Student name: Zakarya Guerinat

Student ID:

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

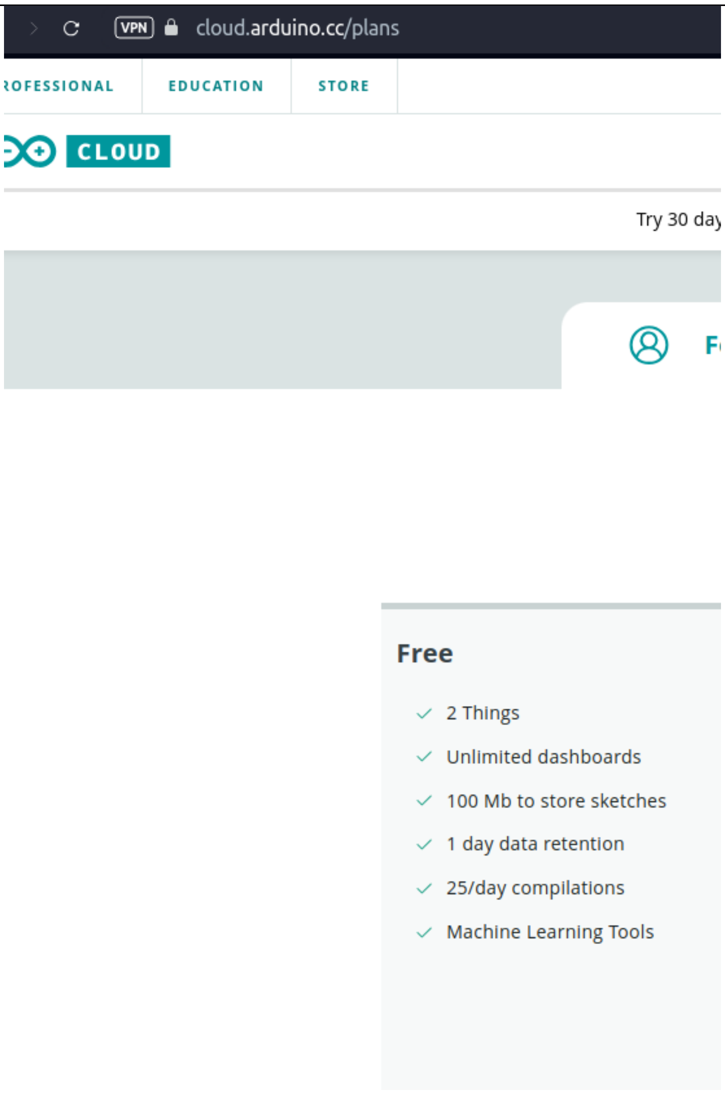
NOTE: *The IoT Remote app requires iOS 12.4 or later for iOS the version. If you are using Android, version 8.0 or later is required. Make sure the iOS or Android version on your device is up to date before downloading the app.*

