MEASUREMENT OF g BY FREEFALL

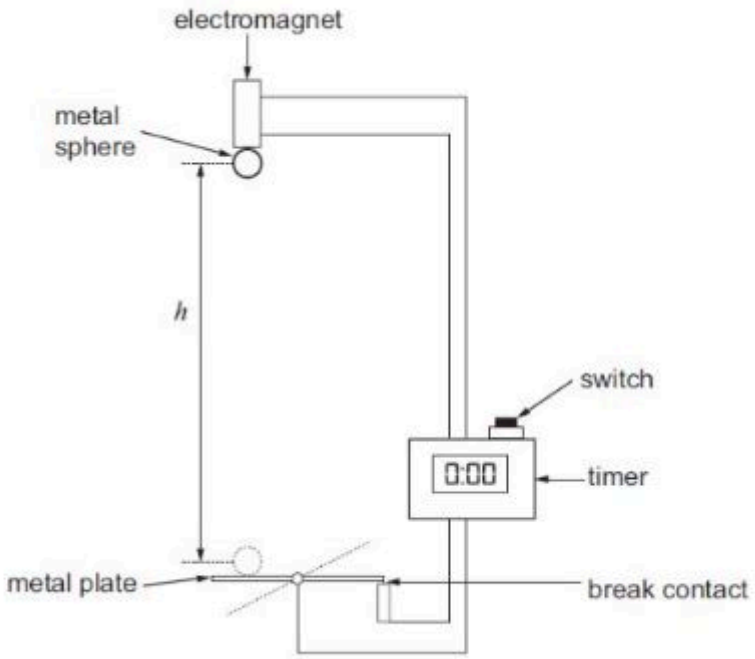
Theory:

An equation of motion can be used to calculate the acceleration due to gravity, g. $s = ut + 1/2at^2$

Where: u = initial velocity = 0, s = height, h and a = acceleration due to gravity, g

If a graph of height, 2h, (y-axis) is plotted against time squared, t^2, (x-axis) the gradient will equal g.

Apparatus:



Experimental Method:

This part needs changing as this was not the method used.

When the switch is pressed it disconnects the electromagnet releasing the metal sphere. At the same instant the timer starts. When the sphere hits the magnetic switch it breaks the circuit stopping the timer, thus recording the time it takes for the sphere to fall through a height, h. The time taken for the ball bearing to fall through a range of different heights needs to be measured. Plot a graph of height, h, (y-axis) against time squared, t^2, (x-axis) and calculate the value of g using: g = 2 × gradient.

In [4]:
```
!pip install tabulate
from matplotlib import pyplot as plt
from scipy import stats
from tabulate import tabulate

#Taking mean
# This is the number of different heights in this example
NumRepeats=3
#This is the resolution of the timer.
Resolutiony = 0.05
Resolutionx = 0.01
#This is a list of list of the multiple readings of time at each height
Xmultraw=[[0.44,0.41,0.44],[0.49,0.47,0.50],[0.50,0.53,0.52],[0.57,0.56,0.54],[0.60,0.62,0.59],[0.61,0.66,0.63],[0.67,0.67,0.67],
#This is the height measurements
yraw = [1.00,1.20,1.40,1.60,1.80,2.00,2.20,2.40]


# Looping over the number of heights. Calculating the average time and the uncertainty in those times and adding them to the
# list xraw and uncertainty
xraw=[]
uncertaintyxraw=[]
for i in range (len(yraw)):
    sumx=0.0
    for j in range (NumRepeats):
        sumx+=Xmultraw[i][j]
    HalfRange=(max(Xmultraw[i])-min(Xmultraw[i]))/2.0
    ave=sumx/NumRepeats
    if HalfRange < Resolutionx:
        HalfRange=Resolutionx
    uncertaintyxraw.append(HalfRange)
    xraw.append(ave)
```

```
# looping over the length of y and calculation 2h and t^2 so that the correct thing is plotted against each other
y=[]
x=[]
uncertaintyx=[]
uncertaintyy=[]
for i in range (len(xraw)):
    y.append(2*yraw[i])
    x.append(xraw[i]*xraw[i])
    uncertaintyx.append(2.0*(uncertaintyxraw[i]/xraw[i])*x[i])
    uncertaintyy.append(Resolutiony*2)

# merging the varaiables into one list of lists so tht tabulate can cope
merged_list = [(y[i],*Xmultraw[i],xraw[i],uncertaintyxraw[i],x[i],uncertaintyx[i]) for i in range (0,len(x))]
labels=['2 x h (m)','Time 1 (s)','Time 2 (s)','Time 3 (s)','Mean t (s)', 'unc t (s)', 'Mean t^2 (s)','unc t^2 (s^2)']
print(tabulate(merged_list, headers=labels,tablefmt='fancy_grid',floatfmt=".2f"))
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: tabulate in c:\programdata\anaconda3\lib\site-packages (0.8.10)
```

2 x h (m)	Time 1 (s)	Time 2 (s)	Time 3 (s)	Mean t (s)	unc t (s)	Mean t^2 (s)	unc t^2 (s^2)
2.00	0.44	0.41	0.44	0.43	0.02	0.18	0.01
2.40	0.49	0.47	0.50	0.49	0.02	0.24	0.01
2.80	0.50	0.53	0.52	0.52	0.02	0.27	0.02
3.20	0.57	0.56	0.54	0.56	0.01	0.31	0.02
3.60	0.60	0.62	0.59	0.60	0.02	0.36	0.02
4.00	0.61	0.66	0.63	0.63	0.03	0.40	0.03
4.40	0.67	0.67	0.67	0.67	0.01	0.45	0.01
4.80	0.70	0.70	0.70	0.70	0.01	0.49	0.01

