Maparea câmpului magnetic folosind un senzor Hall și un Accelerometru

Cuprins

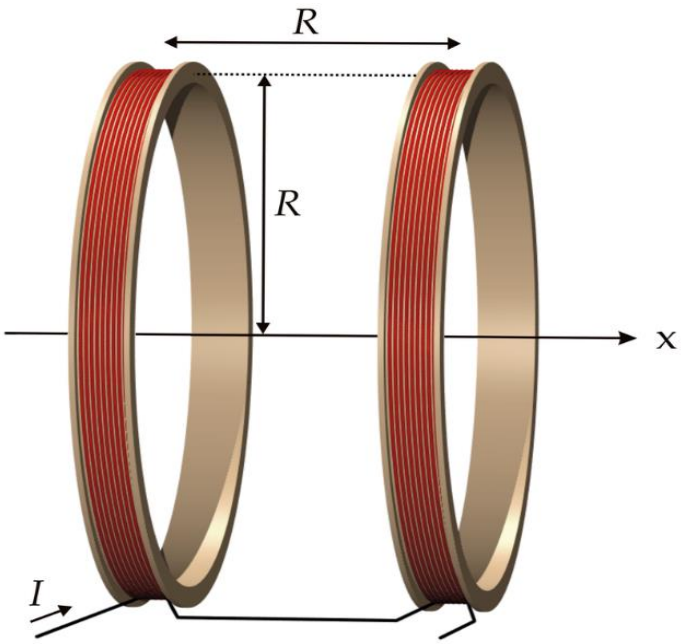
1. **Scopul lucrării :**

Scopul lucrării este crearea unui sistem folosind un senzor Hall și un accelerometru cu care v-om putea verifica (prin maparea acelor date) dacă câmpului magnetic generat de o bobină Helmholtz este constant sau nu.

2. **Teoria lucrării:**

Bobina Helmholtz

O bobină Helmholtz este un dispozitiv pentru producerea unei regiuni cu câmp magnetic aproape uniform, care este alcătuită din doi electromagneți pe aceeași axă, ce transportă un curent electric egal în aceeași direcție.



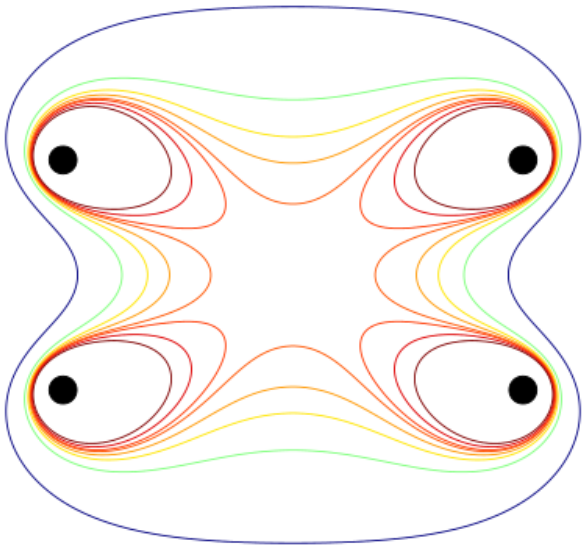
Calculul câmpului magnetic exact în orice punct al spațiului este complex din punct de vedere matematic și implică studiul funcțiilor Bessel. Lucrurile sunt mai simple de-a lungul axei perechii de bobine și este convenabil să ne gândim la extinderea seriei Taylor a intensității câmpului ca o funcție de x distanța de la punctul central al perechii de bobine de-a lungul axului. Prin simetrie, termenii de ordin impar din expansiune sunt zero. Prin aranjarea

bobinelor astfel încât originea $x = 0$ să fie un punct de inflexiune pentru intensitatea câmpului datorată fiecărei bobine separat, se poate garanta că ordinul termenului x^2 este, de asemenea, zero și, prin urmare, termenul principal non-constant este de ordinul x^4. Punctul de inflexiune pentru o bobină simplă este situat de-a lungul axei bobinei la o distanță $R/2$ de centrul acesteia. Astfel, locațiile pentru cele două bobine sunt $x = \pm\frac{R}{2}$. Calculul detaliat mai jos oferă valoarea exactă a câmpului magnetic în punctul central. Dacă raza este **R**, numărul de spire din fiecare bobină este **n** și curentul prin bobine este **I**, atunci intensitatea câmpul magnetic **B** la mijlocul dintre bobine va fi dat de relația :

$$\boldsymbol{B = \left(\frac{4}{5}\right)^{\frac{3}{2}} \left(\frac{\mu_0 nI}{R}\right)},$$