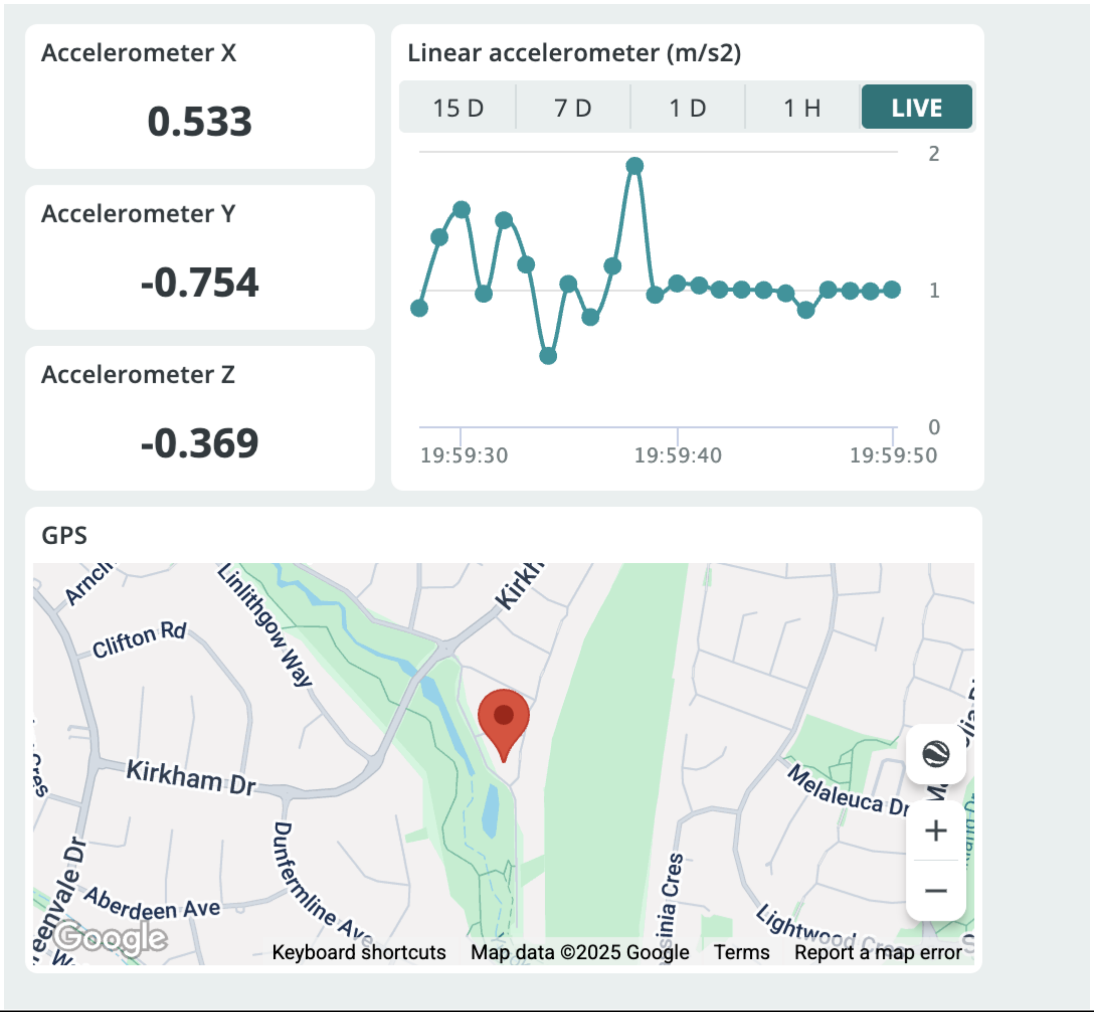
Software Required

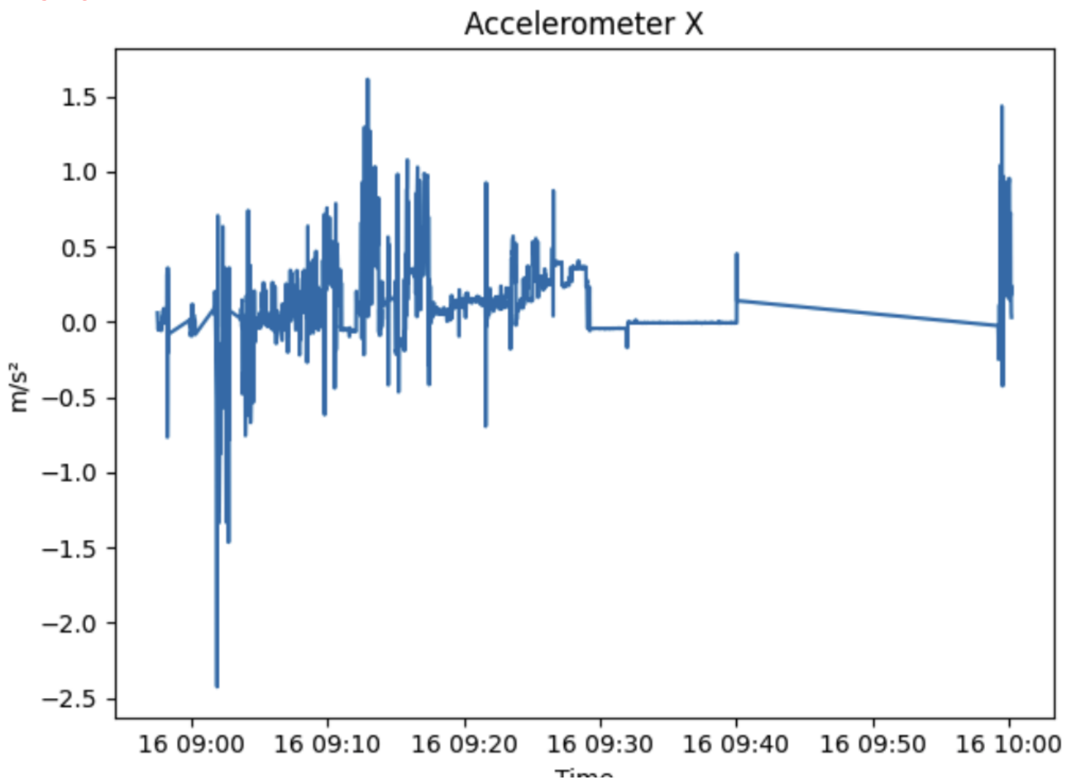
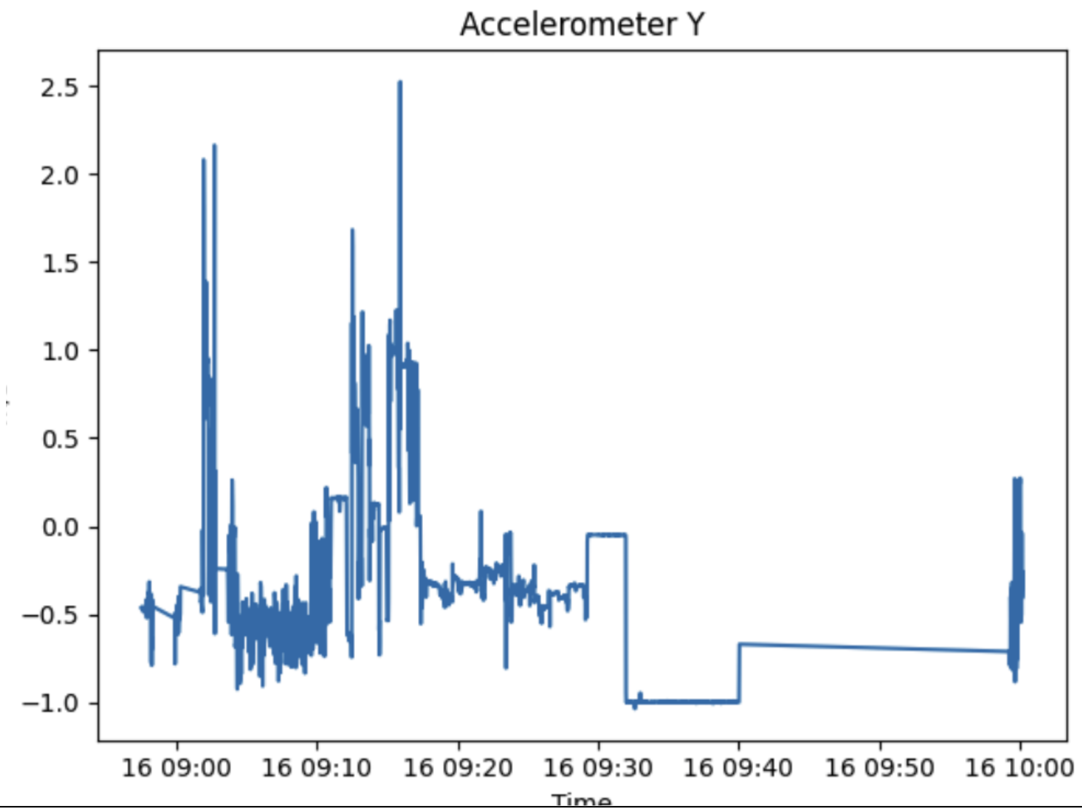
Android / iOS smart phone.
Arduino account
Arduino IoT Remote App (App Store or Google Play)
Python 3 (for custom Python Thing)

