

Student name: Aditya Suhag

Student ID: 224001686

SIT225: Data Capture Technologies

Activity 8.1: Using smartphone to capture sensor data

The **Arduino IoT Remote** phone application lets you control and monitor all of your dashboards in the Arduino Cloud. With the app, you can also access your phone's internal sensors such as GPS data, light sensor, IMU and more (depending on what phone you have).

The phone's sensor data is automatically stored in Cloud variables, which you can also synchronize with other Things such as custom thing in Python board. This means your phone can become a part of your IoT system, acting as another node in your network.

In this activity, you will enable your smartphone to work as a custom device (like an Arduino board) and connect to your smartphone sensors such as accelerometers and GPS and streaming data to Arduino IoT Cloud dashboard.

Hardware Required

Your smartphone – compatible Android or iPhone

NOTE: *The IoT Remote app requires iOS 12.4 or later for iOS the version. If you are using Android, version 8.0 or later is required. Make sure the iOS or Android version on your device is up to date before downloading the app.*

Software Required

Android / iOS smart phone.

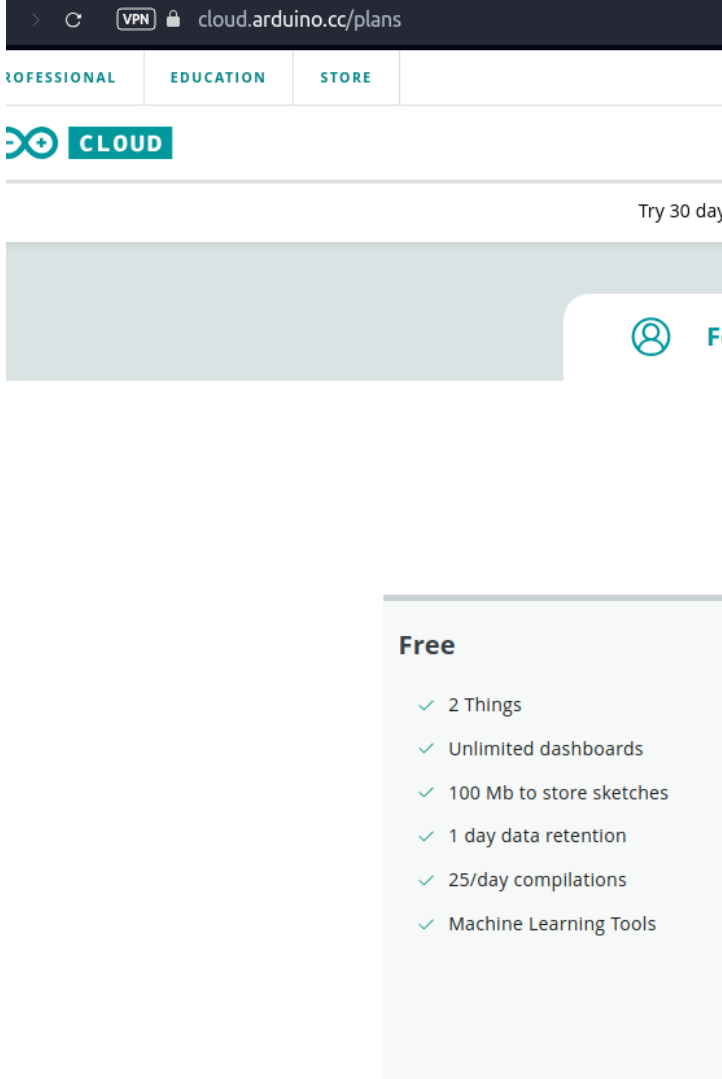
Arduino account

Arduino IoT Remote App (App Store or Google Play)

Python 3 (for custom Python Thing)

