Maker Portal

# Introduction To Capacitive MEMS Accelerometers and A Case Study On An Elevator

*October 19, 2017 · Joshua Hrisko*

For many, acceleration is a concept that is taught in elementary physics but rarely matures past a discussion of gravity: think of a ball being thrown in the air, a pendulum swinging, or a cart rolling down an incline. In engineering, however, acceleration describes many of the complex phenomena at work in the aerospace, automotive, architectural, and seismological fields - just to name a few. A measurement or approximation of acceleration relates to the forces acting on a system, which can prevent disasters or loss of resources.

## MASS-SPRING ANALOGUE

Consider the mass-spring system shown in Figure 1. Its differential equation is commonly written as:

$$\frac{d^2x}{dt^2} - \frac{kx}{m} = 0$$

where $k$ is the spring constant, $m$ is the mass of the object, and $x$ is displacement. We can say that the second time derivative of the displacement is the acceleration, and we end up with the following:

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

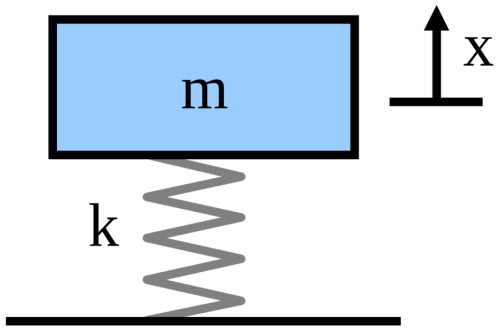



Figure 1: Mass-spring system.

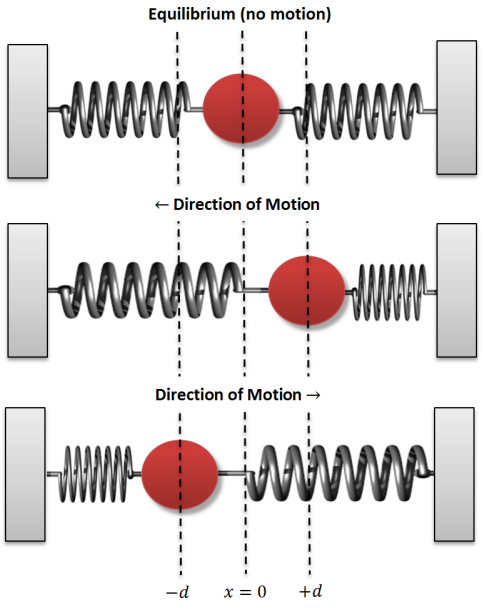



Figure 2: Double-walled mass spring system in a moving inertial reference frame. The walls are the primary while the ball moves in opposition to the

Now, imagine the double-walled mass-spring system shown in Figure 2. The only difference between the double-walled case and the simple mass-spring system above is a factor of two, assuming the springs are identical. It is also important to realize that the entire system moves, resulting in opposition of motion from the mass according to Newton's 2nd law, $F = ma$. Consequently, with prescribed mass and springs, the displacement of the mass is proportional to the accelerations experienced by the system. This problem is the first fundamental step toward creating what is called a capacitive accelerometer.

Maker Portal

## CAPACITIVE MEMS ACCELEROMETER

A micro-electromechanical system (MEMS) is a classification that describes microscopic devices made of both electrical and mechanical parts. A MEMS accelerometer is a device that measures acceleration by utilizing the techniques described above, but on a micro scale. The mass-spring system is the mechanical side of the MEMS device, and capacitance is the electrical portion. The capacitance of a parallel-plate capacitor is:

$$C = \frac{Q}{V} = \frac{A\epsilon}{d}$$

where C is capacitance, Q is charge, V is voltage, A is the area of the plate, $\varepsilon$ is called the permitivity (constant), and $d$ is the distance between the plates. Capcitance plays a key role in calculating the acceleration from the mass-spring system by measuring a change in voltage as the distance between two capacitive plates is changed.