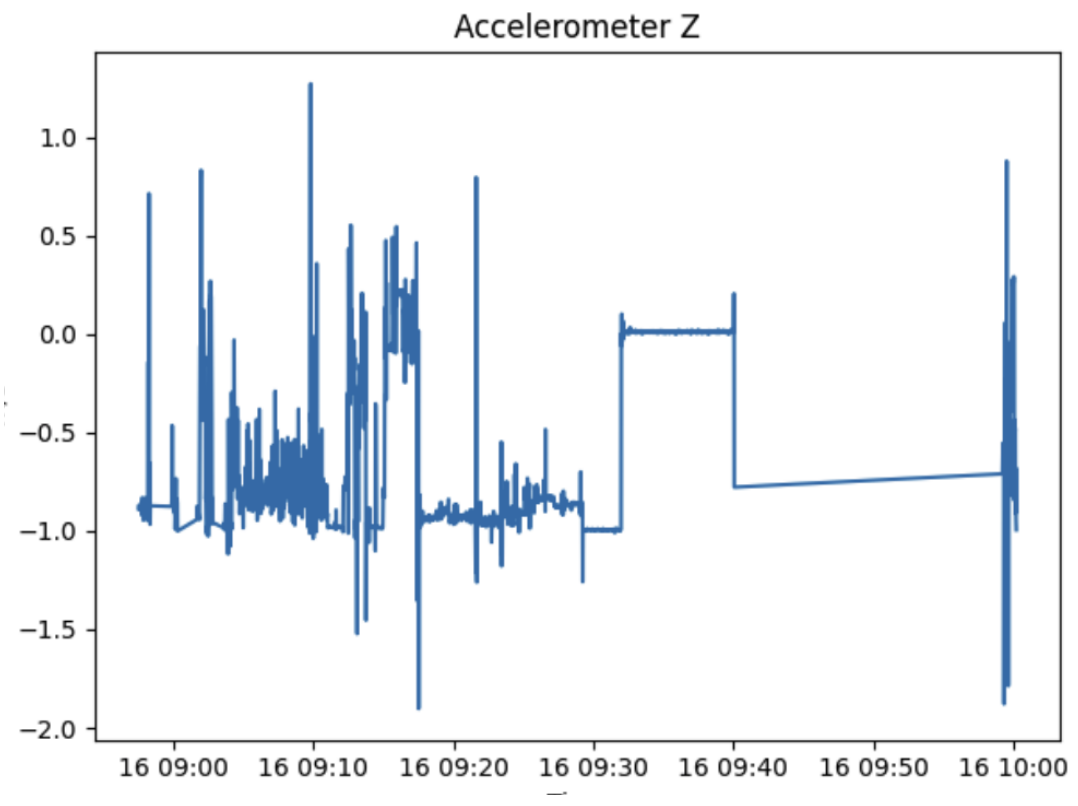
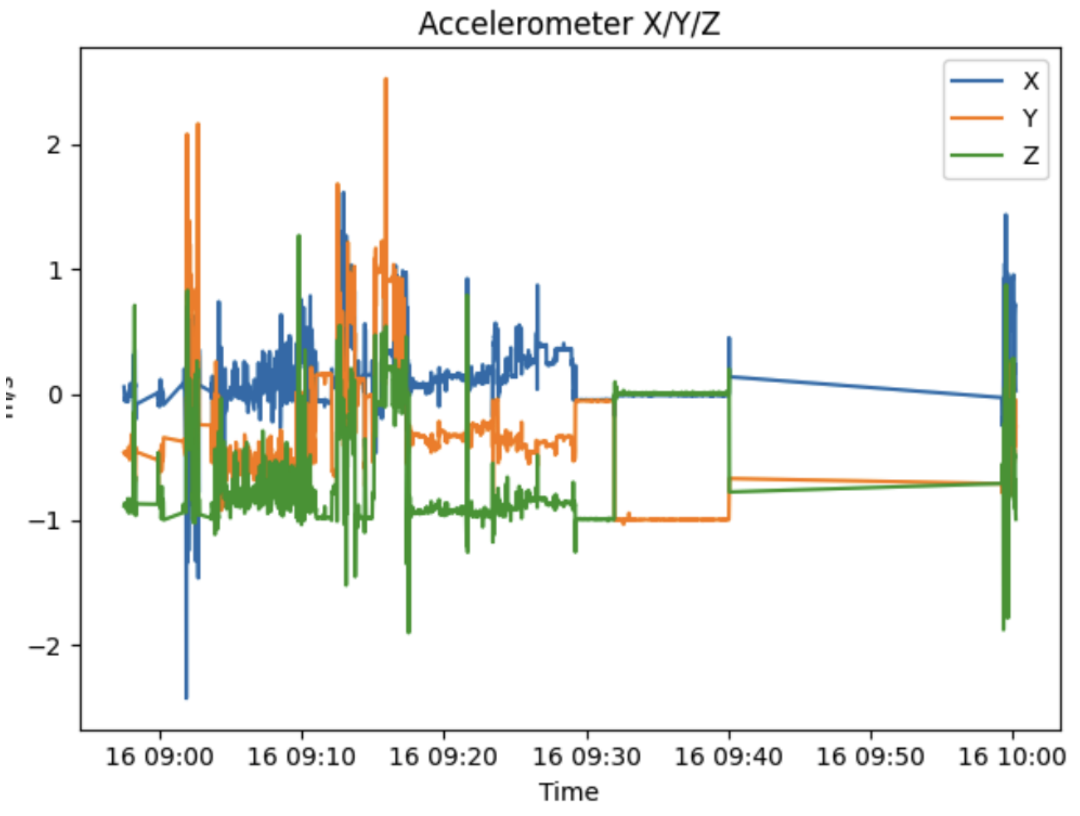
Steps

