Student name: Hoang Long Tran

Student ID: s223128143

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

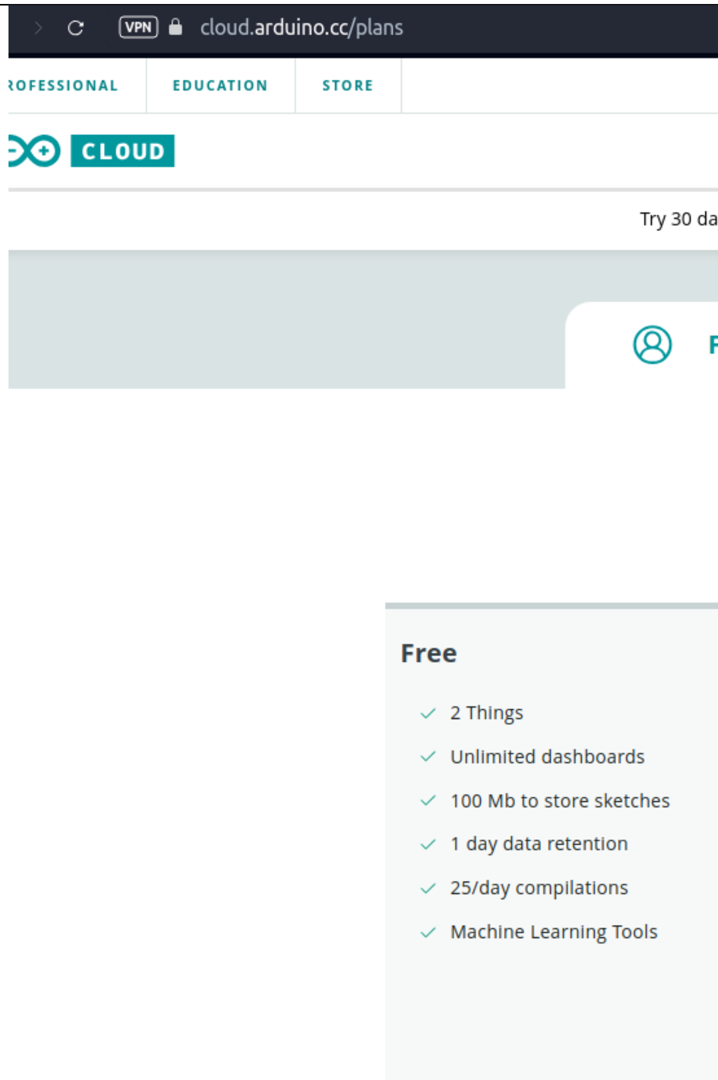
NOTE: *The IoT Remote app requires iOS 12.4 or later for iOS the version. If you are using Android, version 8.0 or later is required. Make sure the iOS or Android version on your device is up to date before downloading the app.*

