Pt. Ravishankar Shukla University
Center for Basic Sciences

PL-501

Physics Laboratory

Submitted To:
Mr. Saket Agrawal
Asst. Professor
Physics

Submitted By :
Hemant Kumar
1750616
V Semester

Contents

1 Aim: To demonstrate Electromagnetic Induction.

Apparatus:-

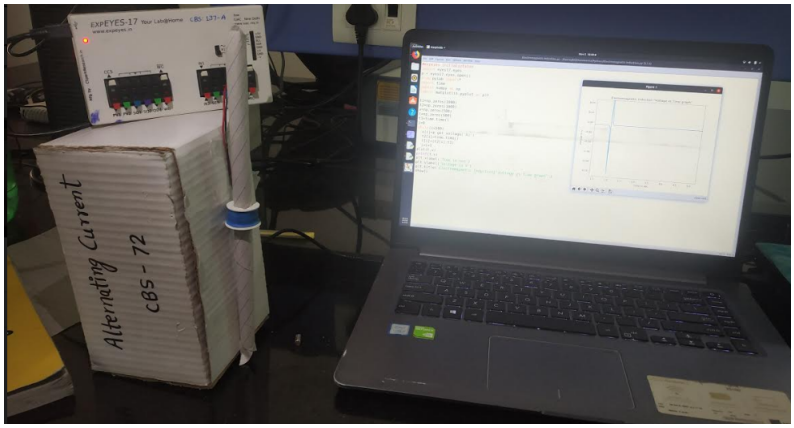
Expeyes-17, Magnet, coil, paper pipe.

Theory:-

Faraday's law of electromagnetic induction (referred to as Faraday's law) is a basic law of electromagnetism predicting how a magnetic field will interact with an electric circuit to produce an electromotive force (EMF). This phenomenon is known as electromagnetic induction.

Faraday's law states that a current will be induced in a conductor which is exposed to a changing magnetic field. Lenz's law of electromagnetic induction states that the direction of this induced current will be such that the magnetic field created by the induced current opposes the initial changing magnetic field which produced it. The direction of this current flow can be determined using Fleming's right-hand rule.

Faraday's law of induction explains the working principle of transformers, motors, generators, and inductors. The law is named after Michael Faraday, who performed an experiment with a magnet and a coil. During Faraday's experiment, he discovered how EMF is induced in a coil when the flux passing through the coil changes. According to Faraday's law of electromagnetic induction, the rate of change of flux linkage is equal to induced emf.
In these experinment we just take a coil and paper pipe through which small magnet is passed.



Python code:-

```
#expeyes initializtaion
import eyes17.eyes
p = eyes17.eyes.open()
from pylab import*
import time
import numpy as np

v=p.get_voltage
t,v= p.capture1('A1', 3000000, 100)
plot(t,v)
print(t,v)
show()
```

Graphs:

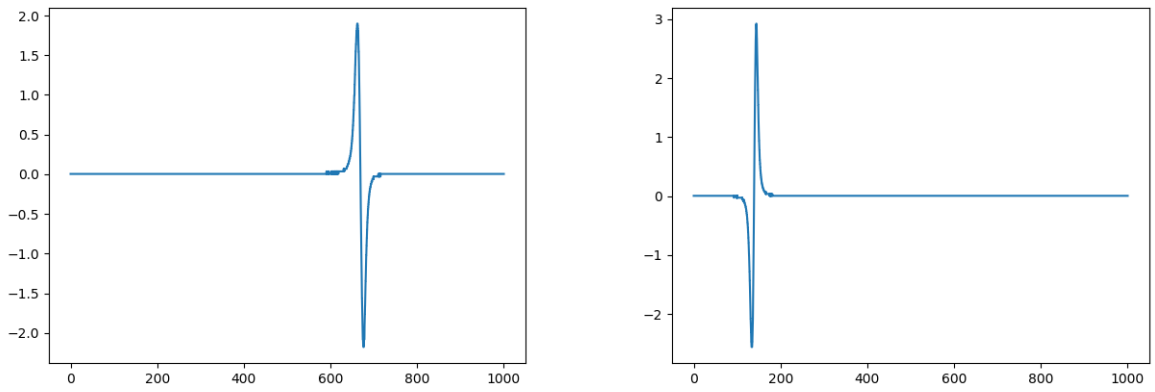


Figure 1: X-axis = Time (s), Y-axis = Electromotive force (V)

Result:-

We are getting a graph as per our expectation.