Step	Action
1	**Install App**: To use the Arduino IoT Remote app, visit Google Play / App Store and search for "Arduino IoT Remote". After installing the app, you will need to log in to your Arduino account. 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).
2	**Add device**: 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. 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.

	 VPN cloud.arduino.cc/plans ROFESSIONAL EDUCATION STORE CLOUD Try 30 day F Free ✓ 2 Things ✓ Unlimited dashboards ✓ 100 Mb to store sketches ✓ 1 day data retention ✓ 25/day compilations ✓ Machine Learning Tools 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.
3	**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. You can download data from the Ardiuno 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.

	Question: Take a screenshot of your Ardiuno Cloud Dashboard where smartphone data is streaming and paste it here. Answer:  Accelerometer X 0.533 Accelerometer Y -0.754 Accelerometer Z -0.369 Linear accelerometer (m/s2) 15 D 7 D 1 D 1 H LIVE 2 1 0 19:59:30 19:59:40 19:59:50 GPS Clifton Rd Linlithgow Way Kirkham Dr Dunfermline Ave Aberdeen Ave Melaleuca Dr Keyboard shortcuts Map data ©2025 Google Terms Report a map error
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:  Accelerometer X m/s² 1.5 1.0 0.5 0.0 −0.5 −1.0 −1.5 −2.0 −2.5 16 09:00 16 09:10 16 09:20 16 09:30 16 09:40 16 09:50 16 10:00  Accelerometer Y 2.5 2.0 1.5 1.0 0.5 0.0 −0.5 −1.0 16 09:00 16 09:10 16 09:20 16 09:30 16 09:40 16 09:50 16 10:00 Time

	 Accelerometer Z 1.0 0.5 0.0 −0.5 −1.0 −1.5 −2.0 16 09:00 16 09:10 16 09:20 16 09:30 16 09:40 16 09:50 16 10:00  Accelerometer X/Y/Z X Y Z 2 1 0 −1 −2 16 09:00 16 09:10 16 09:20 16 09:30 16 09:40 16 09:50 16 10:00 Time

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: When looking at the graphs, there are clear repeating peaks and troughs in the accelerometer data. For example, the X-axis shows a consistent wave pattern, which corresponds to the side-to-side movement I repeated while holding the phone. The Y and Z axes remain more stable, showing only smaller variations caused by minor tilts and gravity. This repeating wave pattern confirms that the phone was moved in a regular rhythm, and the periodicity in the X data matches the motion I performed.

