato rilevato x nel peso y in Newton e che verrà utilizzato nel file di Python.

Lo sketch avrà l'aspetto in figura

E’ venuto quindi il momento di scrivere il file in Python. Il file si conosce già con piccole modifiche.

```
import matplotlib
matplotlib.use("Qt5Agg")
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from time import sleep
import serial
import time
fine="c"
pre="ciao"
while (fine != "f"):
 raw_input("\n\n\n Premi Enter per far partire la registrazione dei dati")
 xdata, ydata = [], []
 ser = serial.Serial('/dev/ttyACM0', 9600)
 fig, ax = plt.subplots()
 line, = ax.plot([], [], lw=2)
 ax.set_ylim(0, 2)
 ax.set_xlim(0, 20)
 ax.grid()
 def data_gen():
     t=0
     y=0
     cnt=0
     p=0
     while (True)&(cnt < 110):
          a = ser.readline()
          x = float(a)
          if (x != 0):
              cnt =cnt+1
              b = ser.readline()
              y=float(b)
              y= 0.008175 *y-2.002875 #  I numeri inseriti sono i due numeri ottenuti nella taratura dello strumento
              if (cnt>2):
                yield x, y
 def run(data):
     x,y = data
     xdata.append(x)
     ydata.append(y)
     ax.figure.canvas.draw()
     line.set_data(xdata, ydata)
     return line,
 data_gen.t = 0
 start = time.time()
 ani = animation.FuncAnimation(fig, run, data_gen, blit=True, interval=5,repeat=False)
 plt.show()
 stop =0
 fine=raw_input("Premi f per finire :")
```

Esempi di esperimenti

[La dinamica della molla filmato]

[La dinamica della molla relazione]

Sensore di campo magnetico

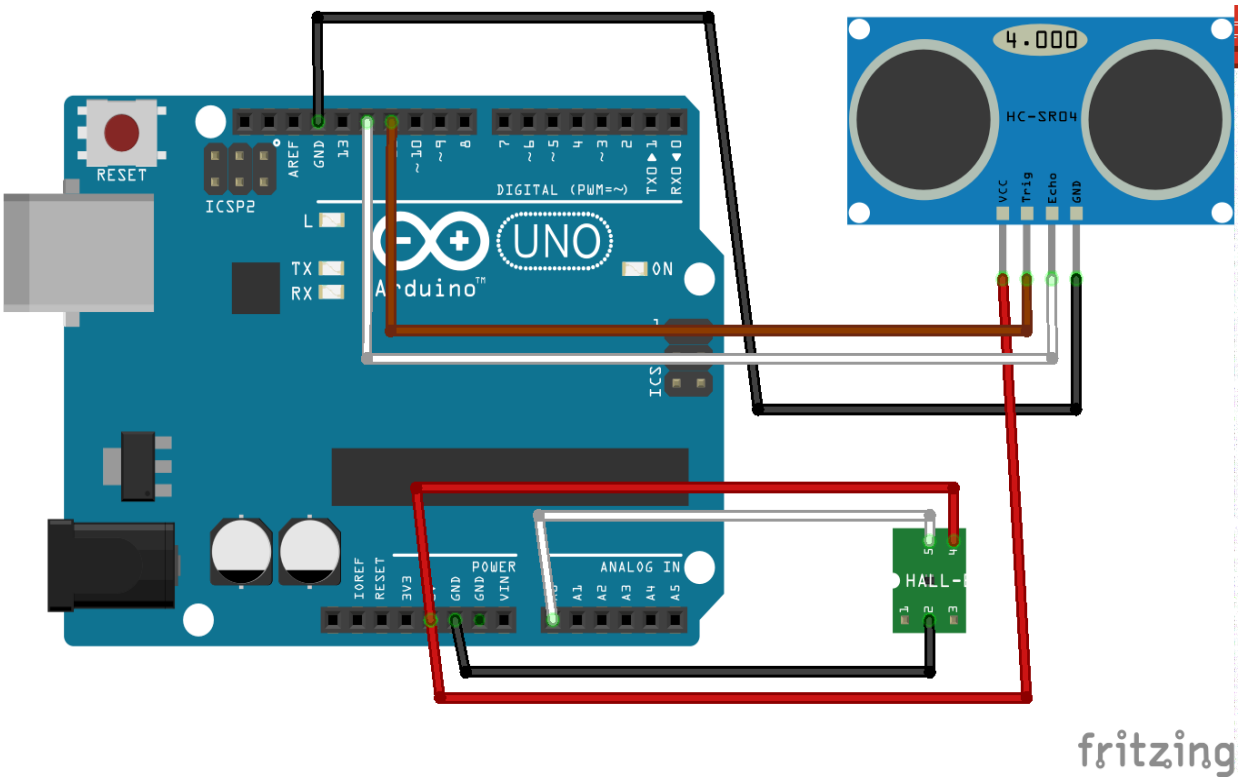
Preparazione dello strumento di misura

Comprare un sensore di Hall per i campi magnetici, per esempio al link https://www.futurashop.it/sensore%20di%20hall

