

Connect to database

```
import serial
import time
from datetime import datetime
import os
import pandas as pd
from firebase_admin import db
import json
import firebase_admin
from firebase_admin import credentials
import matplotlib.pyplot as plt
```

```
databaseURL = "https://gyroscope-ca247-default-rtdb.asia-southeast1.firebaseio.com/"

cred = credentials.Certificate("key.json")
firebase_admin.initialize_app(cred,
                              {'databaseURL':databaseURL})
```

```
<firebase_admin.App at 0x13e9b083a50>
```

Write to database

This code run continuously for 37 minutes and 47 seconds. The phenomena I want to capture with the Gyroscope sensor is an earth quake simulation (similar to that of the last credit task). During the 37 minutes, there are 3 earthquakes, at minute 10, at minute 20 and minute 30.

A bit of context for the code block below. It reads the data (x,y,z) from the Arduino via serial, then save it into a csv and json file along with the timestamp.

```

# Function to get the current time
def timestamp():
    return datetime.now().strftime('%Y%m%d%H%M%S')

# Serial port and saving csv, json file in desire destination
ser = serial.Serial('COM4', 9600)
csv_file = os.path.join(r'C:\Users\tomde\OneDrive\Documents\Deakin\Deakin-Data-Science\T1Y2\')

try:
    while True:
        # Check if data is waiting in serial buffer
        if ser.in_waiting > 0:
            data = ser.readline().decode('utf-8').strip() # read data from serial port and d
            formatted_data = f"{timestamp()}, {data}"

            # Add data to csv file
            with open(csv_file, 'a') as file:
                file.write(formatted_data + '\n')

            # Add data to json file
            df = pd.read_csv("data.csv", header=None, names=['Timestamp', 'x', 'y', 'z']) # r
            json_data = df.to_json(orient='index', date_format='iso') # convert dataframe to
            with open('data.json', 'w') as file2:
                file2.write(json_data)

            # write to database
            ref = db.reference("/") # reference to the root of the DB
            with open("data.json", "r") as f:
                file_contents = json.load(f)
            ref.set(file_contents)

            # print(f"{formatted_data}")

        time.sleep(1)

except KeyboardInterrupt:
    print("Forced stop by user.")

finally:
    ser.close()
    print("Serial port closed.")

```

Forced stop by user.
Serial port closed.

Query the Firebase data

```
firebase_data = ref.get()
display(firebase_data)
display(type(firebase_data))
```

```
[{'Timestamp': 20240811140905, 'x': -0.01, 'y': 0.0, 'z': 1.02},
 {'Timestamp': 20240811140906, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140907, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140909, 'x': 0.0, 'y': 0.0, 'z': 1.02},
 {'Timestamp': 20240811140910, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140911, 'x': -0.01, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140912, 'x': 0.0, 'y': -0.01, 'z': 1.03},
 {'Timestamp': 20240811140913, 'x': 0.0, 'y': 0.01, 'z': 1.02},
 {'Timestamp': 20240811140914, 'x': 0.0, 'y': 0.01, 'z': 1.03},
 {'Timestamp': 20240811140915, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140917, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140918, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140919, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140920, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140921, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140922, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140924, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140925, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140926, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140927, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140928, 'x': 0.0, 'y': 0.0, 'z': 1.02},
 {'Timestamp': 20240811140929, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140930, 'x': 0.0, 'y': 0.0, 'z': 1.02},
 {'Timestamp': 20240811140932, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140933, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140934, 'x': 0.0, 'y': 0.0, 'z': 1.02},
 {'Timestamp': 20240811140935, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140936, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140937, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140938, 'x': 0.0, 'y': 0.0, 'z': 1.03},
 {'Timestamp': 20240811140940, 'x': 0.0, 'y': 0.0, 'z': 1.03},
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]