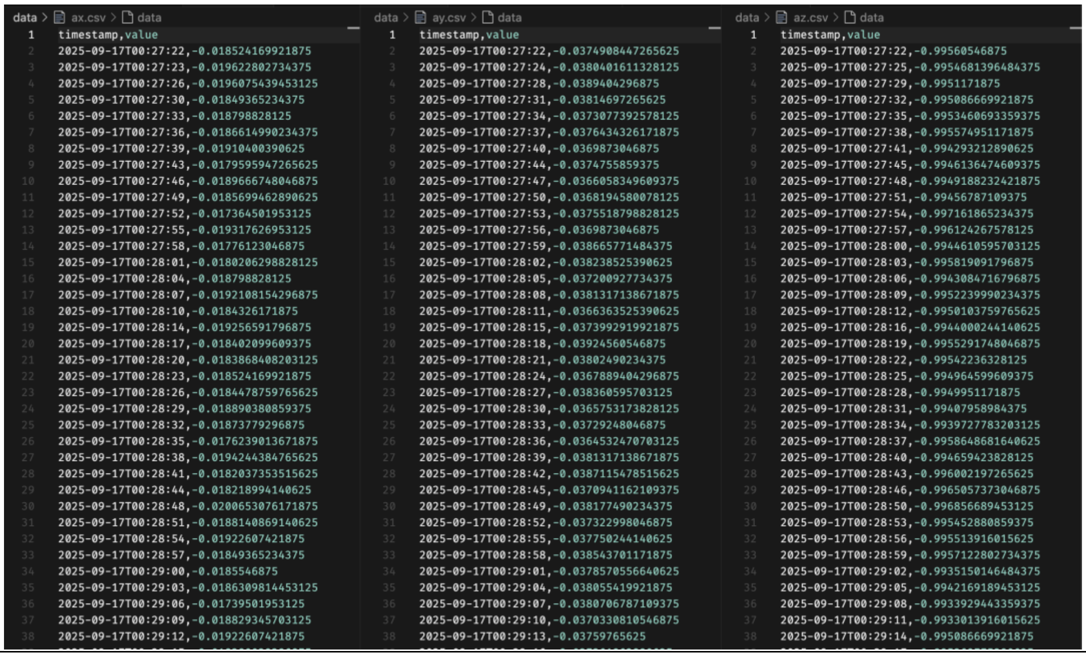
Activity 8.2: Receive smartphone sensor data from Python script