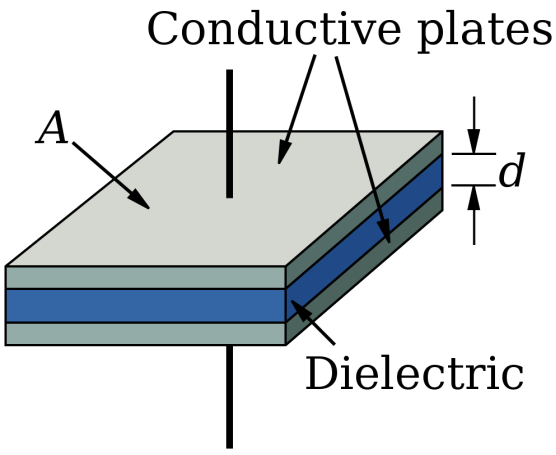
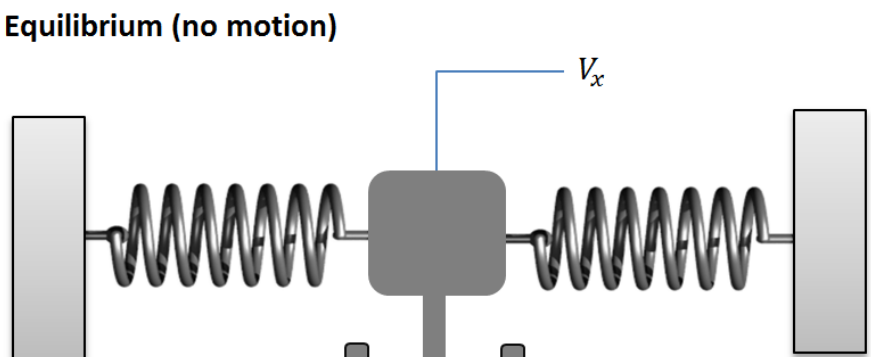



Figure 3: Basic diagram of a parallel plate capacitor.

Imagine now a mass-spring system with the adjustments shown in the following image:



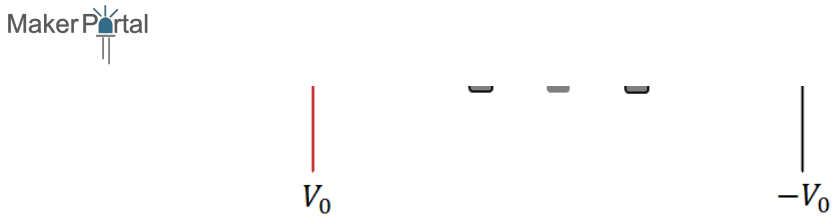



Figure 4: Simplified MEMS accelerometer in equilibrium.

The figure above shows the complete analogy for a MEMS accelerometer. There are two voltages, one applied, $V_0$, and one measured, $V_x$. In this diagram, the applied voltages create two capacitors each with a separation distance, $d$. It is pertinent also to remember the motion of the entire system, where the walls and plates of the capacitors with applied voltages move as a whole, and the mass (with its appending arm) is free to move in opposition to the motion. This results in a change in capacitance through change of separation distance between plates.

One method for balancing the system above is to break down the electrical contributions using conservation of charge:

$$Q_1 + Q_2 = 0$$

Furthermore, when the system is set into motion, the capacitances change and the setup can be explicitly modeled as follows:

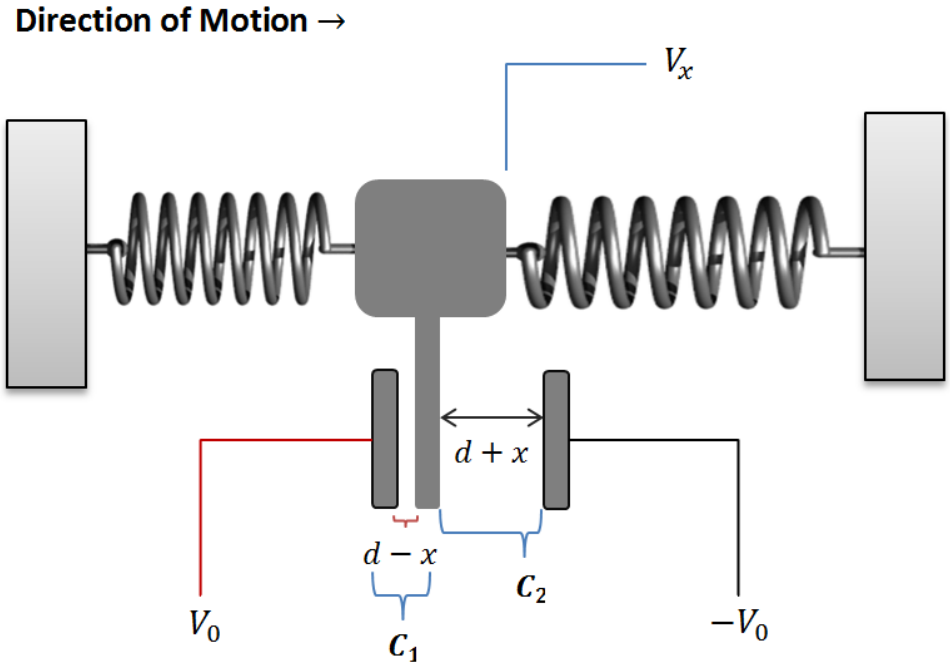



Figure 5: Fully labeled diagram of a simplified MEMS accelerometer.

In Figure 5, the conservation of charge can be written as:

indicator of force applied to the mass, which will ultimately bestow information about the acceleration of the system. Utilizing the equations for capacitance, we have:

$$\frac{V_0 - V_x}{d - x} + \frac{-V_0 - V_x}{d + x} = 0$$

And finally, we can solve the above algebraic equation for the displacement of the mass:

$$x = \frac{V_x d}{V_0}$$

At this point, it is easy to see how we can relate the knowns in the system to find acceleration:

$$F = ma = -kx \Rightarrow a = \frac{-kV_x d}{V_0 m}$$

This simple result endows us with immense power. The above equation states that with prescribed values of spring constants, movable mass, capacitor equilibrium separation distance, and an applied voltage, we can determine the acceleration of the system just by reading a voltage. This is an impressive conclusion despite the complex nature of the problem. It is a basis for understanding the capacitive MEMS accelerometers that are used and still built today, and it is also the foundation for the sensor used in the experiment below, which utilizes a capacitive MEMS accelerometer to measure forces and distance traveled in a commercial elevator.

[For more in-depth analysis and investigation of MEMS accelerometers see here, here, and here].



## ACCELEROMETER ON AN ELEVATOR

On an elevator, weight fluctuates based on the acceleration of the vessel. If an elevator is accelerating upward, then a mass's weight will

knowledge of how a MEMS accelerometer works, I decided to verify Newton's 2nd law and the theory of gravitation (on earth's surface, at least). I used an ADXL345 (datasheet here, get one here), and interfaced with an Arduino and an iPhone using the HM-10 module and Bluetooth. That way, I was able to telemeter the acceleration data at roughly 17 samples/second in real time. The ADXL345 is capable of 100 samples/sec, so there was no worry of accuracy in time. However, the accuracy of the sensor itself proved to be less than ideal, along with positioning inaccuracies, as well as other shortcomings. These, however, will be discussed during the analysis section. The goal for this project was to analyze the maximum accelerations of a specific elevator (located in the Marshak Science Building on the campus of the City College of New York, see Figure 7), and examine the accuracy of numerical integration of a MEMS accelerometer to approximate distance traveled.

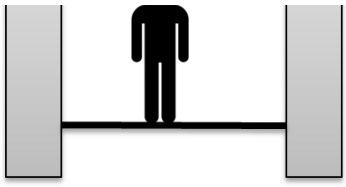
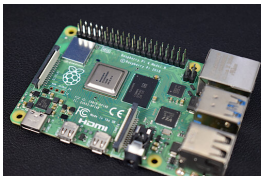


Figure 6: An elevator moving upward will increase a person's weight, while the opposite is true for a downward moving elevator.