```
{'Timestamp': 20240811142042, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142044, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142045, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142046, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142047, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142048, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142050, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142051, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142052, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142053, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142054, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142056, 'x': -0.01, 'y': -0.91, 'z': 0.45},
{'Timestamp': 20240811142057, 'x': -0.01, 'y': -0.9, 'z': 0.47},
{'Timestamp': 20240811142058, 'x': -0.01, 'y': -0.82, 'z': 0.6},
{'Timestamp': 20240811142059, 'x': -0.07, 'y': 0.24, 'z': 1.11},
{'Timestamp': 20240811142101, 'x': 0.0, 'y': 0.95, 'z': 0.32},
{'Timestamp': 20240811142102, 'x': -0.36, 'y': 0.36, 'z': 0.82},
{'Timestamp': 20240811142103, 'x': -0.68, 'y': 0.11, 'z': 0.74},
{'Timestamp': 20240811142104, 'x': -0.71, 'y': 0.22, 'z': 0.75},
{'Timestamp': 20240811142106, 'x': 0.62, 'y': -0.26, 'z': 0.66},
{'Timestamp': 20240811142107, 'x': 0.52, 'y': -0.17, 'z': 0.8},
{'Timestamp': 20240811142108, 'x': 0.1, 'y': 0.13, 'z': 0.99},
{'Timestamp': 20240811142109, 'x': -0.2, 'y': -0.15, 'z': 1.15},
{'Timestamp': 20240811142110, 'x': 0.14, 'y': 0.02, 'z': 1.05},
{'Timestamp': 20240811142111, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142113, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142114, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142115, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142116, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142117, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142119, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142120, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142121, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142122, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142123, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142124, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142126, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142127, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142128, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142129, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142130, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142132, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142133, 'x': 0.0, 'y': 0.0, 'z': 1.03},
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
{'Timestamp': 20240811142833, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142835, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142836, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142837, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142838, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142839, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142841, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142842, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142843, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142844, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142846, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142847, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142848, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142849, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142851, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142852, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142853, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142854, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142855, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142857, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142858, 'x': 0.0, 'y': 0.0, 'z': 1.03},
{'Timestamp': 20240811142859, 'x': 0.0, 'y': 0.0, 'z': 1.02},
{'Timestamp': 20240811142900, 'x': 0.0, 'y': 0.0, 'z': 1.02},
...]
```

list

Since I already have a csv file saved when uploading data to the database, I will use it now.

```
df1 = pd.read_csv("data.csv", header=None, names=['Timestamp', 'x', 'y', 'z'])
display(df1.head())
display(df1.info())
```

	Timestamp	x	y	z
0	20240811140905	-0.01	0.0	1.02
1	20240811140906	0.00	0.0	1.03
2	20240811140907	-0.00	0.0	1.03
3	20240811140909	-0.00	0.0	1.02
4	20240811140910	-0.00	0.0	1.03

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1844 entries, 0 to 1843
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Timestamp    1844 non-null   int64
1   x             1844 non-null   float64
2   y             1844 non-null   float64
3   z             1844 non-null   float64
dtypes: float64(3), int64(1)
memory usage: 57.8 KB
```

None

```
# Convert `Timestamp` column to time dtype
df['Timestamp'] = pd.to_datetime(df['Timestamp'], format='%Y%m%d%H%M%S')
df.head()
```

	Timestamp	x	y	z
0	2024-08-11 14:09:05	-0.01	0.0	1.02
1	2024-08-11 14:09:06	0.00	0.0	1.03
2	2024-08-11 14:09:07	-0.00	0.0	1.03
3	2024-08-11 14:09:09	-0.00	0.0	1.02
4	2024-08-11 14:09:10	-0.00	0.0	1.03

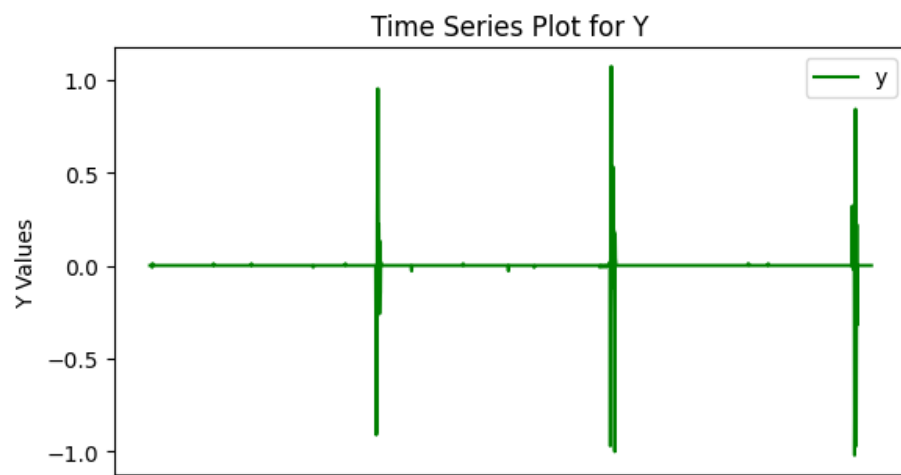
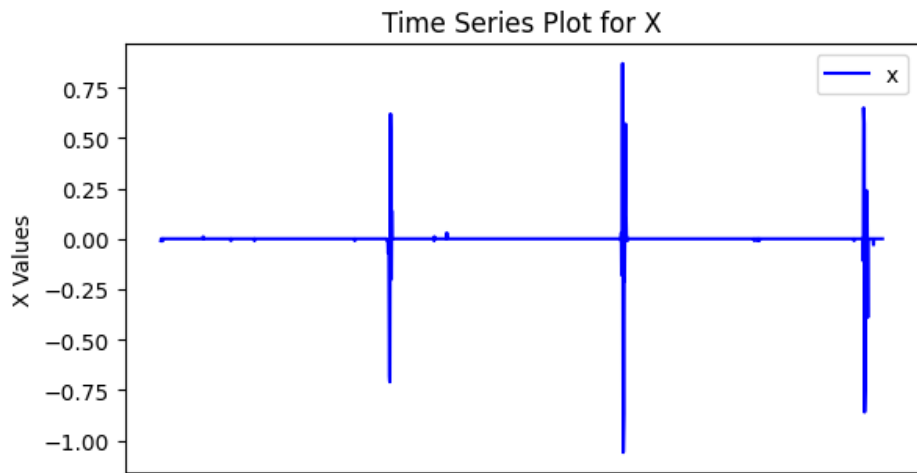
```
# Subplots
df.set_index('Timestamp', inplace=True)
fig, axs = plt.subplots(3, 1, figsize=(6, 10), sharex=True)

# Plot x
axs[0].plot(df.index, df['x'], label='x', color='b')
axs[0].set_title('Time Series Plot for X')
axs[0].set_ylabel('X Values')
axs[0].legend()

# Plot y
axs[1].plot(df.index, df['y'], label='y', color='g')
axs[1].set_title('Time Series Plot for Y')
axs[1].set_ylabel('Y Values')
axs[1].legend()
```

```
# Plot z
axs[2].plot(df.index, df['z'], label='z', color='r')
axs[2].set_title('Time Series Plot for Z')
axs[2].set_xlabel('Date')
axs[2].set_ylabel('Z Values')
axs[2].legend()

# Display the plot
plt.tight_layout()
plt.show()
```