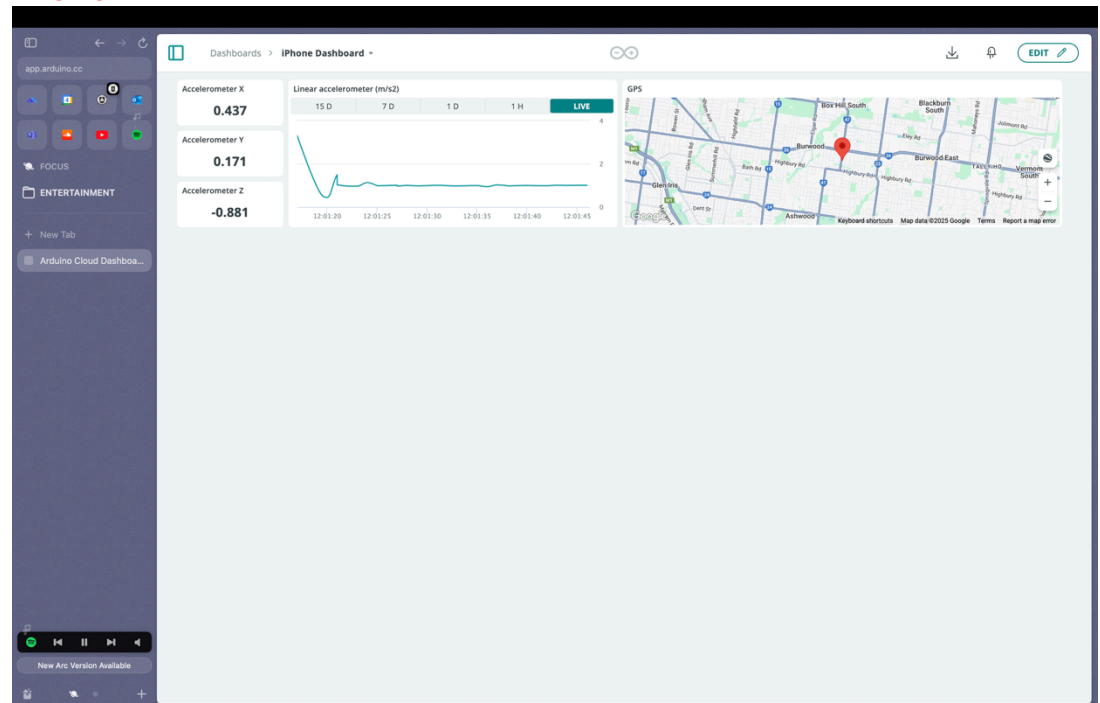
Steps

Step	Action
1	<p>Install App:</p> <p>To use the Arduino IoT Remote app, visit Google Play / App Store and search for "Arduino IoT Remote".</p> <p>After installing the app, you will need to log in to your Arduino account.</p> <p>After you login, you will discover all your dashboards (if you have any), in the main menu. Based on the app version, home screen may vary. There will be 3 tabs at the bottom – Dashboards, Devices and Activity. You can follow the tutorial (https://docs.arduino.cc/arduino-cloud/iot-remote-app/getting-started).</p>
2	<p>Add device:</p> <p>Tap into the Devices tab. You will be able to create a new device. Alternatively, you can your profile (top right corner), in the settings section, you will see "Phone as device" which you can turn ON if it is OFF. There, you can select sensors in your smartphone such as accelerometer linear, accelerometer x/y/z and GPS among others.</p> <p>Note: A free account is enough for this experiment. If you are asked to upgrade your account, you can remove all other Things from your Arduino IoT Cloud account since the Free account allows at most 2 Things to configure, see below image.</p>

	 <p>The add device wizard will allow you to setup your sensor and also create a dashboard which you can see in your smartphone app. If you login to Arduino IoT Cloud in web browser, you can see the dashboard for your smartphone is already created.</p>
3	<p>Keep your smartphone screen ON for a while: Keep data coming through your smartphone for 10-15 minutes. During this time, keep moving your smartphone in a pattern so the accelerometer data can be analysed to discover the pattern.</p> <p>You can download data from the Arduino Cloud dashboard page by clicking on a download icon at top right corner which shows – Download historic data. A data download link will be sent to your account email from there you can download data.</p>

Question: Take a screenshot of your Arduino Cloud Dashboard where smartphone data is streaming and paste it here.