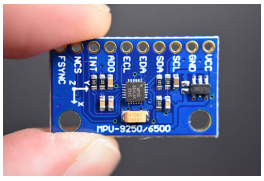
***Products from our shop useful for replicating this experiment:***



Raspberry Pi 4 Model B Computer

$55.00

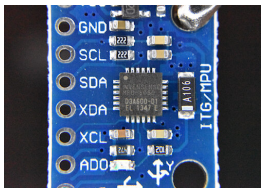
Purchase




MPU9250 Inertial Measurement Unit (IMU)

$13.00

Purchase



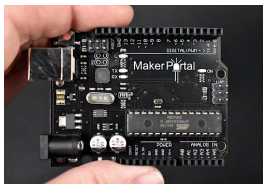

MPU6050 6-DoF Accelerometer and Gyroscope (Arduino and Raspberry Pi Compatible)

$8.00

Quantity:

1

Purchase




Maker Portal Arduino Uno Rev3 Board

$13.00

Quantity:

1

Purchase

Maker Portal

## Methodology

The procedure for this experiment is simple:

1. Take continuous measurements of acceleration on an elevator
2. Ensure that multiple distances are measured - this means taking multiple rides in the elevator to different floors
3. Normalize the accelerometer data to account for gravity
4. Integrate the acceleration data over time to approximate instantaneous velocity
5. Integrate velocity data and sum to approximate displacement
6. Find maximum acceleration to investigate maximum forces

An example acceleration plot after application of a simple running mean filter is shown in Figure 8.

Figure 7: Marshak Science building where the acceleration measurements were made on several of the elevators. Image courtesy of www.rsdeng.com

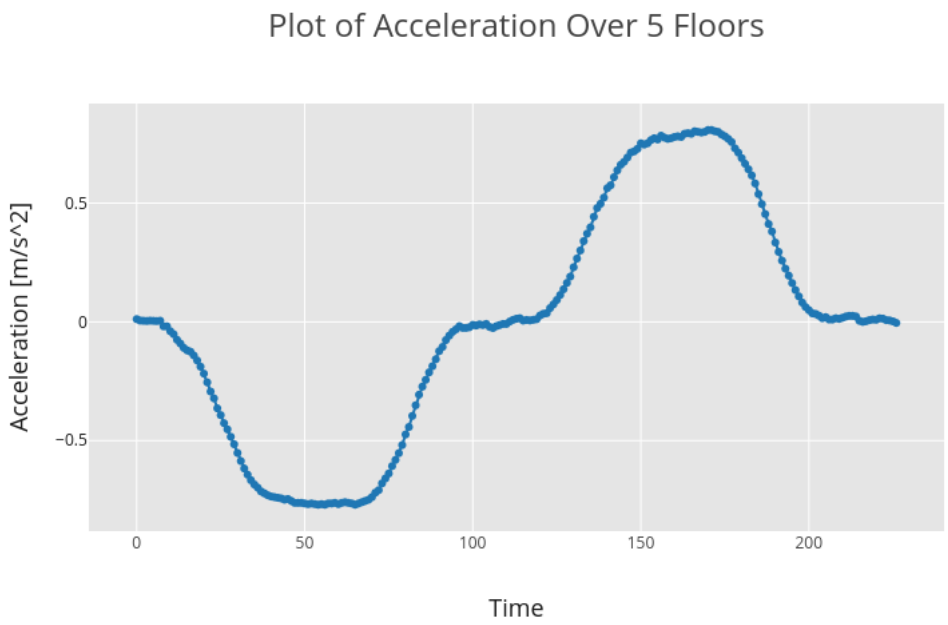



*FIGURE 8: PLOT OF ACCELERATION FOR ONE INSTANCE WHERE 5 FLOORS WERE TRAVERSED BY AN ELEVATOR. IT IS IMPORTANT TO REALIZE THAT THE ELEVATOR ACCELERATES IN ONE DIRECTION TO MOVE UPWARD OR DOWNWARD, AND DECELERATES TO STOP AND LET PEOPLE DISEMBARK. THE ABOVE DATA WAS SMOOTHED TO ELIMINATE ERRONEOUS DATA POINTS. THE ACCELEROMETER DATA WAS ALSO NORMALIZED BY GRAVITY TO ACCOUNT FOR ONLY THE ACCELERATION OF THE BODY IN ITS OWN FRAME OF REFERENCE.*