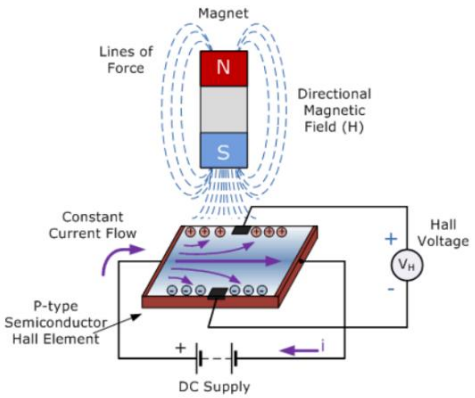
unde $\boldsymbol{\mu_0} = 1.257 * 10^{-6} T * \frac{m}{A}$ (permeabilitatea magnetică)

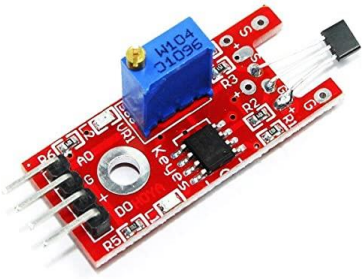
Forma câmpului generat de o bobină Helmholtz



Senzorul Hall

Senzorul Hall este un tip de senzor magnetic care poate fi utilizat pentru detectarea intensității și direcției unui câmp magnetic produs de un magnet permanent sau un electromagnet.





KY-024

Într-un senzor Hall, un curent este aplicat unei benzi subțiri de metal. În prezența unui câmp magnetic perpendicular pe direcția curentului, purtătorii de sarcină sunt deviați de forța Lorentz, producând o diferență de potențial electric (tensiune) între cele două părți ale benzii. Această diferență de tensiune (tensiunea Hall) este proporțională cu puterea câmpului magnetic.

Valoarea minimă a intensității câmpului pe care acest senzor o poate detecta este de **2 mT**. Pentru a obține această valoare avem nevoie de un curent de aproximativ **1.2 A** (considerând și rezistența firelor).

$$\boldsymbol{I = \left(\frac{5}{4}\right)^{\frac{3}{2}} \left(\frac{BR}{\mu_0 n}\right)}$$

R ≈ 0.15 m

n ≈ 300

B = 0.002 T

$\mu_0 = 1.257 * 10^{-6} T * m/A$

$$\boldsymbol{I = \left(\frac{5}{4}\right)^{\frac{3}{2}} \left(\frac{0.002 * 0.15}{1.257 * 10^{-6} * 300}\right)} \approx 1.11\ A$$

Accelerometrul

Un accelerometru este un dispozitiv care măsoară vibrația sau accelerația mișcării unei structuri. Forța cauzată de vibrație sau de o schimbare a mișcării (accelerație) face ca masa să „strângă” materialul piezoelectric care produce o sarcină electrică proporțională cu forța exercitată asupra acestuia. Deoarece

sarcina este proporțională cu forța, iar masa este o constantă, atunci sarcina este, de asemenea, proporțională cu accelerația.

ADXL345

3. Componente necesare

1. O bobina Helmholtz pentru generarea câmpului electromagnetic
2. O sursă de curent de cel puțin 1.2 A pentru alimentarea bobinelor
3. O plăcută Arduino
4. Un accelerometru
5. Un senzor Hall
6. Breadboard
7. Fire pentru conectarea senzorilor la Arduino

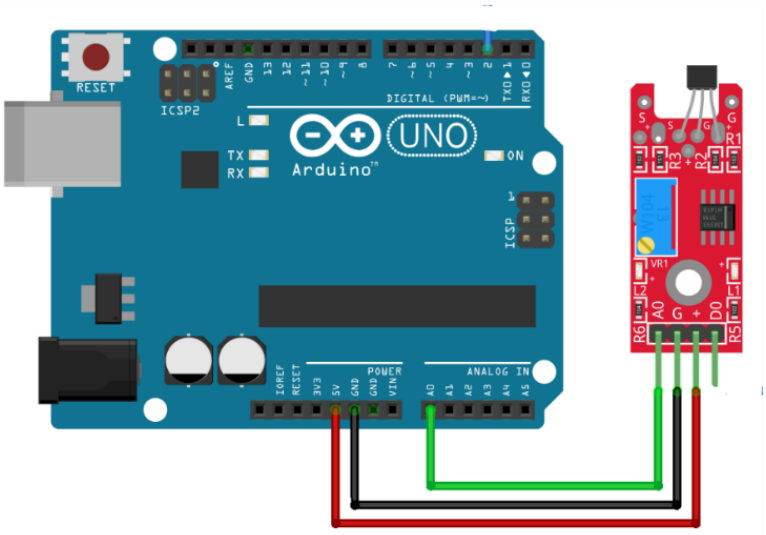
De asemenea sunt utile un ampermetru pentru a măsurarea curentului prin bobine, o șurubelniță pentru a ajusta sensibilitatea senzorului Hall și un magnet pentru a verifica funcționalitatea senzorului.