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
```
# === Activity 8.2 · Q3: log each axis to its own CSV (Arduino IoT Cloud Python client) ===
from datetime import datetime
import csv, pathlib
from arduino_iot_cloud import ArduinoCloudClient   # official client
from secrets import DEVICE_ID, SECRET_KEY

# --- file helpers ---
DATA_DIR = pathlib.Path("data"); DATA_DIR.mkdir(exist_ok=True)
def ts(): return datetime.now().isoformat(timespec="seconds")
def append_row(path, values):
    new = not path.exists()
    with path.open("a", newline="") as f:
        w = csv.writer(f)
        if new: w.writerow(["timestamp", "value"])
        w.writerow(values)

# --- callbacks (called whenever the cloud variable updates) ---
def on_ax(client, value):
    append_row(DATA_DIR/"ax.csv", [ts(), value])
    print(f"[ax] {value}")

def on_ay(client, value):
    append_row(DATA_DIR/"ay.csv", [ts(), value])
    print(f"[ay] {value}")

def on_az(client, value):
    append_row(DATA_DIR/"az.csv", [ts(), value])
    print(f"[az] {value}")

# --- connect to Arduino IoT Cloud and register your variables ---
client = ArduinoCloudClient(device_id=DEVICE_ID, username=DEVICE_ID,
password=SECRET_KEY)
# These names MUST match the variable names you created in the Thing (ax,
ay, az)
# We listen for updates from the Cloud (linked to your phone’s
Accelerometer X/Y/Z)
client.register("ax", value=None, on_write=on_ax)
client.register("ay", value=None, on_write=on_ay)
client.register("az", value=None, on_write=on_az)

print("Connecting to Arduino IoT Cloud…")
client.start()  # starts the client loop; on_write triggers whenever
values change
```



```
timestamp,value
2025-09-17T00:27:22,-0.018524169921875
2025-09-17T00:27:23,-0.019622802734375
2025-09-17T00:27:26,-0.0196075439453125
2025-09-17T00:27:30,-0.01849365234375
2025-09-17T00:27:33,-0.018798828125
2025-09-17T00:27:36,-0.0186614990234375
2025-09-17T00:27:39,-0.01910400390625
2025-09-17T00:27:43,-0.0179595947265625
2025-09-17T00:27:46,-0.0189666748046875
2025-09-17T00:27:49,-0.0185699462890625
2025-09-17T00:27:52,-0.017364501953125
2025-09-17T00:27:55,-0.019317626953125
2025-09-17T00:27:58,-0.01776123046875
2025-09-17T00:28:01,-0.0180206298828125
2025-09-17T00:28:04,-0.018798828125
2025-09-17T00:28:07,-0.0192108154296875
2025-09-17T00:28:10,-0.0184326171875
2025-09-17T00:28:14,-0.019256591796875
2025-09-17T00:28:17,-0.018402099609375
2025-09-17T00:28:20,-0.0183868408203125
2025-09-17T00:28:23,-0.018524169921875
2025-09-17T00:28:26,-0.0184478759765625
2025-09-17T00:28:29,-0.018890380859375
2025-09-17T00:28:32,-0.01873779296875
2025-09-17T00:28:35,-0.0176239013671875
2025-09-17T00:28:38,-0.0194244384765625
2025-09-17T00:28:41,-0.0182037353515625
2025-09-17T00:28:44,-0.018218994140625
2025-09-17T00:28:48,-0.0200653076171875
2025-09-17T00:28:51,-0.0188140869140625
2025-09-17T00:28:54,-0.01922607421875
2025-09-17T00:28:57,-0.01849365234375
2025-09-17T00:29:00,-0.0185546875
2025-09-17T00:29:03,-0.0186309814453125
2025-09-17T00:29:06,-0.01739501953125
2025-09-17T00:29:09,-0.018829345703125
2025-09-17T00:29:12,-0.01922607421875
```

```
timestamp,value
2025-09-17T00:27:22,-0.0374908447265625
2025-09-17T00:27:24,-0.0380401611328125