After approximating the influence of gravity from the steady-state of the sensor, I numerically integrated the discrete data using the following trapezoidal method:

$$v(t) = \int_b^c a(t)\,dt$$

$$v(t) = \int_t^{t+1} a(t)\,dt \approx \frac{a(t_i) + a(t_{i+1})}{2}\Delta t_{i+1}$$

After carrying out the above integration, we now have instantaneous velocity data and with instantaneous velocity, displacement can easily be calculated. At this point, the velocity can be integrated via the same procedure, or the displacement can be obtained from the sum of the velocities multiplied by the time spacing between them - either one should be a good approximation for this particular case.

$$x(t) \approx \sum v(t) \cdot \Delta t$$

Finally, we arrive at a displacement. I carried out several of these calculations of displacement for multiple traverses between building floors in the Marshak Science Building (Figure 7), and I arrived at a plot of floors traversed vs. distance calculated, as shown in Figure 9.

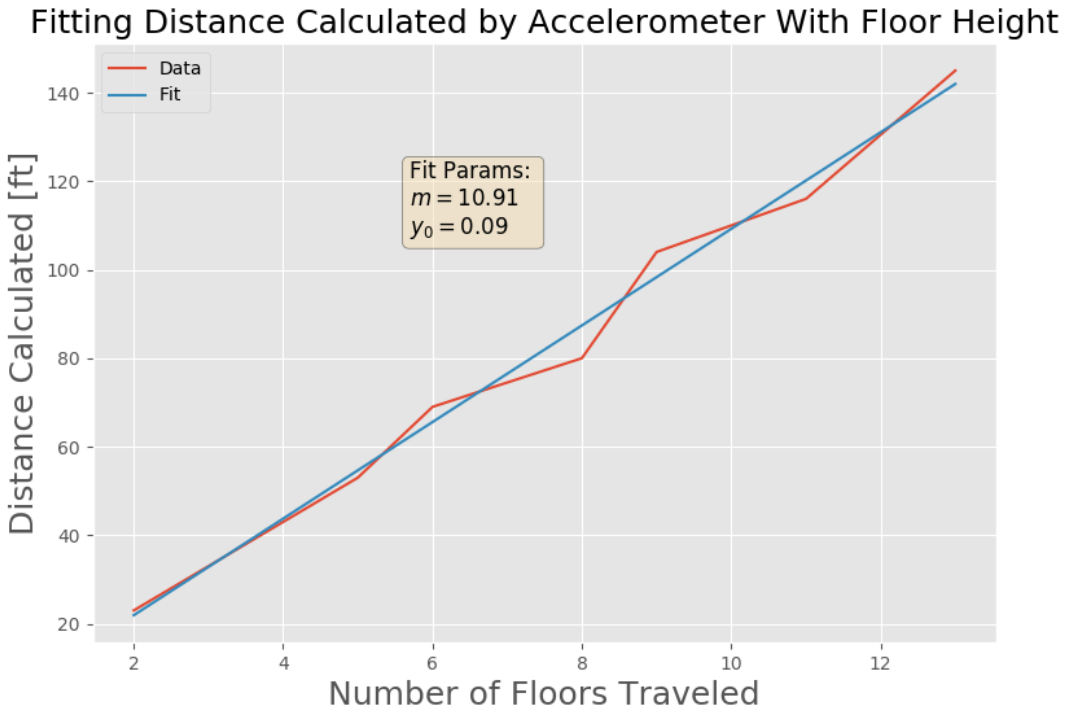



Figure 9: Plot of number of floors traversed vs. distance calculated. This plot contains the raw approximations with the linear fit. It can be seen that the approximate height of each floor is about 11 ft. This makes perfect sense considering the height of the building (166 ft). There are 13 regular floors, plug roughly two others to get to the roof. Therefore, if the building is approximated to be 15 stories, then 10.91 x 15 = 163.7 ft. This appears to be a decent approximation, considering the level of accuracy of the sensor.

The fit shown in Figure 9 concludes that the height of each floor is approximately 10.9 ft. This appears to be a fairly accurate prediction of the height of this particular building, due to the prior knowledge of its

Maker Portal

within 2% of the actual building height.