4. Etape:

1. Realizarea subsistemului de achiziție pentru măsurarea câmpului magnetic.

În această etapa am conectat senzorul Hall la placa Arduino și am scris partea de cod care citește datele de la senzor și le afișează in consolă. De asemenea am verificat dacă datele afișate sunt corecte iar după am calibrat senzorul să afișeze zero atunci când nu acționează nici un câmp asupra lui.

Conexiune:



Partea de cod:

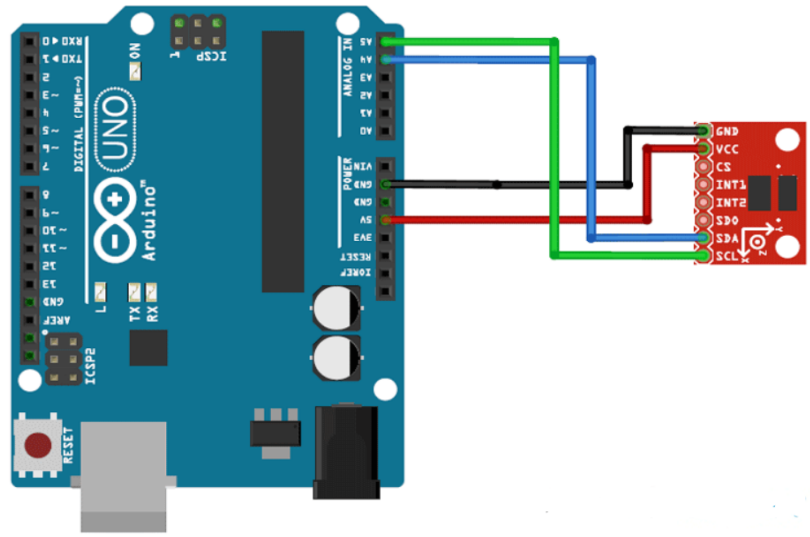
```
int analogPin = A0;
int analogVal;

void setup() {

  Serial.begin(19200);

}

void loop() {

  // Citire date din senzor
  analogVal = analogRead(analogPin);

  Serial.println(" A: ");

  // Calibrare
  analogVal = analogVal - 521;

  // Afisare
  Serial.println(analogVal);

}
```

2. Realizarea subsistemului de identificare a poziției în timp real folosind un accelerometru.

În etapa a doua, am realizat conexiunea dintre accelerometru și Arduino. Pentru acesta lucrare am utilizat senzorul ADXL345, care are toate cele trei axe într-un singur senzor. De asemenea acesta poate fi setat in mai multe intervale de măsură (± 2, 4, 8, 16 g). Noi am utilizat intervalul de 2g.

Pentru partea de cod am utilizat biblioteca deja implementată pentru acest senzor (SparkFun_ADXL345.h) obținând astfel foarte ușor datele de pe cele trei axe.

Conexiune:



Partea de cod:

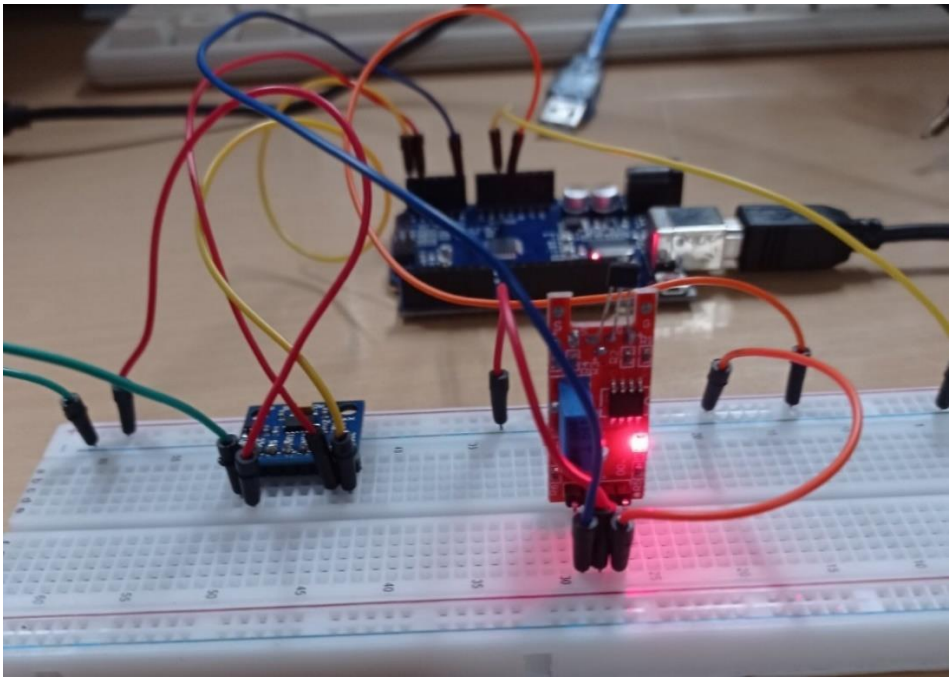
```
#include <SparkFun_ADXL345.h>

ADXL345 adxl = ADXL345();
int range = 2;

void setup() {

    Serial.begin(19200);
```

```
    adxl.powerOn();
    adxl.setRangeSetting(range);

}

void loop() {

    int x, y, z;
    adxl.readAccel(&x, &y, &z);

    // Calibrare  -1000 < x, y, z < 1000

    x = x * 3.9;
    y = y * 3.9;
    z = z * 3.9;

    // Afișare

    Serial.print(" X: ");
    Serial.print(x);

    Serial.print(" Y: ");
    Serial.print(y);

    Serial.print(" Z: ");
    Serial.println(z);

}
```

3. Combinarea celor doua subsisteme si verificarea funcționalității.

În această etapă am combinat cele două bucăți de cod, am conectat ambii senzori la Arduino și am verificat corectitudinea datelor de ieșire.

Conexiune:



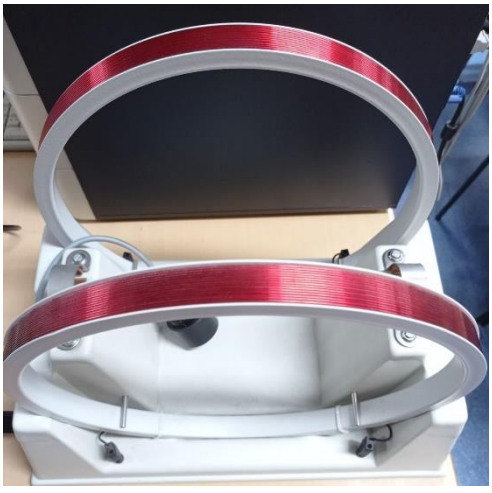
Codul pentru afișarea datelor în consolă:

```
#include <SparkFun_ADXL345.h>
ADXL345 adxl = ADXL345();

int analogPin = A0;
int analogVal;
int range = 2

void setup() {
  Serial.begin(19200);
  adxl.powerOn();
  adxl.setRangeSetting(range);
}

void loop() {

  analogVal = analogRead(analogPin);

  Serial.println(" A: ");

  // Calibrare Sensor Hall
  Serial.println(analogVal - 521);
```

```
  int x, y, z;
  adxl.readAccel(&x, &y, &z);

  // Calibrare Accelerometru  -1000 < x, y, z < 1000

      x = x * 3.9;
      y = y * 3.9;
      z = z * 3.9;

  Serial.print(" X: ");
  Serial.print(x);

  Serial.print(" Y: ");
  Serial.print(y);

  Serial.print(" Z: ");
  Serial.println(z);

}
```

5. Colectarea datelor

După ce am verificat funcționalitatea sistemului, putem începe colectarea datelor generate de cei doi senzori. Dar mai întâi v-a trebui să conectăm bobina la sursa de curent. Pentru aceasta avem nevoie de două fire care se vor duce de la plusul si minusul sursei la capetele celor doua bobine.



Pentru a genera grafice 3D v-om folosi limbajul de programare Python. De aceea datele citite de senzori le v-om scrie într-un fișier folosind aplicația CoolTerm (este o aplicație simplă de terminal cu port serial, utilizată pentru schimb de date cu hardware-ul conectat la porturi seriale).