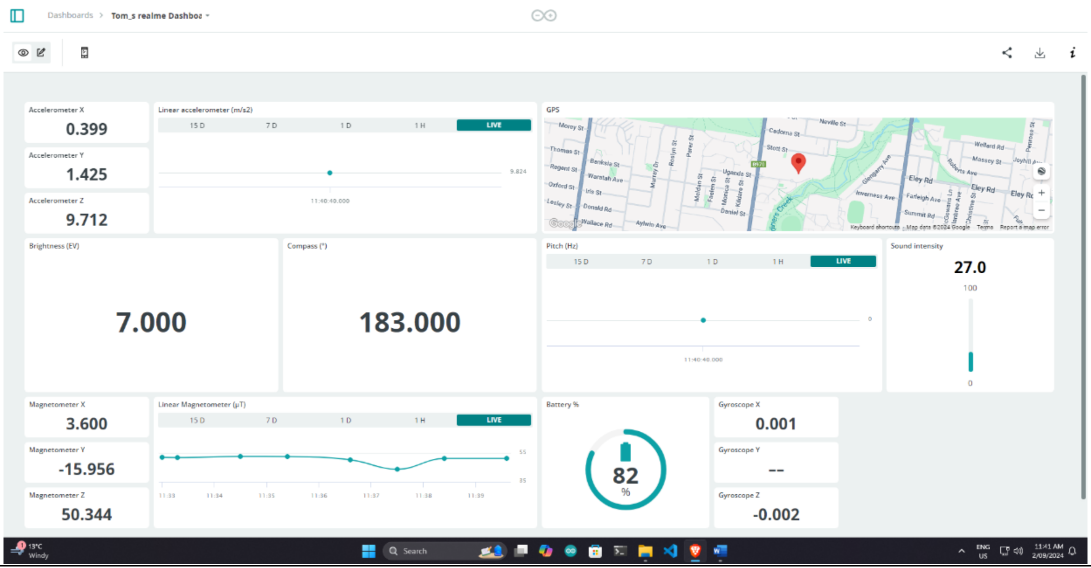
Software Required

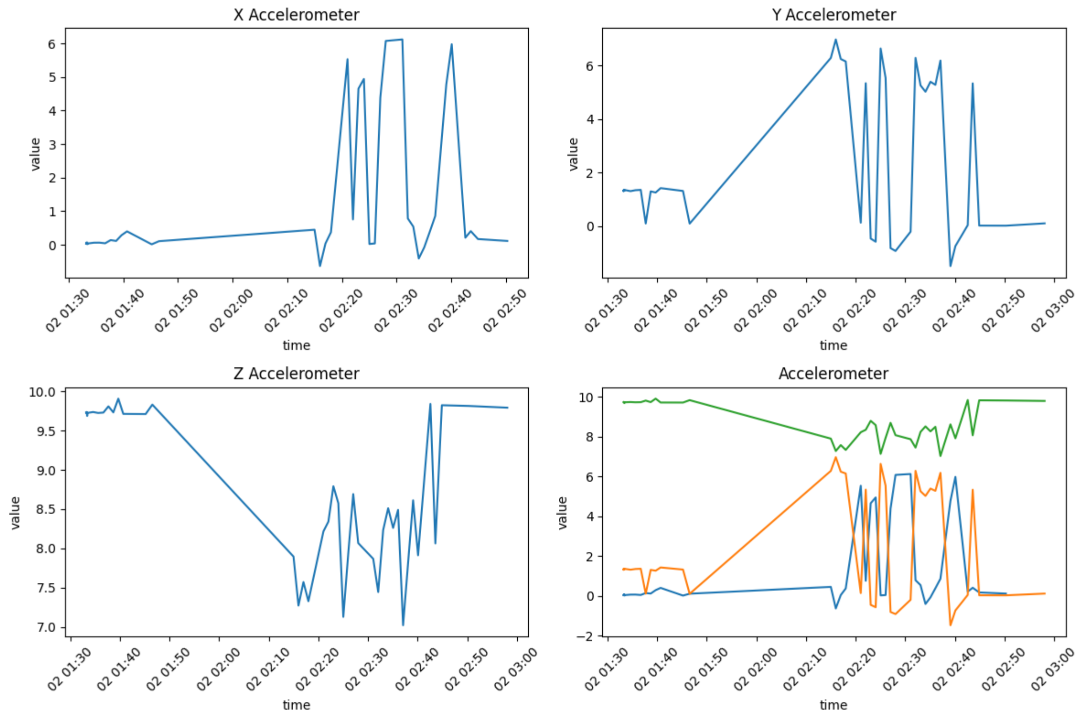
Android / iOS smart phone.
Arduino account
Arduino IoT Remote App (App Store or Google Play)
Python 3 (for custom Python Thing)

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.

	 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: 
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: The time is not in Australian time, I recorded the data at around 12PM. There is a repeated pattern from 12:15 to 12:45, where I was in the kitchen cooking a little bit and eating. During that period of time, I moved my phone a lot, so that explains the fluctuations in the axes.