Additionally, I was interested in the forces exerted during a trip on the elevator. The observed maximum acceleration was on average 1.1 m/s$^2$ higher than gravity, and ocassionally approached 2.6 m/s$^2$ above gravity. This indicates that on any given elevator ride (in this particular elevator), a person's weight can shift by 10%, and can ocassionally flutuate by 25%! As an example, an 175 lbs person who steps on one of the elevators will weigh 194 lbs on the way up, and 159 lbs on the way down. If it is unclear why, I included the simple weight balance with the added acceleration below:

$$\% \text{ Change in Weight} = \frac{W - W^*}{W} = \frac{mg - mg^*}{mg} = 1 - \frac{g*}{g}$$

$$1 - \frac{g*}{g} = 1 - \frac{g + a'}{g} = -\frac{a'}{g}$$

Where $a'$ is the fluctuating acceleration after gravity is subtracted. The negative sign states that if the fluctuating acceleration is greater than zero, then the body is traveling in the same direction as gravity, resulting in a net loss of weight; and for a negative fluctuating acceleration, the body is traveling against gravity, resulting in a net gain of weight. This is a fascinating study, and one that I'm sure manufacturers have to account for in fabrication of elevators and other transportation machinery. The experiments using an Arduino board, Bluetooth communication, Python for post-processing, and some elementary physics proved to be highly accurate and elucidatory for a classic physics and engineering problem that many of us take for granted.

## PYTHON CODE

### POLYFIT FOR FLOOR HEIGHT LINEAR REGRESSION

```
# polyfitting for floor height
#
#
import matplotlib as mpl
mpl.use('tkagg')
import numpy as np
import matplotlib.pyplot as plt
##plt.style.use('ggplot')
plt.style.use('ggplot')



bldg_floors = [2,5,6,8,9,11,13]
bldg_heights = [23,53,69,80,104,116,145]

poly_fit = np.polyfit(bldg_floors,bldg_heights,1)
poly_map = np.poly1d(poly_fit)
```

Maker Portal

```
#plt.figure(figsize=(9,6),dpi=100)

ax.plot(bldg_floors,bldg_heights,label='Data')
plt.plot(bldg_floors,poly_map(bldg_floors),label='Fit')
plt.title('Fitting Distance Calculated by Accelerometer With Floor Height'
plt.xlabel('Number of Floors Traveled',fontsize=18)
plt.ylabel('Distance Calculated [ft]',fontsize=18)
plt.legend()


txtstr_fit = 'Fit Params: \n$m = %0.2f$\n$y_0 = %0.2f$'%(poly_fit[0],poly_
props = dict(boxstyle = 'round',facecolor='wheat',alpha=0.5,ec='k')
plt.text(0.35,0.8,txtstr_fit,transform=ax.transAxes,fontsize=12,\
         verticalalignment='top',bbox=props)

plt.show()
plt.savefig('bldg_fitting.png',transparent=True,dpi=400)
```