Datele v-or fi de forma: “ A: 0 X: 0 Y: 0 Z: 1000 …“

Unde **A** este intensitatea câmpului iar **X, Y** și **Z** sunt accelerațiile

6. Prelucrarea datelor:

Acum că avem toate datele într-un fișier, putem realiza programul ce v-a citi acele date pe care le vom adăuga în vectori.

```
f = open("data.txt", "r")
lines = f.readlines()
f.close()

lines = str(lines)
text = lines.split(" ")

A = []
x = []
y = []
z = []

nr = []
for data in text:
    nr.append(data)

for i in range(1, len(nr)-7, 8):
    A.append(int(nr[i].split("' ")[0]))
    x.append(int(nr[i + 2].split("' ")[0]))
    y.append(int(nr[i + 4].split("' ")[0]))
    z.append(int(nr[i + 6].split("' ")[0]))
```

Astfel am creat patru vectori, unul pentru intensitatea câmpului iar ceilalți trei pentru accelerațiile pe x, y, z.

Pentru a putea reprezenta grafic intensitatea câmpului magnetic, trebuie să știm deplasarea, pe care o putem determina din accelerație.

Viteza v_B la intervalul t_B este integrala accelerației:

$$v_B - v_A = \int_{t_A}^{t_B} \left(\frac{a_B - a_A}{t_A} t + (2a_A - a_B) \right) dt => v_B - v_A = \frac{1}{2} t_A (a_B + a_A)$$

$$x_B - x_A = \int_{t_A}^{t_B} \left(\frac{v_B - v_A}{t_B - t_A} t + (2v_A - v_B) \right) dt => x_B - x_A = \frac{1}{2} t_A (v_B + v_A)$$

Formula generală pentru deplasare atunci când accelerația se modifică pentru fiecare interval de timp este :

$$x_{n+1} - x_n = \left(\frac{1}{2} a_{n+1} + \frac{3}{2} a_n + 2 \sum_{j=1}^{n-1} a_j \right) * \frac{t^2}{2}$$

Iar din această formulă putem determina deplasarea pentru toate elementele din vector:

```
N = len(a)
x = [0] * N

x[1] = (0.5 * a[1] + 1.5 * a[0])*(t**2 / 2)

for i in range(2, N):
   x[i] = (0.5*a[i] + 1.5*a[i-1] + 2*sum(x[0:i-2]))*(t**2 / 2) + x[i-1]
```

A doua metodă ar fi să folosim regula trapezoidului pentru a calcula mai întâi viteza și apoi din viteză să aflăm poziția:

```
dt = np.linspace(0.0, N * timeStep, N)
v = scipy.integrate.cumtrapz(a, dx = dt, initial = 0)
x = scipy.integrate.cumtrapz(v, dx = dt)
```

Teoretic metoda de integrare dublă ne va oferi ceea ce avem nevoie, dar în practică integrarea dublă acumulează erori mari de măsurare, astfel încât rezultatele nu sunt niciodată exacte sau de încredere pentru a fi utilizate pentru găsirea poziției. De aceea este necesar de folosit filtre pentru a micșora aceste erori.

Un astfel de filtru este implementat mai jos:

```
timeStep = 0.017
time = np.linspace(0.0, N * timeStep, N)
dt = mean(diff(time))
fs = 1/dt;  # frecvența

# filtru high-pass
N = 2
```

```
fc = 0.5; # Hz
[B, C] = signal.butter(N, 2*fc/fs, 'high')
x = signal.lfilter(B, C, x)
```

Pentru reprezentarea grafică a datelor am folosit biblioteca matplotlib. Care ne oferă posibilitatea de a vizualiza sub formă de heatmap pozitia împreună cu intensitatea câmpului.

```
from matplotlib import cm
import matplotlib.pyplot as plt


fig = plt.figure(figsize=(12,12))

ax = fig.add_subplot(111,projection='3d')
colmap = cm.ScalarMappable(cmap=cm.hsv)
A.pop()

A = [abs(i) for i in A]

colmap.set_array(A)

A = [i/max(A) for i in A]
print(A)
yg = ax.scatter(x_displacement, y_displacement, z_displacement,
c=cm.hsv(A), marker='o')
cb = fig.colorbar(colmap)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

plt.show()
```

Programul în întregime utilizând metoda trapezoidului este:

```
from matplotlib import cm

import matplotlib.pyplot as plt
import numpy as np
from pylab import *
import scipy.integrate
from scipy import signal

f = open("data.txt", "r")
lines = f.readlines()
f.close()
```