?search=%20sensore%20hall%20 (si possono acquistare anche sensori più sensibili).

Comprare un sensore di movimento SRF05, per esempio al link https://www.robot-italy.com/it/low-cost-ultrasonic-range-finder-1.html e procedere nel caricamento della libreria come in sensore di moto.

Nella figura seguente potete vedere come collegare i jumper (cavetti) per far funzionare il tutto. Il sensore di Hall e quello di moto non sono la riproduzione fedele di quelli acquistati, ma le connessioni sì, ed anche i colori dei cavi sono gli stessi.



Muovere il magnete lungo una rotaia rettilinea verso il sensore di Hall e contemporaneamente registrare la distanza del polo Nord del magnete dal sensore di Hall e il valore del campo magnetico a quella distanza. Il tutto viene rappresentato graficamente con il nostro **Laboratorio di Fisica *Open***, assegnando all'asse *x* la distanza in cm ed all'asse *y* il campo magnetico in Gauss.

Stampare con una stampante 3D la rotaia e il supporto per il sensore di Hall, avendone fatto il progetto con Tinkercad.

Programma per l'IDE

#include <NewPing.h> // *importa la libreria NewPing*

#define TRIGGER_PIN 12 // *PIN per i dati del sensore di moto*

#define ECHO_PIN 11 // *PIN per i dati del sensore di moto*

#define MAX_DISTANCE 200 //*distanza massima tollerata*

```
int sensorPin = A0; // PIN per il sensore di Hall
int sensorValue = 0; // variabile per la lettura di A0
int gss =0; // variabile per la conversione in Gauss
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
void setup() {
  delay(3000); //tempo di attesa prima di far partire la registrazione dei dati
  Serial.begin(9600);
}
void loop() {
  delay(100); // attesa di 1/10 di secondo
  unsigned int uS = sonar.ping(); // misura del tempo dell'eco
   float s = uS*0.034385/2; // misura della distanza in cm
   Serial.println(s); // output della distanza
   sensorValue = analogRead(sensorPin); // lettura del valore di Hall
   gss = map(sensorValue, 205, 818, -1000, +1000); // conversione in Gauss (*)
   Serial.println(gss); //output del campo magnetico (**)
}
```

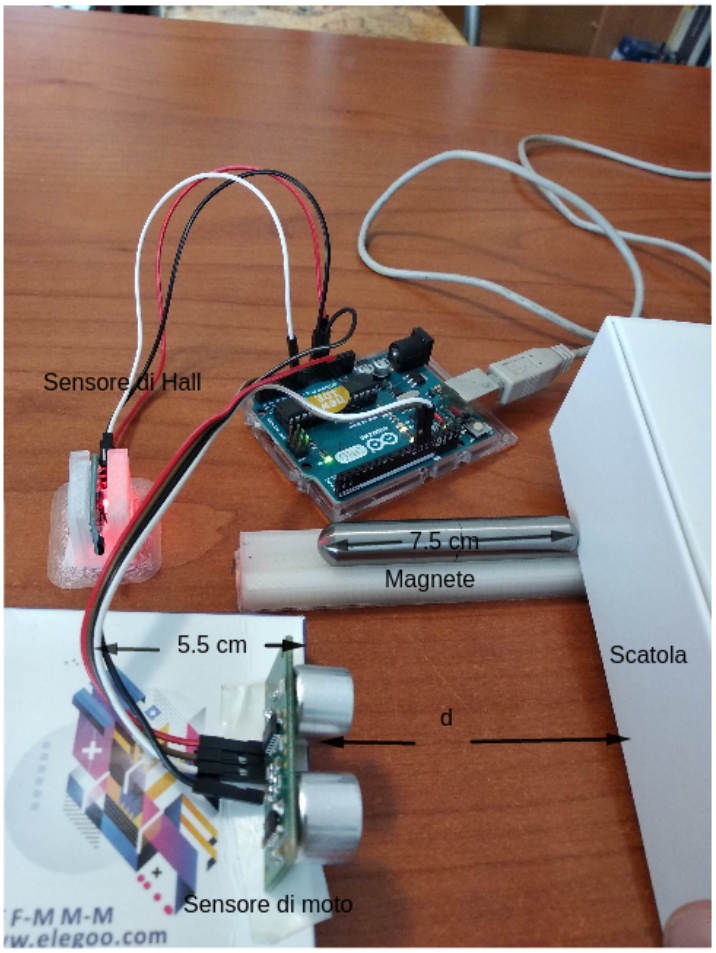
(*) La funzione *map* converte i dati del primo intervallo [205,818] in quelli nel secondo intervallo [-1000,1000] in una corrispondenza lineare. I primi sono i dati in input analogico (che sapete variano da 0 a 1023 in corrispondenza alle tensioni da 0 a 5 V).