## IMPORTING, FILTERING, AND PLOTTING CSV ACCELEROMETER DATA

```
import plotly.plotly as py
import plotly.graph_objs as go
import csv
import numpy as np
from scipy import integrate
from datetime import datetime

dat_val = np.array([])
time_vec = np.array([])
file_name = 'G_to_13th_fl'
with open(file_name+'.csv') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        (t_hr,t_min) = (row['Date']).split('+')
        t_sec = datetime.strptime(t_hr,'%Y-%m-%d %H:%M:%S ')
        time_vec = np.append(time_vec,t_sec)
        dat_val = np.append(dat_val,float(row['Data Reading']))
##        time_mon.append(int(row['Interval_End_Time'][0]))
##        time_vec.append(int(t_hr))

n_start =0
n_end = 404#len(dat_val)
interv_to_int = (np.linspace(n_start,n_end,n_end-n_start+1,dtype=int))
interv_to_int = interv_to_int[0:len(interv_to_int)-2]

dat_val = dat_val[interv_to_int]-np.mean(dat_val[0:15])

conv_size = 10

dat_val = np.convolve(dat_val,np.ones(conv_size),'same')/conv_size


##dat_val = dat_val+9.95
time_vec = time_vec[interv_to_int]
delta_x = np.mean(np.diff(time_vec))
dx = delta_x.total_seconds()
vel = integrate.cumtrapz(dat_val)*dx
```

```
##time_vec = datetime.strptime(time_vec,'%Y-%m-%d %H:%M:%S')
time_vec = np.linspace(0,len(time_vec),len(time_vec)+1)
time_vec = time_vec[0:len(time_vec)-1]
trace = go.Scatter(
    x = time_vec,
    y = dat_val,
    name = 'Plot of Ground to 13th Floor Acceleration in y-direction',
    mode='lines+markers'
    )

layout = go.Layout(
    paper_bgcolor = 'rgb(255,255,255)',
    plot_bgcolor='rgb(229,229,229)',
    width = 800,
    title = 'Plot of Ground to 13th Floor Acceleration in y-direction',
    titlefont = dict(
        size=24
        ),
    xaxis = dict(
        title='Time',
        titlefont=dict(
            size = 18,
            color='rgb(0.1,0.1,0.1)'
            ),
        gridcolor='rgb(255,255,255)',
        zeroline = False,
##        range=[min(dat_val),max(dat_val)]
        ),
    yaxis = dict(
        title='Acceleration [m/s^2]',
        titlefont=dict(
            size = 18,
            color='rgb(0.1,0.1,0.1)'
            ),
        tickfont = dict(
            color = 'rgb(0.1,0.1,0.1)'
            ),
        gridcolor='rgb(255,255,255)',
        zeroline = False
        )
    )


fig = go.Figure(data=[trace],layout=layout)
plot_url = py.plot(fig,filename=file_name,auto_open=False)
```

**Citation for This Page:**

Hrisko, J. (2017). Introduction To Capacitive MEMS Accelerometers and A Case Study On An Elevator. Maker Portal. https://makersportal.com/blog/2017/9/25/accelerometer-on-an-elevator

*See More in Raspberry Pi and Inertial Measurement Units:*

Maker Portal

Calibration of a Magnetometer with Raspberry Pi

Jan 11, 2021

Gyroscope and Accelerometer Calibration with Raspberry Pi

Jan 3, 2021

Calibration of an Inertial Measurement Unit (IMU) with Raspberry Pi - Part I

Dec 29, 2020

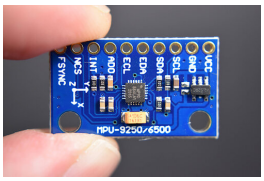
Accelerometer, Gyroscope, and Magnetometer Analysis with Raspberry Pi Part I: Basic Readings

Nov 15, 2019


MPU6050 Arduino High-Frequency Accelerometer and Gyroscope Data Saver

Aug 31, 2019

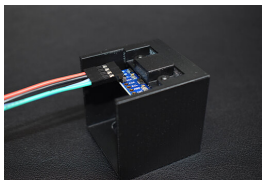
*Related Products:*




MPU9250 Inertial Measurement Unit (IMU)

$13.00

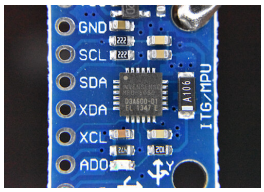
*Purchase*



IMU Calibration Block Kit with MPU9250

$25.00

*Purchase*



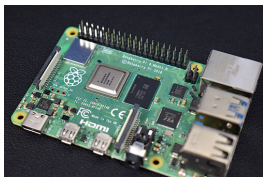

MPU6050 6-DoF Accelerometer and Gyroscope (Arduino and Raspberry Pi Compatible)

$8.00

Quantity:

1

*Purchase*



Raspberry Pi 4 Model B Computer

$55.00

*Purchase*

**Delgado Johan** • a year ago
Hello, this is very helpful. Can I ask why did you use the mean of your DT for your integrating? Why not the "real" DT?

PREVIOUS

Geospatial Analysis Using QGIS and Open-Source Data

*October 30, 2017*

NEXT

Arduino - Sending A String Over Bluetooth Using The HM-10

*October 15, 2017*