Answer:



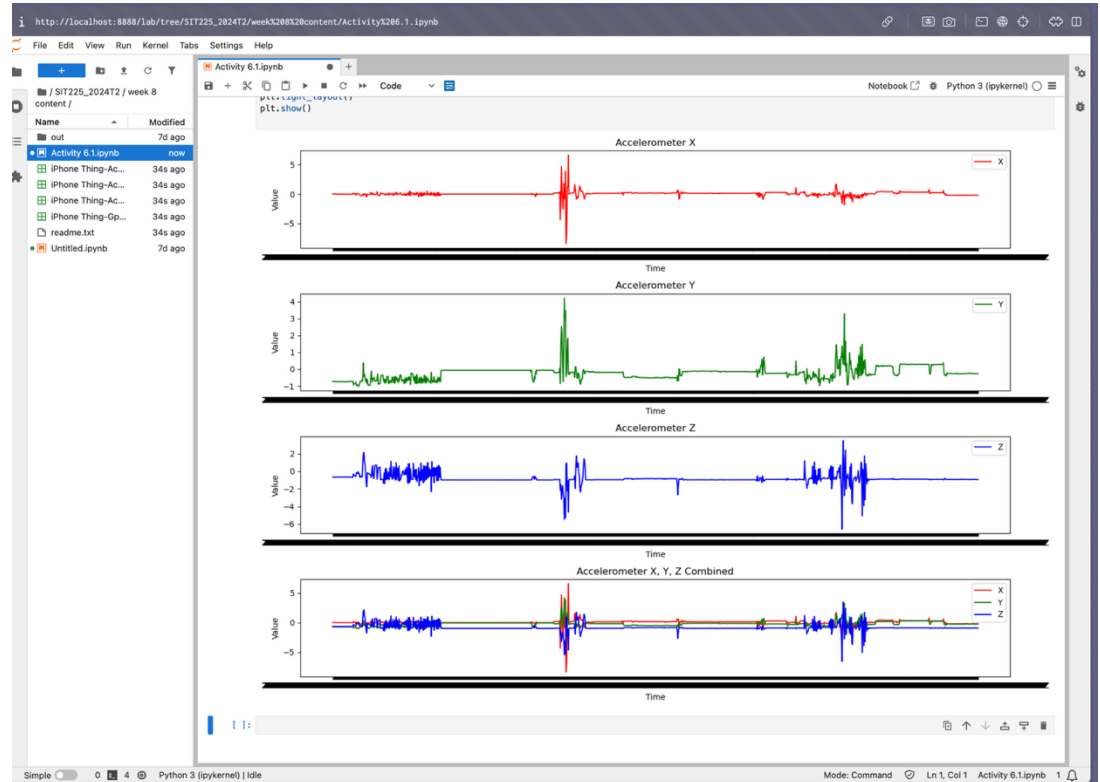
4

Plot accelerometer data:

The zipped data file you downloaded from the cloud contains separate files per variable including accelerometer_linear, accelerometer_x, accelerometer_y, accelerometer_z and Gps. Each file has 2 columns – time and value.

Question: Open Jupyter Notebook by using command line, go to the data folder and write command (`$ jupyter lab`). Using Pandas, read CSV file and fetch the data column for accelerometer_x and plot it using Python plotting library (matplotlib or any other convenient for you). Repeat the plotting process for accelerometer y and z to have 3 separate graphs. Now create a fourth graph with all 3 variables x, y and z. Screenshot the 4 graphs and paste here.

Answer:



5

Question: Analyse accelerometer variables to find any repeating pattern. Remember that you were repeatedly moving your phone in a single pattern which should be manifested in the graphs. Justify your answer.

Answer: Because of these irregular spikes and irregular fluctuations, I could not find a repeating pattern in these graphs for X, Y, and Z in the accelerometer. If I had been moving my phone steady in rhythm, then smoother wave-like cycles would be evident on the graphs; yet here, the signals seem more chaotic and uneven. It makes a sort of sense because while recording, I had been cooking and playing with my dog, which involved sudden and unpredictable movements. Hence, the data mainly caters to these irregular movements of mine rather than some sort of rhythm.