Sources of Errors:-

1. Connections should be proper and tight.
2. Magnet should be properly dropped through coil.
3. Set-up has to be vertical alinged.

2 Aim: To find natural frequency of coupled pendulum in different modes.

Materials Required

Pendulums x2, magnets x2, SR04 module, expeyes17, computer

Theory

Coupled pendulums using magnets have been designed, as opposed to classically used springs for coupling. Mathematical analysis of the system has not been done, but the results are expected to be similar to spring coupling. The main difference is that, while in case of spring force increases with displacement from mean position, in case of magnets force decreases with increasing distance of the two magnets.

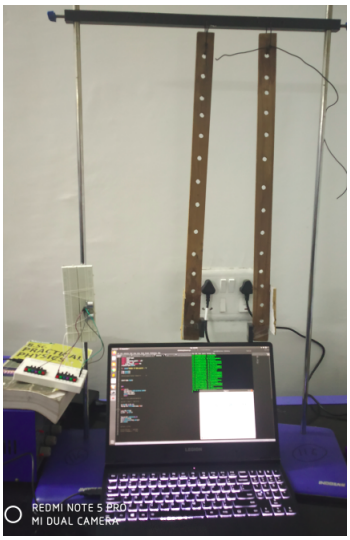


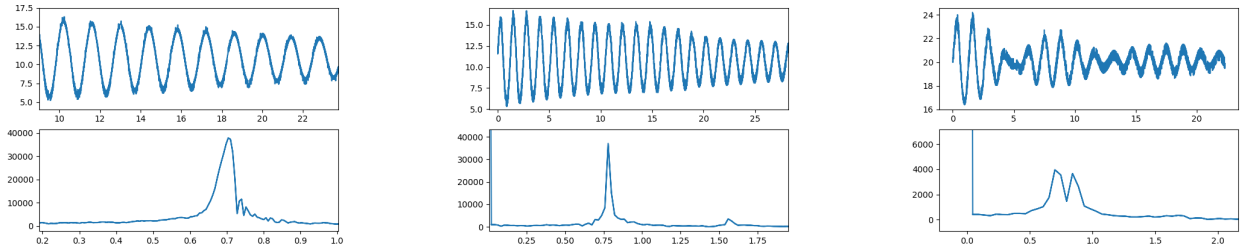
Figure 2: Setup

Python Code

```
import eyes17.eyes
import time
import matplotlib.pyplot as plt
import numpy as np
from scipy.fftpack import fft
p=eyes17.eyes.open()

l = input("Number of data points : ")

T=np.zeros(l)
D=np.zeros(l)

start=time.time()

i=0

while(i<l):
    T[i],D[i]=p.sr04_distance_time()
    T[i]=T[i]-start
    print(T[i],D[i])
    i=i+1

```

```
ss=abs(T[-1]-T[-2])
frq=np.fft.fftfreq(D.size,d=ss)


y=np.fft.fft(D)
y_sort = y.sort()


fig,ax=plt.subplots(2,1)
ax[0].plot(T,D)
ax[1].plot(abs(frq),abs(y))
plt.show()
```

Graph



(a) In-Phase (b) Out of Phase (c) Beats

Figure 3: Distance vs Time graphs for oscillations of coupled pendulums

Result

The frequency of various modes in this experiment are as follows :
In phase : 0.7 Hz
Out of phase : 0.8 Hz
Beat : 0.7 Hz and 0.8 Hz

The results are in accordance with the expectations, as the in-phase frequency is seen to be less than the out of phase frequency.
The beat frequency is f1 - f1 = 0.8 - 0.7 = 0.1 Hz

Sources of errors

1. Pendulums should oscillate in only one direction.
2. Connections should be proper.
3. Air resistance should be minimised (switch of fans)
4. Magnets should be strong enough to exchange energy at set distance.
5. Echo module should be stable.