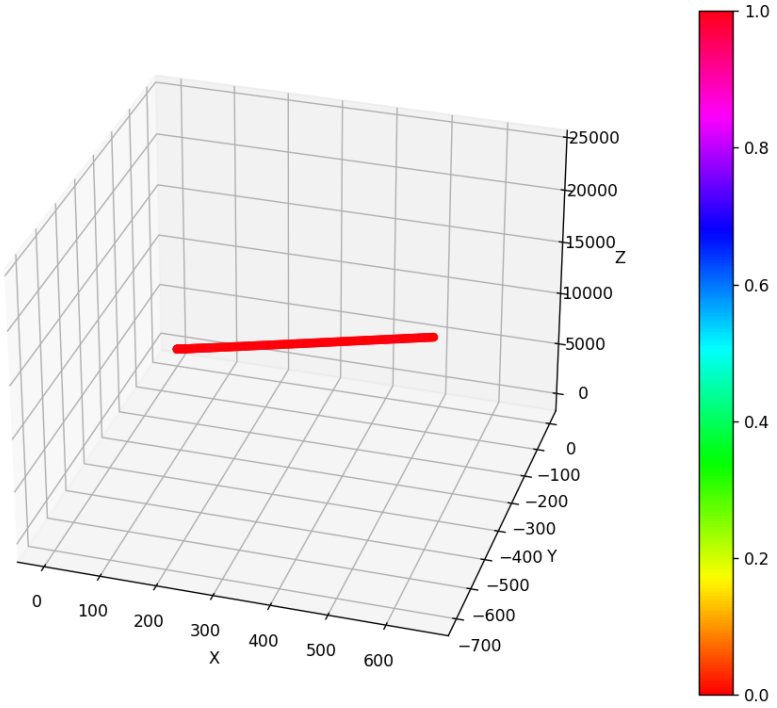
```
lines = str(lines)
text = lines.split(" ")

A = []
x = []
y = []
z = []

nr = []
for data in text:
    nr.append(data)

for i in range(1, len(nr)-7, 8):
    A.append(int(nr[i].split("' ")[0]))
    x.append(int(nr[i + 2].split("' ")[0]))
    y.append(int(nr[i + 4].split("' ")[0]))
    z.append(int(nr[i + 6].split("' ")[0]))


x = signal.detrend(x)
y = signal.detrend(y)
z = signal.detrend(z)

N = len(x)
timeStep = 0.017  # pasul pentru fiecare inregistrare
time = np.linspace(0.0, N * timeStep, N) # vector durata
dt = mean(diff(time))
fs = 1/dt;  # frequency

# filtrul “high pass”
N = 2
fc = 0.5; #Hz
[B,C] = signal.butter(N,2*fc/fs,'high')

x = signal.lfilter(B, C, x)
y = signal.lfilter(B, C, y)
z = signal.lfilter(B, C, z)

# din accelerație in viteză
x_velocity = scipy.integrate.cumtrapz(x, dx = dt, initial = 0)
y_velocity = scipy.integrate.cumtrapz(y, dx = dt, initial = 0)
z_velocity = scipy.integrate.cumtrapz(z, dx = dt, initial = 0)


x_velocity = detrend(x_velocity)
y_velocity = detrend(y_velocity)
z_velocity = detrend(z_velocity)

# din viteză in poziție
```

```
x_displacement = scipy.integrate.cumtrapz(x_velocity, dx = dt)
y_displacement = scipy.integrate.cumtrapz(y_velocity, dx = dt)
z_displacement = scipy.integrate.cumtrapz(z_velocity, dx = dt)


fig = plt.figure(figsize=(12,12))
ax = fig.add_subplot(111, projection='3d')
colmap = cm.ScalarMappable(cmap=cm.hsv)
A.pop()
A = [abs(i) for i in A]
colmap.set_array(A)
A = [i/max(A) for i in A]
yg = ax.scatter(x_displacement, y_displacement, z_displacement,
c=cm.hsv(A), marker='o')
cb = fig.colorbar(colmap)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

plt.show()
```