Activity 8.2: Receive smartphone sensor data from Python script

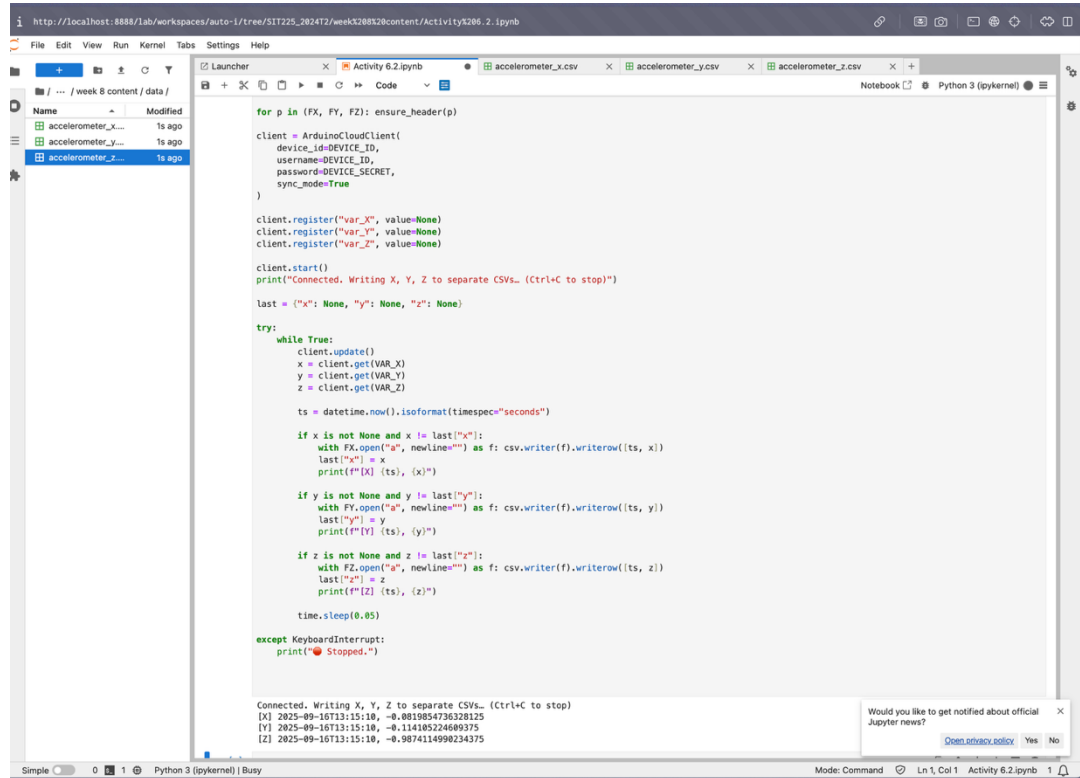
You can connect anything to Arduino Cloud including a wide range of compatible Arduino boards such as Arduino Nano 33 IoT or a third-party device that speaks Python. In activity 3.2, you have configured custom Python board and created a cloud variable that was synced to your Arduino Thing such as DHT22 sensor variables.

In this activity, you will need to synchronise smartphone's accelerometer x, y and z variables to Python script. If you can recall, you have already done a similar function in Activity 3.2.

Steps:

Step	Action
1	Configure Python board in Arduino Cloud and create a Thing where define 3 variables at a time and sync to corresponding accelerometer variable of smartphone Thing.
2	Write Python script to keep listening to data from the 3 variables to come through. You may need to create 3 call-back functions – a single function per variable (x, y and z).
3	Question: Keep storing each variable data in a separate file. Append each value with a timestamp so each data reading forms a comma separated line - <timestamp>, <data-value>. New data is written in a separate line. Keep storing them in a CSV file, where there will be 3 separate files. Screenshot your Python script here and screenshot the files opened side-by-side you have created and paste it here.

Answer:



```
for p in (FX, FY, FZ): ensure_header(p)

client = ArduinoCloudClient(
    device_id=DEVICE_ID,
    username=DEVICE_ID,
    password=DEVICE_SECRET,
    sync_mode=True
)

client.register("var_X", value=None)
client.register("var_Y", value=None)
client.register("var_Z", value=None)

client.start()
print("Connected. Writing X, Y, Z to separate CSVs... (Ctrl+C to stop)")

last = {"X": None, "Y": None, "Z": None}

try:
    while True:
        client.update()
        x = client.get(VAR_X)
        y = client.get(VAR_Y)
        z = client.get(VAR_Z)

        ts = datetime.now().isoformat(timespec="seconds")

        if x is not None and x != last["X"]:
            with FX.open("a", newline="") as f: csv.writer(f).writerow([ts, x])
            last["X"] = x
            print(f"[X] {ts}, {x}")

        if y is not None and y != last["Y"]:
            with FY.open("a", newline="") as f: csv.writer(f).writerow([ts, y])
            last["Y"] = y
            print(f"[Y] {ts}, {y}")

        if z is not None and z != last["Z"]:
            with FZ.open("a", newline="") as f: csv.writer(f).writerow([ts, z])
            last["Z"] = z
            print(f"[Z] {ts}, {z}")

        time.sleep(0.05)
except KeyboardInterrupt:
    print("Stopped.")
```