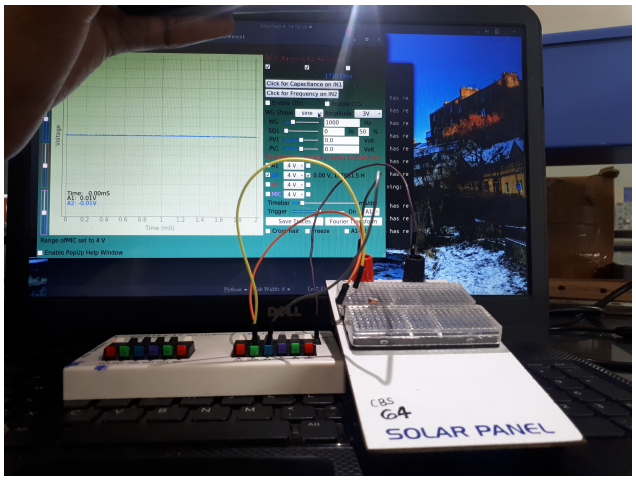
3 Aim: To study variation in voltage with intensity of the given solar panel.

Apparatus:-

Light Dependent Resistors(LDR), Expeyes, solar panel, torch, cello tape.

Theory:-

We can simply guess that as we apply more intensity of light on LDR and solar cell, LDR becomes less resistive and solar gives more voltage, so that they are inversely praprotional in relation and we get graph according to that.