2025-09-17T00:27:28,-0.0389404296875
2025-09-17T00:27:31,-0.03814697265625
2025-09-17T00:27:34,-0.0373077392578125
2025-09-17T00:27:37,-0.0376434326171875
2025-09-17T00:27:40,-0.0369873046875
2025-09-17T00:27:44,-0.0374755859375
2025-09-17T00:27:47,-0.0366058349609375
2025-09-17T00:27:50,-0.0368194580078125
2025-09-17T00:27:53,-0.0375518798828125
2025-09-17T00:27:56,-0.0369873046875
2025-09-17T00:27:59,-0.038665771484375
2025-09-17T00:28:02,-0.038238525390625
2025-09-17T00:28:05,-0.037200927734375
2025-09-17T00:28:08,-0.0381317138671875
2025-09-17T00:28:11,-0.0366363525390625
2025-09-17T00:28:15,-0.0373992919921875
2025-09-17T00:28:18,-0.03924560546875
2025-09-17T00:28:21,-0.03802490234375
2025-09-17T00:28:24,-0.0367889404296875
2025-09-17T00:28:27,-0.038360595703125
2025-09-17T00:28:30,-0.0365753173828125
2025-09-17T00:28:33,-0.03729248046875
2025-09-17T00:28:36,-0.0364532470703125
2025-09-17T00:28:39,-0.0381317138671875
2025-09-17T00:28:42,-0.0387115478515625
2025-09-17T00:28:45,-0.0370941162109375
2025-09-17T00:28:49,-0.038177490234375
2025-09-17T00:28:52,-0.03732298046875
2025-09-17T00:28:55,-0.037750244140625
2025-09-17T00:28:58,-0.038543701171875
2025-09-17T00:29:01,-0.0378570556640625
2025-09-17T00:29:04,-0.038055419921875
2025-09-17T00:29:07,-0.0380706787109375
2025-09-17T00:29:10,-0.0370330810546875
2025-09-17T00:29:13,-0.03759765625
```

```
timestamp,value
2025-09-17T00:27:22,-0.99560546875
2025-09-17T00:27:25,-0.9954681396484375
2025-09-17T00:27:29,-0.9951171875
2025-09-17T00:27:32,-0.995086669921875
2025-09-17T00:27:35,-0.9953460693359375
2025-09-17T00:27:38,-0.995574951171875
2025-09-17T00:27:41,-0.994293212890625
2025-09-17T00:27:45,-0.9946136474609375
2025-09-17T00:27:48,-0.9949188232421875
2025-09-17T00:27:51,-0.99456787109375
2025-09-17T00:27:54,-0.997161865234375
2025-09-17T00:27:57,-0.996124267578125
2025-09-17T00:28:00,-0.9944610595703125
2025-09-17T00:28:03,-0.995819091796875
2025-09-17T00:28:06,-0.9943084716796875
2025-09-17T00:28:09,-0.9952239990234375
2025-09-17T00:28:12,-0.9950103759765625
2025-09-17T00:28:16,-0.9944000244140625
2025-09-17T00:28:19,-0.9955291748046875
2025-09-17T00:28:22,-0.99542236328125
2025-09-17T00:28:25,-0.994964599609375
2025-09-17T00:28:28,-0.9949951171875
2025-09-17T00:28:31,-0.99407958984375
2025-09-17T00:28:34,-0.9939727783203125
2025-09-17T00:28:37,-0.9958648681640625
2025-09-17T00:28:40,-0.994659423828125
2025-09-17T00:28:43,-0.996002197265625
2025-09-17T00:28:46,-0.9965057373046875
2025-09-17T00:28:50,-0.996856689453125
2025-09-17T00:28:53,-0.995452880859375
2025-09-17T00:28:56,-0.995513916015625
2025-09-17T00:28:59,-0.9957122802734375
2025-09-17T00:29:02,-0.9935150146484375
2025-09-17T00:29:05,-0.9942169189453125
2025-09-17T00:29:08,-0.9933929443359375
2025-09-17T00:29:11,-0.9933013916015625
2025-09-17T00:29:14,-0.995086669921875
```

4

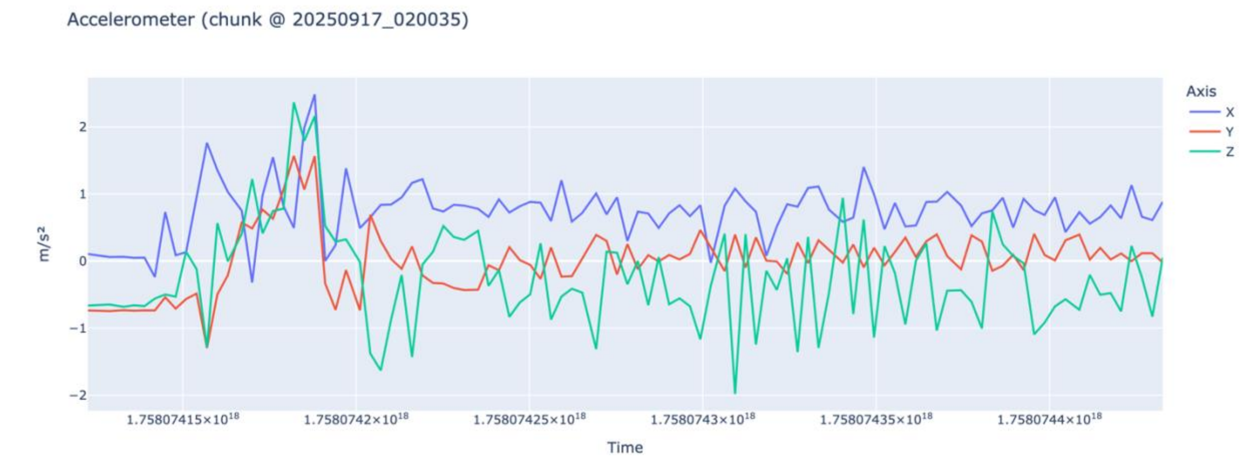
Question: Now manage 3 variable data so they can be stored in a single CSV file where each line consists of comma separated sensor values with a timestamp - <timestamp>, <x>, <y>, <z>. Store data once you gather 3 variables and repeat the process. Screenshot your Python script here and screenshot the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