Connected. Writing X, Y, Z to separate CSVs... (Ctrl+C to stop)
[X] 2025-09-10T13:15:10, -0.0019054736328125
[Y] 2025-09-10T13:15:10, -0.114105224600375
[Z] 2025-09-10T13:15:10, -0.9874114990234375

4

Question: Now manage 3 variable data so they can be stored in a single CSV file where each line consists of comma separated sensor values with a timestamp - <timestamp>, <x>, <y>, <z>. Store data once you gather 3 variables and repeat the process. Screenshot your Python script here and screenshot the file you have created opened and paste it here.

Answer:

The screenshot displays a JupyterLab environment with two main panels. The top panel shows a Python script in a code editor, and the bottom panel shows a table of data in a file viewer.

Python Script (Pass.env):

```
[!]: # creating only one file instead of creating 3 different csv files

import os, csv, time
from datetime import datetime
from pathlib import Path
from dotenv import load_dotenv
from arduino_iot_cloud import ArduinoIoTCloudClient

# 1) Load Python device creds (Alaya)
load_dotenv('Pass.env') # change to ".env" if that's your filename
DEVICE_ID = os.getenv("DEVICE_ID")
DEVICE_SECRET = os.getenv("DEVICE_SECRET")
if not DEVICE_ID or not DEVICE_SECRET:
    raise SystemExit("Missing DEVICE_ID/DEVICE_SECRET in env file")

# 2) PhoneBridge variable names (Linked to iPhone Thing)
VAR_X = "var_x"
VAR_Y = "var_y"
VAR_Z = "var_z"

# 3) Output file
DATA_DIR = Path("data"); DATA_DIR.mkdir(exist_ok=True)
OUT = DATA_DIR / "accelerometer_xyz.csv"
if not OUT.exists():
    with OUT.open("w", newline='') as f:
        csv.writer(f).writerow(["timestamp", "x", "y", "z"])

# 4) Connect client (polling mode)
client = ArduinoIoTCloudClient(
    device_id=DEVICE_ID,
    username=DEVICE_ID,
    password=DEVICE_SECRET,
    sync_node=True
)

client.register(VAR_X, value=None)
client.register(VAR_Y, value=None)
client.register(VAR_Z, value=None)

client.start()
print("Connected. Logging <timestamp,x,y,z> once per second. (Ctrl+C to stop)")

latest_x = latest_y = latest_z = None

try:
    while True:
        # Poll fast for fresh values during this 1-second window
        end = time.time() + 1.0
        while time.time() < end:
            client.update()
            x = client.get(VAR_X); y = client.get(VAR_Y); z = client.get(VAR_Z)
            if x is not None: latest_x = x
            if y is not None: latest_y = y
            if z is not None: latest_z = z
            time.sleep(0.03) # ~33 Hz polling

        timestamp = datetime.now().isoformat()
        with OUT.open("a", newline='') as f:
            csv.writer(f).writerow([timestamp, latest_x, latest_y, latest_z])

        latest_x = latest_y = latest_z = None
```

Data Table (accelerometer_xyz.csv):