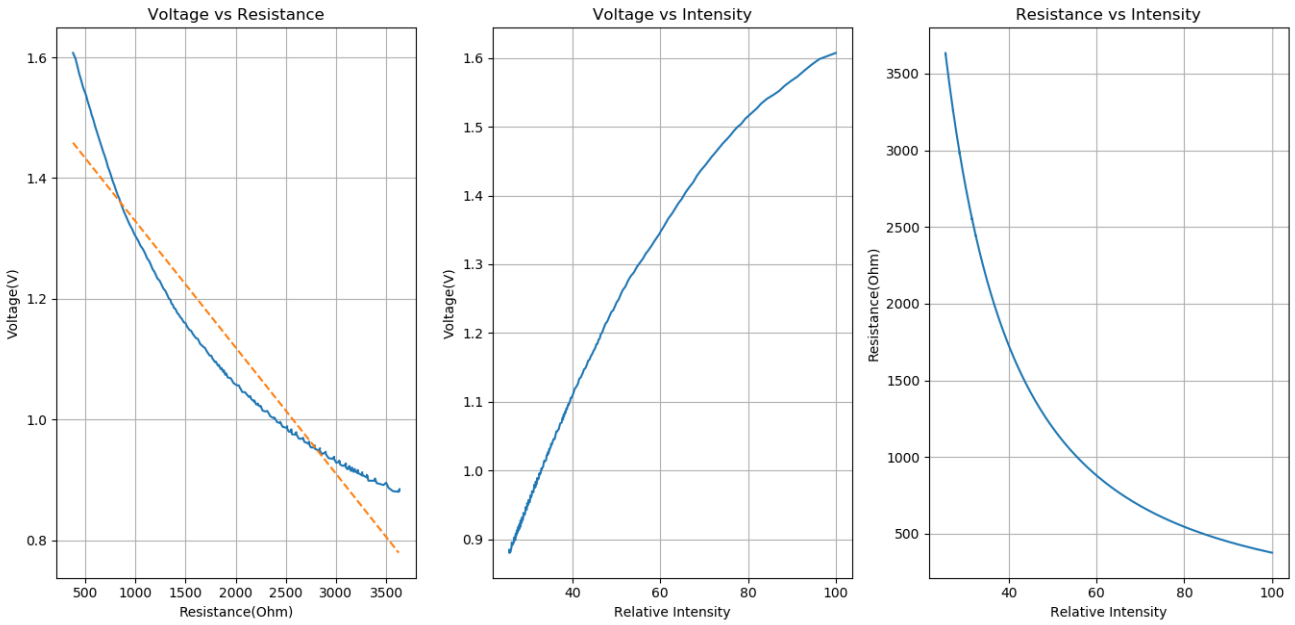


Python Code

```
import matplotlib.pyplot as plt
import numpy as np
import eyes17.eyes
from scipy import stats
p = eyes17.eyes.open()

v=np.zeros(200)
r=np.zeros(200)
I=np.zeros(200)
I[0]=100
r[0]=p.get_resistance()
v[0]=(p.get_voltage('A2'))
for i in range(1,200):
 v[i]=(p.get_voltage('A2'))
 r[i]=p.get_resistance()
 I[i]=I[i-1]*pow((r[i-1]/r[i]),0.6)
 # code for least square fit
m, c,  corr, s, err=stats.linregress(r,v)
r1=np.linspace(r[0],r[199])
v1=m*r1+c
fig, ax=plt.subplots(1,3)
s="Voltage vs Resistance"
ax[0].set_title(s)
ax[0].plot(r,v,'-',r1,v1,'--')
ax[0].set_xlabel("Resistance(Ohm)")
ax[0].set_ylabel("Voltage(V)")
ax[0].grid()
s="Voltage vs Intensity"
```

```
ax[1].set_title(s)
ax[1].plot(I,v,'-')
ax[1].set_xlabel("Relative Intensity")
ax[1].set_ylabel("Voltage(V)")
ax[1].grid()
s="Resistance vs Intensity"
ax[2].set_title(s)
ax[2].plot(I,r,'-')
ax[2].set_xlabel("Relative Intensity")
ax[2].set_ylabel("Resistance(Ohm)")
ax[2].grid()
plt.show()
```

Observation:-



Result:-

We are getting a graph as per our expectation.

Souces of Errors

1. Connections should be proper and tight.
2. Torch should be properly moving during the observation.
3. LDR should be kept properly with a surface of solar panel.

4 Aim: To demonstrate and design a weighing machine.

Apparatus:-

Spring, Some Masses, Spring Stand, HCSR04, Laptop, Expyes17.

Theory:-

This experiment is performed mainly in two steps
(i) Find k for spring and
(ii) To find unknown mass
Hooke's Law states that the restoring force of a spring is directly proportional to a small displacement (x)

$$F = -kx$$

where k is the spring constant. and x is small displacement.
So net force on suspended mass, in spring is $mg = kx$ (Accordings to Newton's 2^{nd} law). So,

$$k = \frac{mg}{x}$$

where g is 9.8 ms^{-2} using known masses we find spring constant. After measuring the spring constant k, we put unknown mass and find extention of spring(x) and using this formula

$$m = \frac{kx}{g}$$

we can easily find unknown mass.

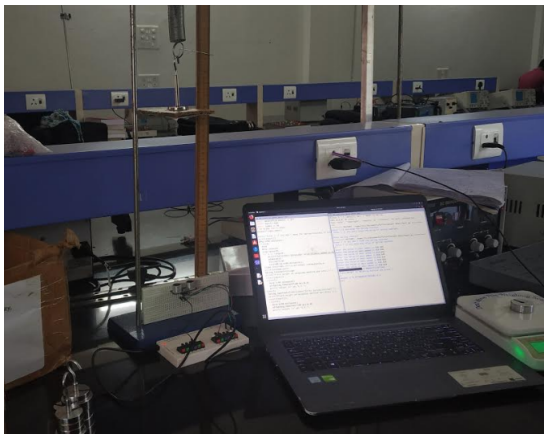


Figure 4: weighing machine

Python code:-

```
import matplotlib.pyplot as plt
import eyes17.eyes
import numpy as np
from scipy import stats
p=eyes17.eyes.open()

print("Enter 1 if you know the Spring Constant \n Enter 2 if you don't know the
    value of spring constant")
n=int(input())
d0=p.sr04_distance()
if(n==1):
  m1=0
  m=np.zeros(8)
```

```
  x=np.zeros(8)
  for i in range(8):
      m[i]=float(input("Enter the value of mass added in KG"))
      m1=m1+m[i]
      m[i]=m1*9.81
      x[i]=d0-(p.sr04_distance())
  slope,intercept,corr,s,err=stats.linregress(m,x)
  print(1/slope,corr)
  Spring_Constant=1/slope
  print("Place weight on weighing machine and press 1")
  c=int(input())
  if(c==1):
    dx=p.sr04_distance()
    wt=Spring_Constant*(d0-dx)/9.81
    print("weight is",wt,"K.G.")
if(n==2):
  Spring_Constant=float(input("Enter spring constant"))
  print("Place weight on weighing machine and press 1")
  c=int(input())
  if(c==1):
    dx=p.sr04_distance()
    wt=Spring_Constant*(d0-dx)/9.81
    print("weight is",wt,"K.G.")
```