X, Y, Z are rotation on each axis. The rotation of all three axis are mostly static, except at minute 10, 20 and 30, in which I shake the sensor to simulate earthquake. The pattern of earthquake is 10 minute interval, To be specific of the time, we need further analysis.

```
df.describe()
```

	x	y	z
count	1844.000000	1844.000000	1844.000000
mean	-0.001377	-0.000960	1.019555
std	0.062594	0.076862	0.071252
min	-1.060000	-1.020000	-0.160000
25%	-0.000000	0.000000	1.020000
50%	-0.000000	0.000000	1.030000
75%	0.000000	0.000000	1.030000
max	0.870000	1.070000	1.150000

50% percentile is the median value for the 3 rotations. I will filter plus minus 0.2 the median for each rotations.

```
median = df.describe().loc['50%']
median
```

```
x    -0.00
y     0.00
z     1.03
Name: 50%, dtype: float64
```

```
df[
    (df['x'] > (median.iloc[0] + 0.2)) | (df['x'] < (median.iloc[0] - 0.2)) |
    (df['y'] > (median.iloc[1] + 0.2)) | (df['y'] < (median.iloc[1] - 0.2)) |
    (df['z'] > (median.iloc[2] + 0.2)) | (df['z'] < (median.iloc[2] - 0.2))
]
```

	x	y	z
Timestamp			
2024-08-11 14:20:56	-0.01	-0.91	0.45
2024-08-11 14:20:57	-0.01	-0.90	0.47
2024-08-11 14:20:58	-0.01	-0.82	0.60

	x	y	z
Timestamp			
2024-08-11 14:20:59	-0.07	0.24	1.11
2024-08-11 14:21:01	0.00	0.95	0.32
2024-08-11 14:21:02	-0.36	0.36	0.82
2024-08-11 14:21:03	-0.68	0.11	0.74
2024-08-11 14:21:04	-0.71	0.22	0.75
2024-08-11 14:21:06	0.62	-0.26	0.66
2024-08-11 14:21:07	0.52	-0.17	0.80
2024-08-11 14:33:11	0.03	-0.97	-0.14
2024-08-11 14:33:12	-0.18	0.31	0.78
2024-08-11 14:33:14	-0.02	1.07	0.08
2024-08-11 14:33:15	0.87	-0.12	0.18
2024-08-11 14:33:16	-0.21	0.08	0.99
2024-08-11 14:33:17	-1.06	0.08	0.13
2024-08-11 14:33:19	-0.87	0.53	0.38
2024-08-11 14:33:20	-0.20	0.23	1.05
2024-08-11 14:33:21	-0.21	0.10	0.90
2024-08-11 14:33:24	0.57	-0.25	0.32
2024-08-11 14:33:25	0.05	-1.00	0.50
2024-08-11 14:45:51	-0.11	0.32	0.85
2024-08-11 14:45:52	0.65	-0.01	0.80
2024-08-11 14:45:54	0.57	-0.02	1.07
2024-08-11 14:45:55	-0.86	0.26	0.67
2024-08-11 14:45:57	-0.72	0.01	0.80
2024-08-11 14:45:58	-0.12	-1.02	-0.16
2024-08-11 14:45:59	-0.04	-0.69	1.11
2024-08-11 14:46:01	0.24	0.84	0.39
2024-08-11 14:46:02	-0.00	0.35	0.96
2024-08-11 14:46:03	0.12	-0.97	0.31
2024-08-11 14:46:05	-0.39	0.22	0.92
2024-08-11 14:46:06	-0.02	-0.32	1.03

The first movement is from 14:20:56 to 14:21:07. The second is 14:33:11 to 14:33:25. The last is 14:45:51 to 14:46:06. The interval is not quite exact 10 minutes, but around that.