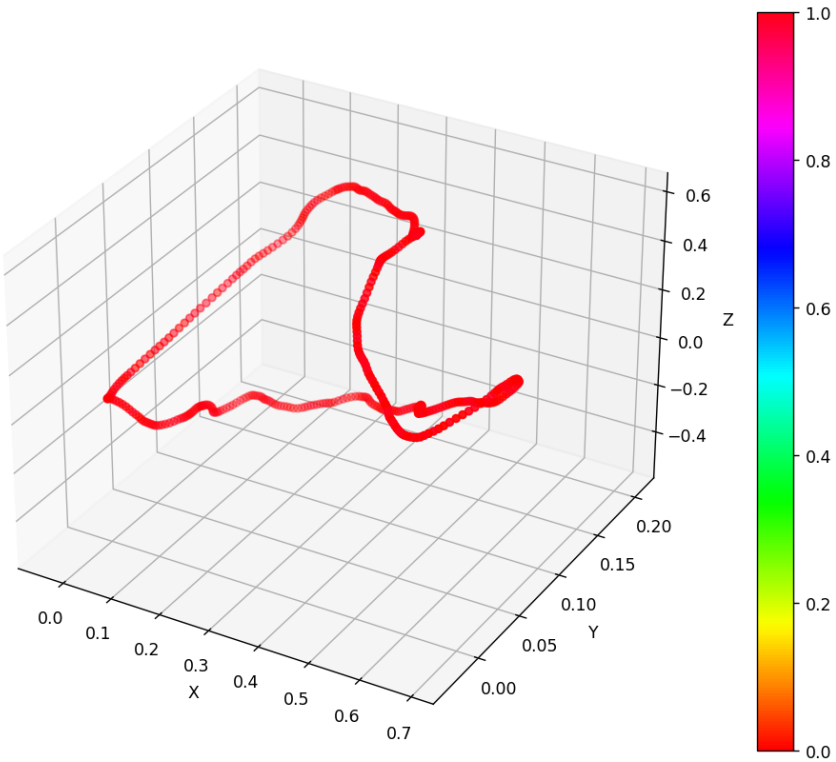
7. Vizualizarea datelor

1. Atunci când accelerometrul este nemișcat și nu acționează nici un câmp

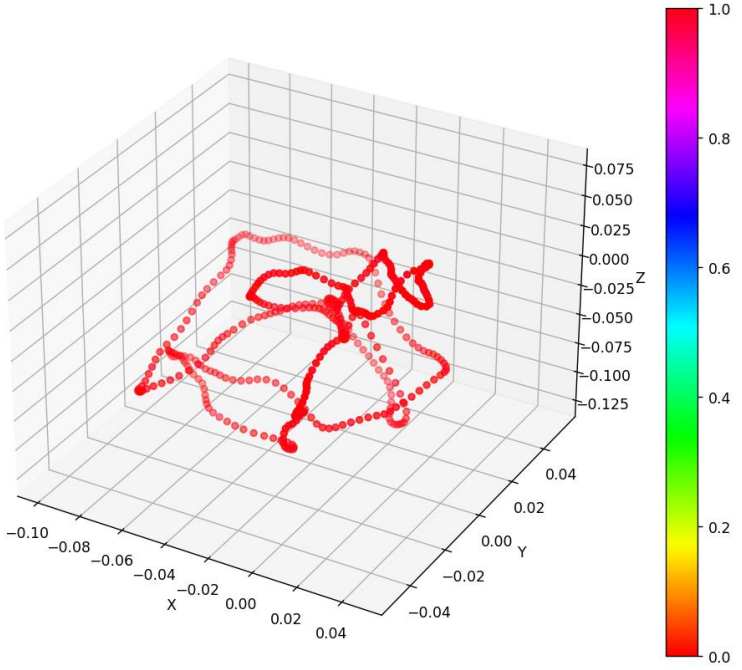
Fără filtrare și fără ”detrending”:



Doar ”detrending”:

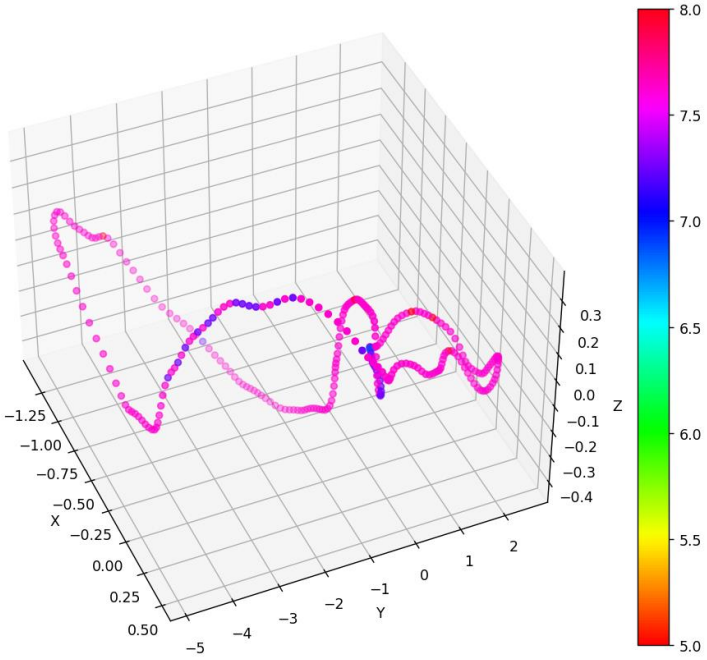


”Detrending” și filtrare ”high pass” :

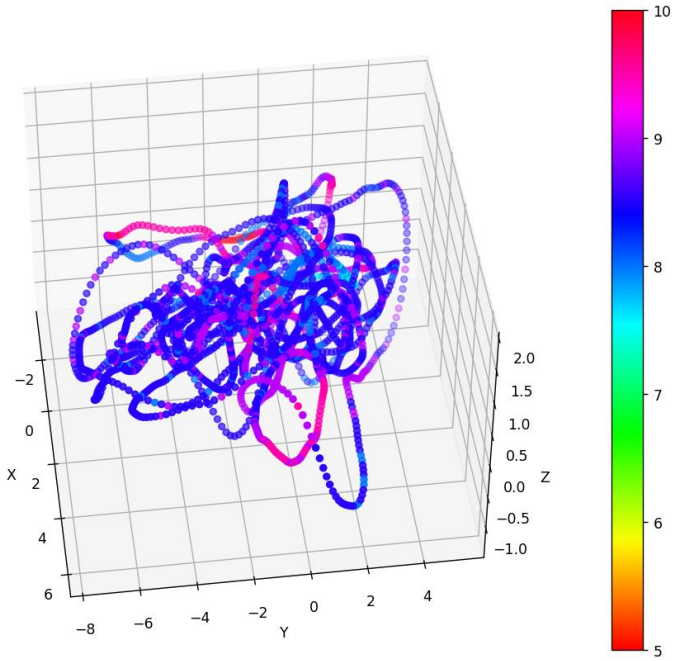


Putem observa că există o deplasare foarte mică din cauza zgomotelor iar intensitatea câmpului, după cum ne așteptam, este zero to timpul.

2. Atunci când sistemul este mișcat în mijlocul bobinelor



În acest grafic putem vedea că intensitatea câmpului se modifică foarte puțin



În schimb e mult mai greu de observat acest lucru atunci când avem multe date din cauza erorilor mari

8. Concluzii

În urma acestui experiment chiar dacă poziția nu era reprezentată foarte corect, s-a putut observa că intensitatea câmpului în interiorul bobinelor este constant sau cel puțin aproape constant.

De fapt, utilizarea doar accelerometrul nu vă va oferi coordonatele corecte. Să presupunem că axa verticală este axa z, iar axele x, y sunt în planul paralel cu solul. Problema este că accelerometrul nu are nicio modalitate de a determina care este „poziția" sau „orientarea" lui. Să presupunem că rotim accelerometrul în jurul axei z cu 90 de grade, deci ceea ce era mai devreme axa x devine acum y și y devine axa x negativă. Dar integrarea continuă, presupunând că axele sunt originale.

Pentru a rezolva asta, trebuie să folosim împreună cu el un giroscop (care îi indică viteza de rotație a axelor). De asemenea, ar fi utila utilizarea filtrelor mai complexe, deoarece din primele grafuri se poate observa că cel utilizat de noi nu ajută foarte mult.

Referințe

Bobina Helmholtz, https://en.wikipedia.org/wiki/Helmholtz_coil

Senzorul Hall, https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html

Reprezentare 3D, https://onelinerhub.com/python-matplotlib/how-to-plot-3d-heatmap

Accelerometru, https://makersportal.com/blog/2017/9/25/accelerometer-on-an-elevator

Filtru, https://koalatea.io/python-detrending-time-series/

Integrare trapezoidal, https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.integrate.cumtrapz.html

Determinare poziție, https://cris.brighton.ac.uk/ws/portalfiles/portal/219655/Displacement+from+Accelerometer+%281%29.pdf

Aplicația pentru scriere date in fișier (CoolTerm), https://freeware.the-meiers.org/