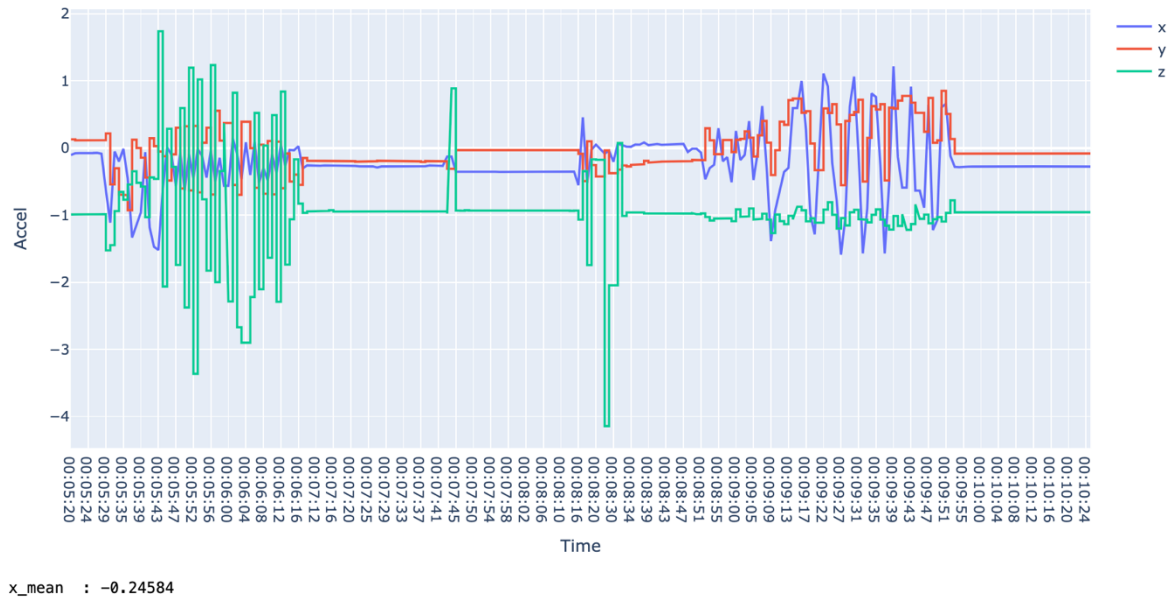
	timestamp	x	y	z
1	2025-09-18T14:13-40	-0.44207786071875	0.3365020275193125	-1.1291901669921875
2	2025-09-18T14:13-41	-0.44207786071875	0.3365020275193125	-1.1291901669921875
3	2025-09-18T14:13-42	-0.44207786071875	0.3365020275193125	-1.1291901669921875
4	2025-09-18T14:13-43	-0.44207786071875	0.3365020275193125	-1.1291901669921875
5	2025-09-18T14:13-44	-0.44207786071875	0.3365020275193125	-1.1291901669921875
6	2025-09-18T14:13-45	-0.44207786071875	0.3365020275193125	-1.1291901669921875
7	2025-09-18T14:13-46	-0.44207786071875	0.3365020275193125	-1.1291901669921875
8	2025-09-18T14:13-47	-0.44207786071875	0.3365020275193125	-1.1291901669921875
9	2025-09-18T14:13-48	-0.44207786071875	0.3365020275193125	-1.1291901669921875
10	2025-09-18T14:13-49	-0.44207786071875	0.3365020275193125	-1.1291901669921875
11	2025-09-18T14:13-50	-0.44207786071875	0.3365020275193125	-1.1291901669921875
12	2025-09-18T14:13-51	-0.44207786071875	0.3365020275193125	-1.1291901669921875
13	2025-09-18T14:13-52	-0.44207786071875	0.3365020275193125	-1.1291901669921875
14	2025-09-18T14:13-53	-0.44207786071875	0.3365020275193125	-1.1291901669921875
15	2025-09-18T14:13-54	-0.44207786071875	0.3365020275193125	-1.1291901669921875
16	2025-09-18T14:13-55	-0.44207786071875	0.3365020275193125	-1.1291901669921875
17	2025-09-18T14:13-56	-0.44207786071875	0.3365020275193125	-1.1291901669921875
18	2025-09-18T14:13-57	-0.44207786071875	0.3365020275193125	-1.1291901669921875
19	2025-09-18T14:13-58	-0.44207786071875	0.3365020275193125	-1.1291901669921875
20	2025-09-18T14:13-59	-0.44207786071875	0.3365020275193125	-1.1291901669921875
21	2025-09-18T14:14-00	-0.44207786071875	0.3365020275193125	-1.1291901669921875
22	2025-09-18T14:14-01	-0.44207786071875	0.3365020275193125	-1.1291901669921875
23	2025-09-18T14:14-03	-0.44207786071875	0.3365020275193125	-1.1291901669921875
24	2025-09-18T14:14-04	-0.44207786071875	0.3365020275193125	-1.1291901669921875
25	2025-09-18T14:14-05	-0.44207786071875	0.3365020275193125	-1.1291901669921875
26	2025-09-18T14:14-06	-0.44207786071875	0.3365020275193125	-1.1291901669921875
27	2025-09-18T14:14-07	-0.44207786071875	0.3365020275193125	-1.1291901669921875
28	2025-09-18T14:14-08	-0.44207786071875	0.3365020275193125	-1.1291901669921875
29	2025-09-18T14:14-09	-0.44207786071875	0.3365020275193125	-1.1291901669921875
30	2025-09-18T14:14-10	-0.44207786071875	0.3365020275193125	-1.1291901669921875
31	2025-09-18T14:14-11	-0.44207786071875	0.3365020275193125	-1.1291901669921875
32	2025-09-18T14:14-12	-0.44207786071875	0.3365020275193125	-1.1291901669921875
33	2025-09-18T14:14-13	-0.44207786071875	0.3365020275193125	-1.1291901669921875
34	2025-09-18T14:14-14	-0.44207786071875	0.3365020275193125	-1.1291901669921875
35	2025-09-18T14:14-15	-0.44207786071875	0.3365020275193125	-1.1291901669921875
36	2025-09-18T14:14-16	-0.44207786071875	0.3365020275193125	-1.1291901669921875
37	2025-09-18T14:14-17	-0.44207786071875	0.3365020275193125	-1.1291901669921875
38	2025-09-18T14:14-18	-0.44207786071875	0.3365020275193125	-1.1291901669921875
39	2025-09-18T14:14-19	-0.44207786071875	0.3365020275193125	-1.1291901669921875
40	2025-09-18T14:14-20	-0.44207786071875	0.3365020275193125	-1.1291901669921875
41	2025-09-18T14:14-21	-0.44207786071875	0.3365020275193125	-1.1291901669921875