Observation:-

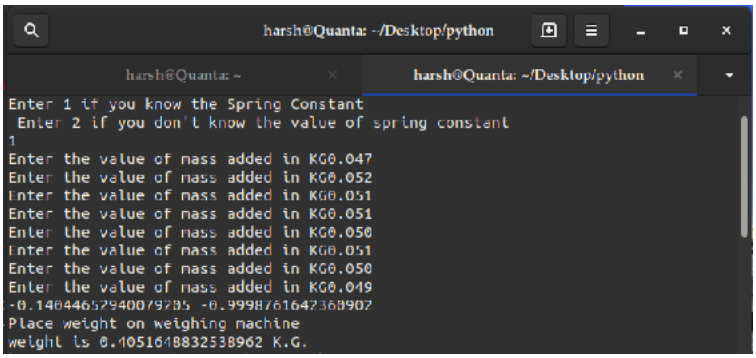


Figure 5: output

actual weight(a): 0.400 Kg
weight through weighing machine(b): 0.405 Kg
percentage error is

$$|\frac{a-b}{a}| * 100 = 1.25\%$$

Result:-

So our weighing machine provides weight with minimum error.

Sources of errors:-

(i) Wires should be proper.
(ii) Connections should be proper.
(iii) Masses should be placed such that the spring is stable during experiment.
(iv) It should be noted that air, magnet like materials not affect this experiment.

5 Aim: To determine the time period of variable mass pendulum with variable m

Apparatus:-

HCSR04 Distance Measurement Sensor,Bottle,Water, Clamp Stand

Theory:-

Our setup contains a bottle filled with sand suspended from a clamp stand.Sand comes out of the bottle at a constant rate while the bottle oscillates, as the mass of the bottle decreases centre of mass shifts downwards as a result time period increases and when bottle is empty more than the com of empty bottle then com shifts upwards as a result timperiod decreases.

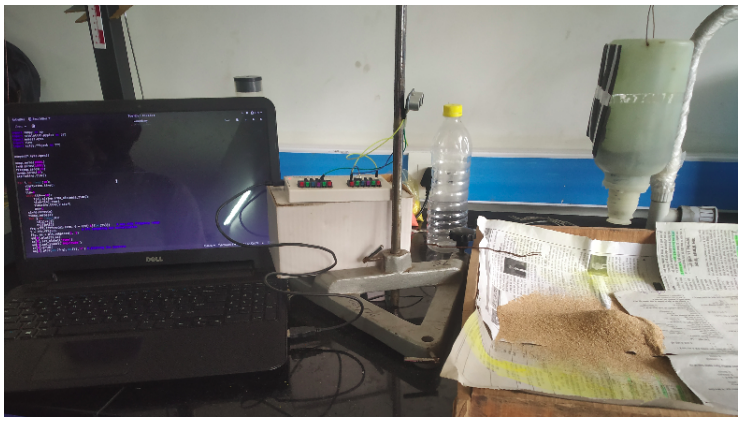


Figure 6: Setup

Python Code

```
import numpy as np
import matplotlib.pyplot as plt
import eyes17.eyes
import time
import scipy.fftpack as fft


p=eyes17.eyes.open()

x=np.zeros(10001)
t=np.zeros(10001)
timp=np.zeros(10)
start=time.time()

for i in range(10):
    start=time.time()
    a=0
    tim=0
    while(tim<=2):
        t[a],x[a]=p.sr04_distance_time()
        t[a]=t[a]-start
        tim=time.time()-start
        a=a+1
    x1=np.zeros(a)
    t1=np.zeros(a)
    for j in range(a):
```

```
        x1[j]=x[j]
        t1[j]=t[j]
    tp =fft.fftfreq(x1.size, d = (t1[-1]/len(t1)))    # one side frequency range
    Y = fft.fft(x1)
    maxi=np.max(abs(Y))
    for j in range(len(Y)):
        if(abs(Y[j])==maxi):
            Y=np.delete(Y,j)
            tp=np.delete(tp,j)
            break
    maxi=np.max(abs(Y))
    for j in range(len(Y)):
        if(abs(Y[j])==maxi):
            timp[i]=tp[j]
            break
    print(timp[i])
```