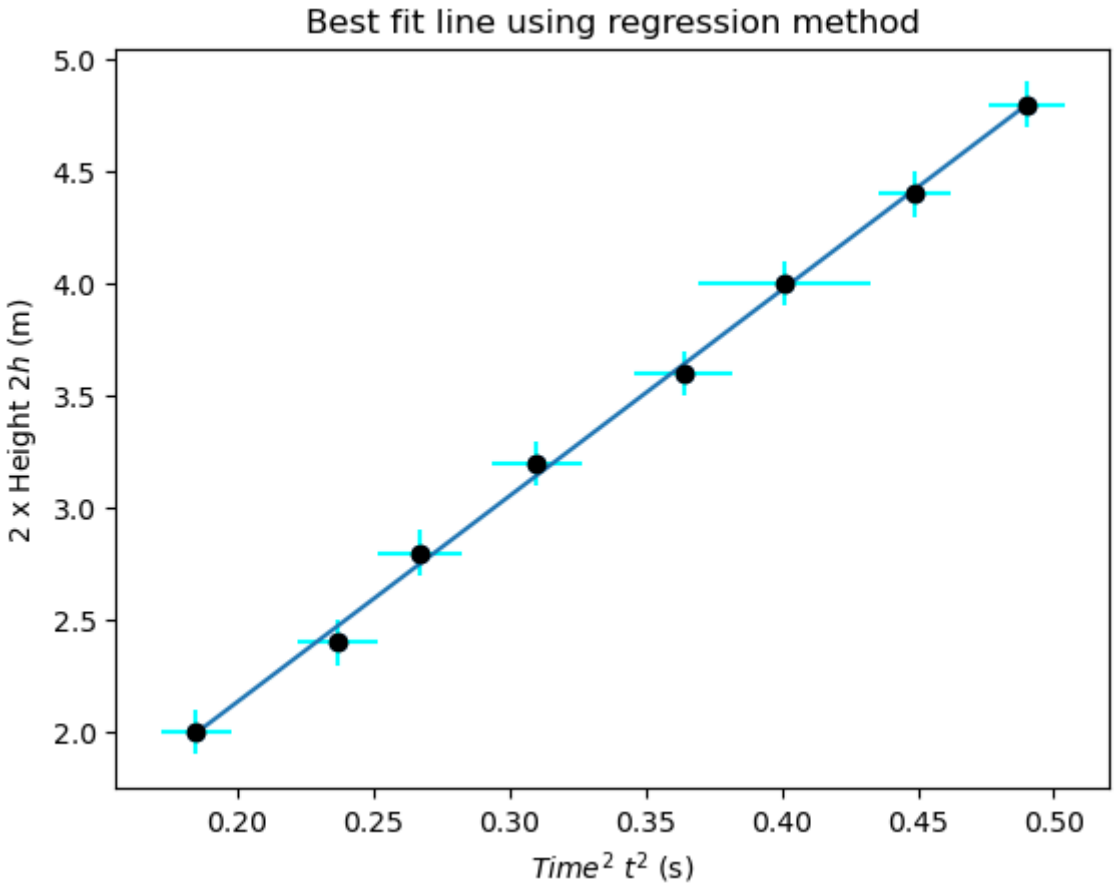
Analysis

In [5]:
```
from scipy import stats
# Calling linear regression assigning the output to the variables on the left of the equation
```

```
gradient, intercept,r,p,st_err = stats.linregress(x,y)
#Looping over the length of x and applying y=mx+c using the m and c from the output above to find the regression line
line=[]
for i in range (len(x)):
    line.append(gradient*x[i]+intercept)
# Plotting the data points and the best fit line
plt.scatter(x, y)
plt.errorbar(x,y,xerr=uncertaintyx,yerr=uncertaintyy,fmt='o',ecolor='cyan', color='black')
plt.plot(x,line)
plt.title('Best fit line using regression method')
plt.ylabel('2 x Height $2h$ (m)')
plt.xlabel('$Time^2$ $t^2$ (s)')

plt.show()
print ("Gradient=%.2f"%gradient,'ms-2')
print("Our y intercept should be zero it is equal to = ",intercept)
#This part is with own linear regression
#Calculating mean
sumx=0.0
sumy=0.0
for i in range (len(x)):
    sumx+=x[i]
    sumy+=y[i]
xbar=sumx/len(x)
ybar=sumy/len(y)

#Calculating sxx and sxy
sxy=0.0
sxx=0.0
for i in range (len(x)):
    sxy+=((x[i]-xbar)*(y[i]-ybar))
    sxx+=(x[i]-xbar)**2
#Calculating new gradient and intercept
newgradient=sxy/sxx
print(newgradient)
newintercept=ybar-newgradient*xbar
print(newintercept)
```




```
Gradient=9.18 ms-2
Our y intercept should be zero it is equal to =  0.29863797357735233
9.180418195821726
0.2986379735773528
```

Conclusion

In [6]:
```
print('Our calculated value of %.2f m/s^2 is close the value stated in our data booklet of 9.81 m/s^2' %gradient)
print('The line of best fit goes through all the error bars. This means we have verified the law and found a reasonable value of
```

```
Our calculated value of 9.18 m/s^2 is close the value stated in our data booklet of 9.81 m/s^2
The line of best fit goes through all the error bars. This means we have verified the law and found a reasonable value of g.
```