Si legge nel *datasheet* (una sorta di manuale) del sensore di Hall che, a 1 V di tensione del sensore in oputput corrispondono -1000 Gauss del campo magnetico e che a 4 V corrispondono 1000 Guass e dato che a 1V corrisponde anche il valore 205 in input ed a 4V corrisponde 818, il passaggio è immediato.

Per calibrare il sensore del campo magnetico si leggono i dati in output senza magneti vicini. Il valore costante evidenziato nel nostro caso era 57 Gauss, quindi si è modificatolo sketch in **(**)** ed ho scritto **Serial.println(gss-56),** poiché il campo magnetico terrestre è sicuramente minore di 1 Gauss ed il campo magnetico misurato dal sensore, in assenza di magneti vicini, non può dare valori maggiori di 1.

Una volta fissati sul tavolo con nastro biadesivo, rotaia e supporti, ho misurato con il righello la lunghezza del magnete (7.5 cm) e la distanza approssimativa del sensore di Hall dal sensore di moto prendendo a riferimento la metà dei sensori (5.5 cm).

Così se la distanza tra il sensore di moto e la scatola che muove il magnete (vedere figura 5) è *d*, la misura della distanza del magnete dal sensore sarà $x = (d\text{-}7.5) + 5.5\ cm$ cioè $x = d - 2$.