Observation:-

```
harsh@Quanta:~/Desktop/python$ python3 variable.py
0.6998506292621336
0.6998813258173653
0.6999027315849684
0.6999103565589087
0.6999384218984411
0.6999639863423699
0.6999680915119416
0.6999241052906308
0.699919616871012
0.6999073365821828
```

Result:-

The observation is as per the theory.

Sources of errors:-

(i) Wires should be proper.
(ii) Connections should be proper.
(iii) Oscillations must be constrained to one degree of freedom .
(iv) It should be noted that air, magnet like materials not affect this experiment.

6 Aim: To make photogate with expeyes.

Apparatus:-

Expeyes-17, LDR, Torch.

Theory:-

Code can be developed with python by expes17 module. Photogate can be used for many purposes like to estimate acceleration, velocity, stopwatch, Distance. There is an LDR when the amount of light changes suddenly then LDR's resistance alsp change. This property of LDR is used to make Photogate with Expeyes.

Figure 7: Setup

Python code:-

```
import time
import numpy as np
import eyes17.eyes
import time
p=eyes17.eyes.open()


print("\tWelcome")
print("Enter your choice")
print("1. Stopwatch \n2. Velocity\n3. Acceleration\n4. g by freefall\n5. Time
    Period ")
choice=int(input())

def stopwatch():
    print ("\tStopwatch")
    print (" Enter your Choice")
    print (" 1.Start\n2. Exit")
    ch=int(input())
    if ch==1 :
        start=time.time()
```

```
        print (" Enter 1 for lap and anything else to Stop")
        n=int(input())
        lap=np.zeros(200)
        l=1
        while n==1 :
            if n==1 :
                lap[l]=time.time()-start
                l=l+1
            n=int(input())
        stop=time.time()-start
        print (" Time elapsed is ",stop)
        print ("laps are")
        for i in range(l):
            print(i,lap[i])

def velocity():
    x=float(input("Enter distance between pickets or photogates"))
    n=1
    while n==1 :
        a=p.get_resistance()
        b=p.get_resistance()
        while (abs(a-b)<5000):
          b=p.get_resistance()
        start1=time.time()

        a=p.get_resistance()
        b=p.get_resistance()
        while (abs(a-b)<5000):
          b=p.get_resistance()
        end1=time.time()
        v=x/(end1-start1)
        print(" Velocity is ",v," m/sec")
        print("Press 1 to continue or any other key to quit")
        n=int(input())

def acceleration():
    x=float(input("Enter distance between pickets or photogates"))
    n=1
    while n==1 :
        a=p.get_resistance()
        b=p.get_resistance()
        while (abs(a-b)<5000):
          b=p.get_resistance()
        start1=time.time()
        start=time.time()
        a=p.get_resistance()
        b=p.get_resistance()
        while (abs(a-b)<5000):
          b=p.get_resistance()
        end1=time.time()
        v=x/(end1-start1)
        print(" Velocity is ",v," m/sec")
        a=p.get_resistance()
        b=p.get_resistance()
        while (abs(a-b)<5000):
          b=p.get_resistance()
        start1=time.time()

        a=p.get_resistance()
        b=p.get_resistance()
        while (abs(a-b)<5000):
```

```
          b=p.get_resistance()
        end1=time.time()
        end=time.time()
        v2=x/(end1-start1)
        print(" Velocity is ",v2," m/sec")
        a=(v2-v)/(end-start)
        print("Acceleration is ",a," m/(sec*sec)")
        print("Press 1 to continue or any other key to quit")
        n=int(input())

def g_freefall():
     x=float(input("Enter distance between pickets or photogates"))
     s=float(input("Enter distance between point of drop and photogate"))
     n=1
     while n==1 :
        a=p.get_resistance()
        b=p.get_resistance()
        while (abs(a-b)<5000):
          b=p.get_resistance()
        start1=time.time()

        a=p.get_resistance()
        b=p.get_resistance()
        while (abs(a-b)<5000):
          b=p.get_resistance()
        end1=time.time()
        v=x/(end1-start1)
        print(" Velocity is ",v," m/sec")
        g=(v*v)/(2*s)
        print("Acceleration due to gravity is ",a," m/(sec*sec)")
        print("Press 1 to continue or any other key to quit")
        n=int(input())

def time_period():
      start=time.time()
      a=np.zeros(100000)
      t=np.zeros(100)
      i=1
      j=0
      a[0]=p.get_resistance()
      while i<=10000 :
         a[i]=p.get_resistance()
         if((a[i]-a[i-1]) >150):
             t[j]=time.time()-start
             print("time when pendulum in mid position",t[j])
             j=j+1
         i=i+1
      n=len(t)
      sum_=0
      for i in range(n-1):
          sum_=sum_+t[j+1]-t[j]
      sum_=sum_/n
      print("Average time of half oscillation =",sum)
      print("program execution time =",time.time()-start)
if choice==1 :
   stopwatch()

elif choice==2 :
   velocity()

elif choice==3 :
```

```
    acceleration()

elif choice==4 :
    g_freefall()

elif choice==5 :
    time_period()

else :
    print("Invalid Choice")
```