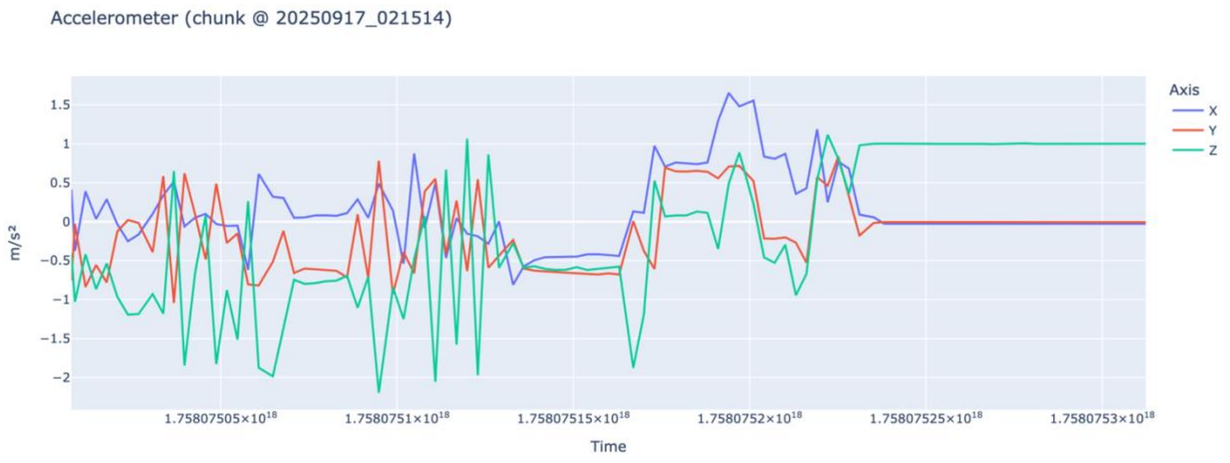
Answer:
```python
# === Activity 8.2 · Q4: single CSV with timestamp, x, y, z ===
from datetime import datetime
import csv, pathlib
from arduino_iot_cloud import ArduinoCloudClient
from secrets import DEVICE_ID, SECRET_KEY

DATA_DIR = pathlib.Path("data"); DATA_DIR.mkdir(exist_ok=True)
ONE = DATA_DIR/"accelerometer_xyz.csv"
latest = {"ax": None, "ay": None, "az": None}

def write_if_ready():
    if all(v is not None for v in latest.values()):
        stamp = datetime.now().isoformat(timespec="seconds")
        new = not ONE.exists()
        with ONE.open("a", newline="") as f:
            w = csv.writer(f)
            if new: w.writerow(["timestamp", "ax", "ay", "az"])
            w.writerow([stamp, latest["ax"], latest["ay"], latest["az"]])
        print(f"[row] {stamp}, {latest['ax']}, {latest['ay']}, {latest['az']}")

def on_ax(client, value): latest["ax"] = value; write_if_ready()
def on_ay(client, value): latest["ay"] = value; write_if_ready()
def on_az(client, value): latest["az"] = value; write_if_ready()

client = ArduinoCloudClient(device_id=DEVICE_ID, username=DEVICE_ID, password=SECRET_KEY)
client.register("ax", value=None, on_write=on_ax)
client.register("ay", value=None, on_write=on_ay)
client.register("az", value=None, on_write=on_az)

print("Connecting to Arduino IoT Cloud…")
client.start()
```

```
timestamp,ax,ay,az
2025-09-17T00:44:21,-0.097991943359375,-0.6552581787109375,-0.6194610595703125
2025-09-17T00:44:29,0.27734375,-0.6552581787109375,-0.6194610595703125
2025-09-17T00:44:30,0.27734375,-0.8281707763671875,-0.6194610595703125
2025-09-17T00:44:31,0.27734375,-0.8281707763671875,-0.0981292724609375
```

```
2025-09-17T00:44:32,0.0138397216796875,-0.8281707763671875,-
0.0981292724609375
2025-09-17T00:44:33,0.0138397216796875,-0.833663940429687 5,-
0.0981292724609375
2025-09-17T00:44:34,0.0138397216796875,-0.833663940429687 5,-
0.52191162109375
2025-09-17T00:44:35,0.0135955810546875,-0.833663940429687 5,-
0.52191162109375
2025-09-17T00:44:36,0.0135955810546875,-0.8098297119140625,-
0.52191162109375
2025-09-17T00:44:37,0.0135955810546875,-0.8098297119140625,-
0.5797271728515625
2025-09-17T00:44:38,0.00830078125,-0.8098297119140625,-0.5797271728515625
2025-09-17T00:44:39,0.00830078125,-0.8112335205078125,-0.5797271728515625
2025-09-17T00:44:40,0.00830078125,-0.8112335205078125,-0.577972412109375
2025-09-17T00:44:41,0.015777587890625,-0.8112335205078125,-
0.577972412109375
2025-09-17T00:44:42,0.015777587890625,-0.81683349609375,-0.577972412109375
2025-09-17T00:44:43,0.015777587890625,-0.81683349609375,-
0.5696563720703125
2025-09-17T00:44:44,-0.0099334716796875,-0.81683349609375,-
0.5696563720703125
2025-09-17T00:44:45,-0.0099334716796875,-0.885955810546875,-
0.5696563720703125
2025-09-17T00:44:46,-0.0099334716796875,-0.885955810546875,-
0.5673980712890625
2025-09-17T00:44:47,-0.3015594482421875,-0.885955810546875,-
0.5673980712890625
2025-09-17T00:44:48,-0.3015594482421875,-0.0231781005859375,-
0.5673980712890625
2025-09-17T00:44:49,-0.3015594482421875,-0.0231781005859375,-
1.101348876953125
2025-09-17T00:44:50,-0.017425537109375,-0.0231781005859375,-
1.101348876953125
2025-09-17T00:44:51,-0.017425537109375,-0.07806396484375,-
1.101348876953125
2025-09-17T00:44:52,-0.017425537109375,-0.07806396484375,-
0.9884185791015625
2025-09-17T00:44:53,-0.01715087890625,-0.07806396484375,-
0.9884185791015625
2025-09-17T00:44:54,-0.01715087890625,-0.078582763671875,-
0.9884185791015625
2025-09-17T00:44:55,-0.01715087890625,-0.078582763671875,-
0.9910125732421875
2025-09-17T00:44:56,-0.017608642578125,-0.078582763671875,-
0.9910125732421875
```

```
2025-09-17T00:44:57,-0.017608642578125,-0.079986572265625,-
0.9910125732421875
2025-09-17T00:44:58,-0.017608642578125,-0.079986572265625,-
0.993194580078125
2025-09-17T00:45:00,-0.01708984375,-0.079986572265625,-0.993194580078125
2025-09-17T00:45:01,-0.01708984375,-0.0779571533203125,-0.993194580078125
2025-09-17T00:45:02,-0.01708984375,-0.0779571533203125,-0.9917144775390625
2025-09-17T00:45:03,-0.0166473388671875,-0.0779571533203125,-
0.9917144775390625
2025-09-17T00:45:04,-0.0166473388671875,-0.0785064697265625,-
0.9917144775390625
2025-09-17T00:45:05,-0.0166473388671875,-0.0785064697265625,-
0.9920654296875
2025-09-17T00:45:06,-0.0173797607421875,-0.0785064697265625,-
0.9920654296875
2025-09-17T00:45:07,-0.0173797607421875,-0.0788421630859375,-
0.9920654296875
2025-09-17T00:45:08,-0.0173797607421875,-0.0788421630859375,-
0.991851806640625
2025-09-17T00:45:09,-0.01727294921875,-0.0788421630859375,-
0.991851806640625
2025-09-17T00:45:10,-0.01727294921875,-0.0772705078125,-0.991851806640625
2025-09-17T00:45:11,-0.01727294921875,-0.0772705078125,-0.99298095703125
2025-09-17T00:45:12,-0.0169525146484375,-0.0772705078125,-0.99298095703125
2025-09-17T00:45:13,-0.0169525146484375,-0.0784912109375,-0.99298095703125
2025-09-17T00:45:14,-0.0169525146484375,-0.0784912109375,-
0.9922332763671875
2025-09-17T00:45:15,-0.0174407958984375,-0.0784912109375,-
0.9922332763671875
2025-09-17T00:45:16,-0.0174407958984375,-0.07830810546875,-
0.9922332763671875
2025-09-17T00:45:17,-0.0174407958984375,-0.07830810546875,-
0.992279052734375
2025-09-17T00:45:18,-0.017303466796875,-0.07830810546875,-
0.992279052734375
2025-09-17T00:45:19,-0.017303466796875,-0.079498291015625,-
0.992279052734375
2025-09-17T00:45:20,-0.017303466796875,-0.079498291015625,-
0.990570068359375
```





In this activity I used Arduino IoT Cloud and a Python Dash app to capture and visualise accelerometer data from my iphone. The script buffered incoming x, y, and z readings, and every set of samples (about 10 seconds of data) was moved into a separate list to plot and save, while new readings continued filling the live buffer. Each chunk of data was saved as both a CSV file and a PNG graph with a timestamp, giving me a history of activities. I performed two different movements with my phone, and the graphs showed clear differences ,one activity had larger and slower swings, while the other showed faster, smaller oscillations. The readings supported this for example, one chunk showed a repeating period of about 6 seconds with higher variation in the Z axis, while the other had a period around 3 seconds with lower spread. These differences in variance, peak-to-peak range, and rhythm make it possible to separate the two activities reliably using the accelerometer data.

https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=9b33add2-8f06-4632-8236-b35a010f9bb7