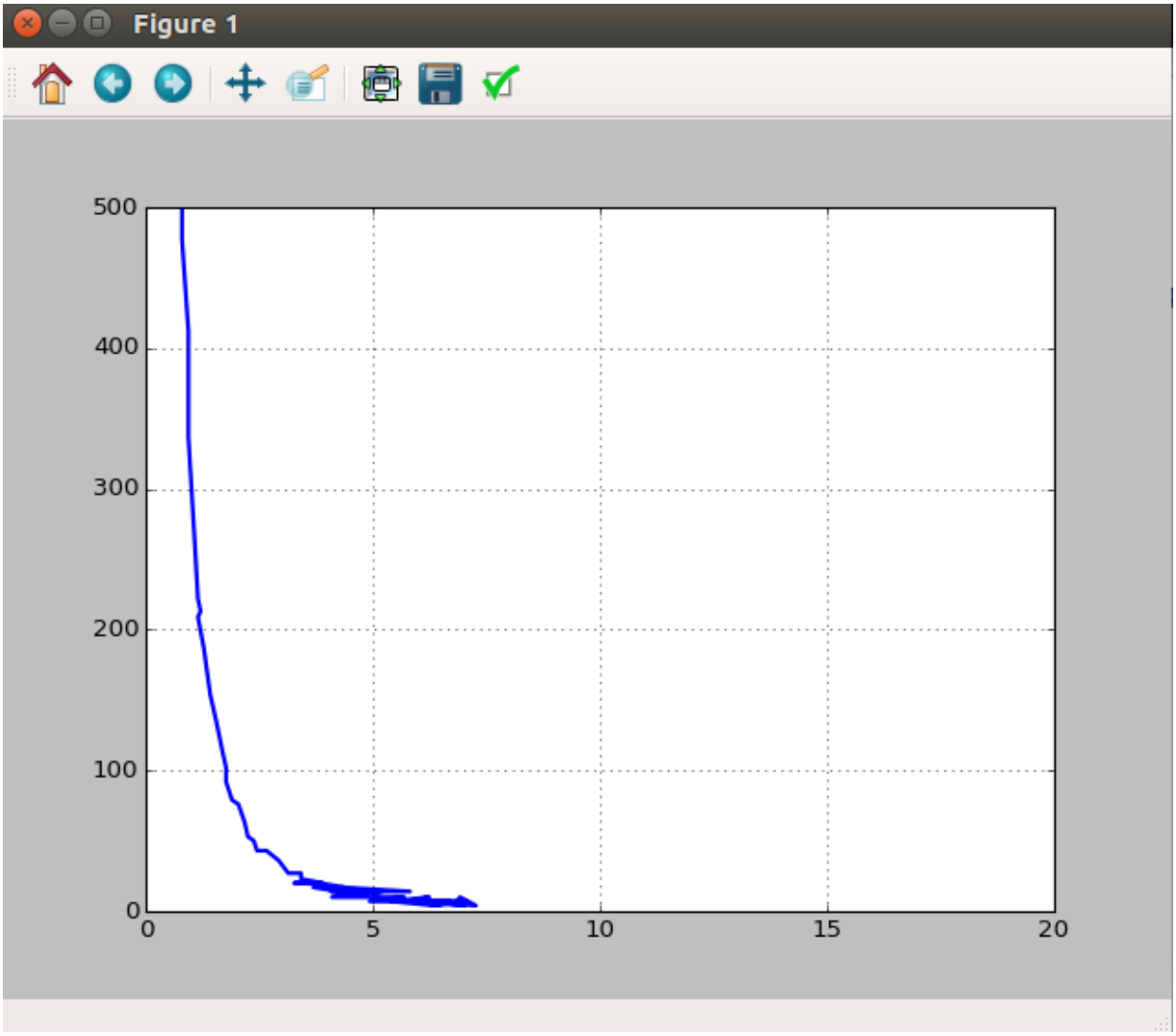


Programma in Python:

```
import matplotlib
matplotlib.use("Qt5Agg")
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from time import sleep
import serial
import time
fine="c"
pre="ciao"
while (fine != "f"):
        raw_input("\n\n\n Premi Enter per far partire la registrazione dei dati") # metto in pausa
        xdata, ydata = [], []
        ser = serial.Serial('/dev/ttyACM0', 9600)
        while (False):
                pre=ser.readline()
        pre=ser.readline()
        prec = float(pre)
        y=prec
        fig, ax = plt.subplots()
        line, = ax.plot([], [], lw=2)
        ax.set_ylim(0, 500) # assegno intervallo di rappresentazione per le y
        ax.set_xlim(0, 20)   # assegno intervallo di rappresentazione per le x
        ax.grid()
        def data_gen():
                y=0
                cnt=0    # assegno un contatore, prendo 100 misure
                a = ser.readline()
                while (True)&(cnt < 100):
                        a = ser.readline()
                        x = float(a)-2  # leggo della distanza e tolgo 2 cm
                        prec=a
                        if (x != 0):
                                cnt =cnt+1  # dopo aver inserito la distanza implemento il contatore
                                b = ser.readline() # leggo il campo magnetico
                                y = float(b)
                                yield x, y
        def run(data):
                x,y = data
                xdata.append(x)
                ydata.append(y)
                ax.figure.canvas.draw()
                line.set_data(xdata, ydata)
                return line,
        data_gen.x = 0
        ani = animation.FuncAnimation(fig, run, data_gen, blit=True, interval=5,repeat=False)
        plt.show()
        fine=raw_input("Premi f per finire :")
```

Al dato della distanza in Input verrà tolto 2 cm: ciò permetterà di ottenere un’idea, anche se un po’ approssimativa, della distanza del polo Nord del magnete dal sensore di Hall.

Di seguito il grafico ottenuto nel nostro primo esperimento.



Esempi di esperimenti

[Campo magnetico filmato]

[I Relazione degli studenti del Linguistico]

[II Relazione degli studenti del Linguistico]

Esperimento sulla costante di Planck

Preparazione dell'apparato

Per prima cosa, per leggere la frequenza della luce dei led, si può costruire uno spettroscopio acquistandolo per pochi euro da https://www.opitec.it/index.php?lang=4&cl=search&searchparam=spettroscopio.

Si utilizza un potenziometro standard, un resistore, R di circa 20 Ω, messo in serie con il diodo_led.Con i Pin A1 e A2 si misura la differenza di potenziale agli estremi della resistenza R_1, dalla quale si può dedurre l'intensità di corrente passante attraverso il circuito, usando la Prima; Legge di Ohm: $\Delta V = R \cdot I$.

Con i Pin A0 e A1 si misura la differenza di potenziale ai poli del diodo_led (vedi figura)

Si è deciso di utilizzare la libreria *Pyfirmata* per collegare direttamente Python con Arduino.