Task Objective Questions –

1. Summaries your analysis of accelerometer activity separation describing your method, results and discuss.

Ans -

Batch size N = 150. Press 'q' in the terminal to stop capture.



From the graph we can see that there are 3 different types of pf readings. On the left hand side, we can see that there are similar patterns of green line(z axis) because I was moving my phone in the 3rd dimension up and down very fast whereas in the middle we can see that the lines are straight kind of static that's because the phone was lying still on the table. On the right hand side we can see similar patterns with purple lines that's because I was moving my phone horizontally in the x axis hence. This concluded the analysis of the live accelerometer data transmission from my phone to lot Cloud and then storing the data and making a graph out of it.

2. Link to the video demonstration - https://youtube.com/shorts/U5En_Qt-Gil

3.

The screenshot displays the GitHub web interface for the repository 'SIT225_2024T2' by user 'Adityasuhag001'. The left sidebar shows the repository's file structure, with 'week 8' selected. The main area shows the commit history for 'week 8', listing files like 'Activity 6.1.ipynb', 'Task objective.ipynb', and various accelerometer CSV files. The commit messages are all 'Add files via upload'.

Files

- main
- .ipynb_checkpoints
- week 1
- week 6
- week 6
- week 7
- week 8**
 - Activity 6.1.ipynb
 - Activity 6.2.ipynb
 - Task objective.ipynb
 - accelerometer_x.csv
 - accelerometer_xyz.csv
 - accelerometer_y.csv
 - accelerometer_z.csv
 - content
 - iPhone Thing-Accelerometer_X...
 - iPhone Thing-Accelerometer_Yc...
 - iPhone Thing-Accelerometer_Z...
 - iPhone Thing-Gps.csv
- week-5
- week2
- 6.1PSIT225.ipynb
- Git setup manual.pdf
- gyro.csv
- plotly_explore.ipynb

SIT225_2024T2 / week 8

Adityasuhag001 Add files via upload 58cb51a · now History

Name	Last commit message	Last commit date
..		
Activity 6.1.ipynb	Add files via upload	now
Activity 6.2.ipynb	Add files via upload	now
Task objective.ipynb	Add files via upload	now
accelerometer_x.csv	Add files via upload	now
accelerometer_xyz.csv	Add files via upload	now
accelerometer_y.csv	Add files via upload	now
accelerometer_z.csv	Add files via upload	now
content	Create content	1 minute ago
iPhone Thing-Accelerometer_X.csv	Add files via upload	now
iPhone Thing-Accelerometer_Y.csv	Add files via upload	now
iPhone Thing-Accelerometer_Z.csv	Add files via upload	now
iPhone Thing-Gps.csv	Add files via upload	now