Observation:-

```
========== RESTART: /media/raju/HP_TOOLS/sp1.py ==========
time when pendulum in mid position 0.29852747917175293
time when pendulum in mid position 0.7593238353729248
time when pendulum in mid position 1.2512247562408447
time when pendulum in mid position 1.7123041152954102
time when pendulum in mid position 2.204580783843994
time when pendulum in mid position 2.6660215854644775
time when pendulum in mid position 3.158275842666626
time when pendulum in mid position 3.619274377822876
time when pendulum in mid position 4.111788749694824
time when pendulum in mid position 4.573041677474976
time when pendulum in mid position 5.06516170501709
time when pendulum in mid position 5.526620864868164
time when pendulum in mid position 6.018871545791626
Average time of half oscillation =  0.47251033782958984
program execution time =  6.307851791381836
>>>
========== RESTART: /media/raju/HP_TOOLS/sp1.py ==========
time when pendulum in mid position 0.4268193244934082
time when pendulum in mid position 0.9184958934783936
time when pendulum in mid position 1.3764410018920898
time when pendulum in mid position 1.8707034587860107
time when pendulum in mid position 2.329728126525879
time when pendulum in mid position 2.823976993560791
time when pendulum in mid position 3.2817819118499756
time when pendulum in mid position 3.7759861946105957
time when pendulum in mid position 4.23481297492981
time when pendulum in mid position 4.7286765575408936
time when pendulum in mid position 5.186977863311768
time when pendulum in mid position 5.681791305541992
Average time of half oscillation =  0.4785749912261963
program execution time =  5.960067987442017
>>>
========== RESTART: /media/raju/HP_TOOLS/sp1.py ==========
time when pendulum in mid position 0.39443182945251465
time when pendulum in mid position 0.850867509841919
time when pendulum in mid position 1.3465759754180908
time when pendulum in mid position 1.7864456176757812
time when pendulum in mid position 2.293006181716919
time when pendulum in mid position 2.756153106689453
time when pendulum in mid position 3.2513840198516846
time when pendulum in mid position 3.7082114219665527
time when pendulum in mid position 4.204369783401489
time when pendulum in mid position 4.660188674926758
time when pendulum in mid position 5.157089948654175
time when pendulum in mid position 5.612689256668091
```

```
time when pendulum in mid position 6.109769821166992
Average time of half oscillation =  0.4772038459777832
program execution time =  6.476891279220581
```

Final average half oscillation time is 0.4760 second
So, the time period is 0.9521 second
Time period calculated from stopwatch is 0.9485 second
Percentage error is 0.36%.

Result:-

As observed and expected value are almost same, so our photogate works efficiently.

Sources of errors:-

(i) Wires should be proper.
(ii) Connections should be proper.
(iii) Oscillations must be constrained to one degree of freedom .
(iv) It should be noted that air, magnet like materials not affect this experiment.
(v) Diameter of penulum rod must be more than the LDR