Seguendo le indicazioni al link https://realpython.com/arduino-python/#reading-analog-inputssi installa la libreria per Python con il seguente comando da terminale:

sudo pip install pyfirmata

poi, dopo aver preparato Arduino, si collega al PC e dal menù dell'IDE di Arduino si carica lo sketch:

Esempi>Firmata>StandardFirmata.ino

Lo sketch permette di dialogare direttamente da Python con Arduino e leggere i pin analogici.

Il programma per Python è il seguente:

```
import pyfirmata # Inserimento delle varie librerie

import matplotlib

matplotlib.use("Qt5Agg")

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.animation as animation

from time import sleep

import serial

import time

board = pyfirmata.Arduino('/dev/ttyACM0')

it = pyfirmata.util.Iterator(board)

it.start()

pin0=board.get_pin('a:0:i')

pin1=board.get_pin('a:1:i')

pin2=board.get_pin('a:2:i')

fine="c" # dichiarazione delle variabili
```

```
pre="0.0"

a="ciao"

while (fine != "f"): #ciclo principale

    input("\n\n\n Premi Enter per far partire la registrazione dei dati") #inserimento controllo

    xdata, ydata = [], []

    fig, ax = plt.subplots() # configurazione della finestra grafica

    line, = ax.plot([], [], lw=2) # configurazione della curva da tracciare

    ax.set_ylim(0, 70) #set degll'asse y

    ax.set_xlim(0, 4) # set dell'asse x

    ax.grid()

    def data_gen(): #funzione di generazione dei dati

        t=0

        while (True): #continua a leggere i dati secondo lo schema impostato per 20 s

            sens0=pin0.read() #il dato letto è un valore tra 0.0 e 1.0 che corrisponde a 0.0 V e 5.0 V

            sens1=pin1.read()

            sens2=pin2.read()

            time.sleep(0.1)

            dd1=abs(sens1-sens2)*5.0 # ddp R converto il dato in Volt
```

```
        dd2=abs(sens0-sens1)*5.0 # ddp Led
         i=dd1/0.02 #corrente nel Led in mA
        x=dd2
        y=i
        yield x, y
        t = (time.time() - start) #legge il tempo in s
        t = float(t)
    def run(data): #funzione di plot dei dati
        x,y = data
        xdata.append(x)
        ydata.append(y)
        ax.figure.canvas.draw()
        line.set_data(xdata, ydata)
        return line,
    data_gen.x = 0
    start = time.time() #tempo di inizio di registrazione dei dati
    ani = animation.FuncAnimation(fig, run, data_gen, blit=True,
interval=5,repeat=False)
    plt.show()
    fine=input("Premi f per finire :")
```

Le istruzioni valgono per il SO Ubuntu 20.04, aprire il terminale nella cartella del file in python e digitare il comando:

sudo python3 grafmar_Planck.py

Esempi dell'esperimento

Presentazione

Prima relazione

Seconda relazione

Ringraziamenti

Al **Professor** Marco Ricci, ora docente Universitario, per aver suggerito Python come linguaggio interprete dei dati ed aver seguito il progetto sul nascere

Ad **HackLab** di Terni, per i suggerimenti sull'Hardware da utilizzare e il supporto nella didattica

Al **TG3 Regione Umbria** che ha dato visibilità al progetto

All'**Istituto Classico e Artistico** di Terni che ha permesso di replicare l'esperienza

All'**Istituto Comprensivo 'G. Marconi'** di Terni che ha permesso di sperimentare il progetto con gli studenti più piccoli

All'**intera comunità scolastica del Liceo 'Donatelli'**:

Alla **Dirigente Scolastica** Professoressa Luciana Leonelli che ha permesso e favorito la realizzazione di questo progetto;

Ai **docenti** che hanno partecipato, supportato e seguito il progetto (non solo i docenti di Matematica e Fisica, ma anche gli insegnanti di Scienze, Inglese.....)

Al **personale di segreteria** che ha seguto la fase burocratica dei progetti e degli acquisti di materiali;

Ai **collaboratori scolastici** che hanno avuto sempre una grande attenzione verso il Laboratorio di Ambiene Arduino (Amb.Uino);

ma soprattutto un grande grazie è rivolto ai **genitori** e agli **alunni** di questa scuola che si sono succeduti negli anni ed hanno promosso ed apprezzato la nostra attività didattica in maniera sempre crescente.