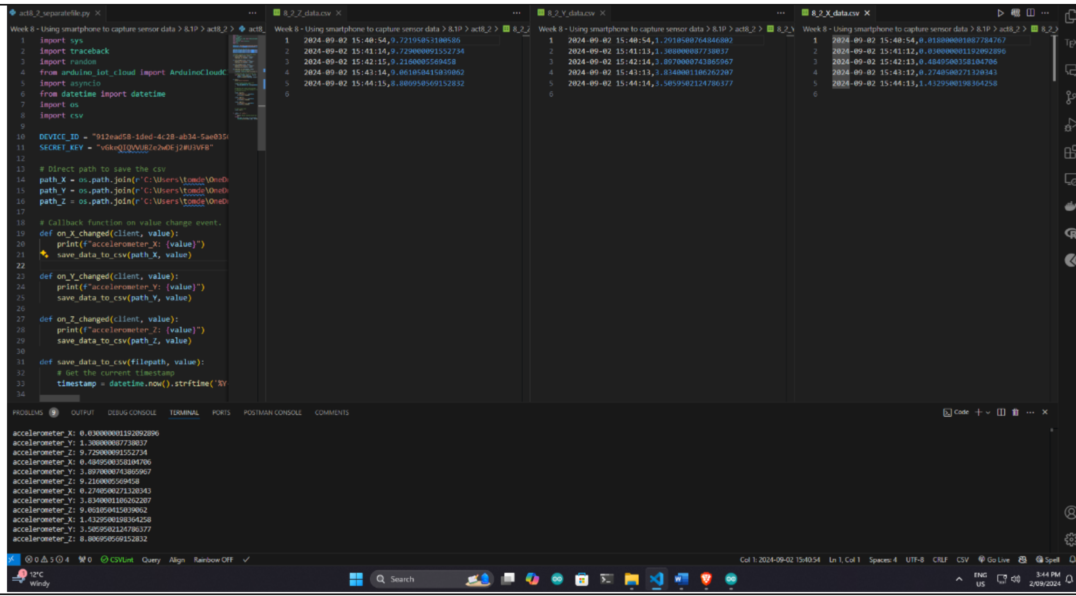
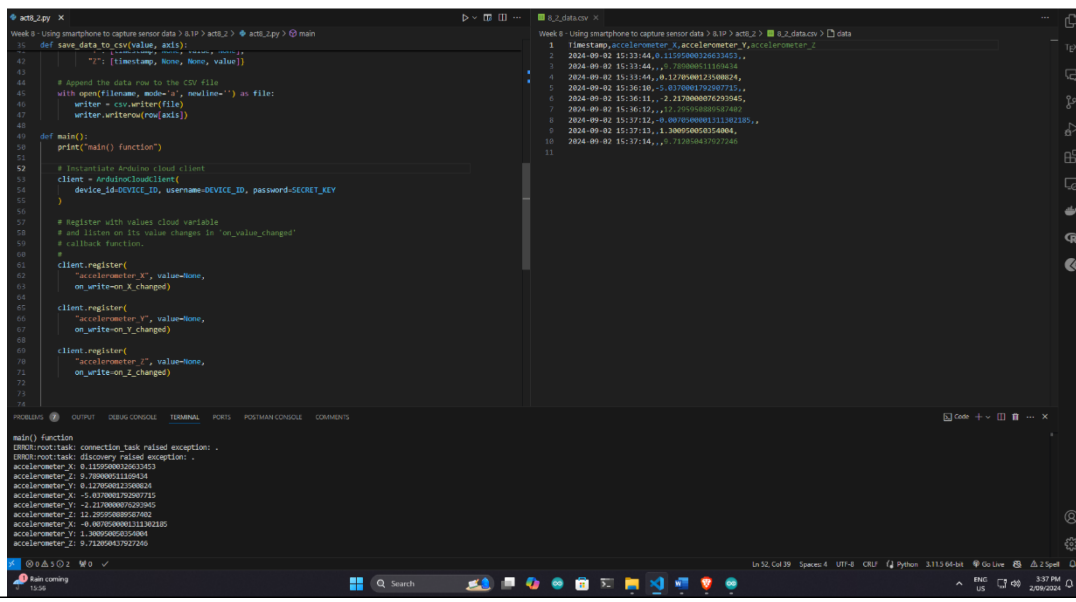
Activity 8.2: Receive smartphone sensor data from Python script

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:

	
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: 

Weekly activity

Q2.

```
import sys
import traceback
from arduino_iot_cloud import ArduinoCloudClient
from datetime import datetime
import os
import csv
import pandas as pd
from dash import Dash, dcc, html
import plotly.graph_objs as go
from dash.dependencies import Input, Output
import threading

# Configuration
DEVICE_ID = "912ead58-1ded-4c28-ab34-5ae0350d52e2"
SECRET_KEY = "vGkeQIQVVUBZe2wDEj2#U3VFB"
N = 510  # Number of samples before plotting

filename = os.path.join(r'C:\Users\tomde\OneDrive\Documents\Deakin\Deakin-Data-Science\T1Y2\SIT225 - Data Capture Technologies\Week 8 - Using smartphone to capture sensor data\8.1P\weekly_act', '8_2_data.csv')
current_row = {"Timestamp": None, "accelerometer_X": None, "accelerometer_Y": None, "accelerometer_Z": None}

# Buffer for data
incoming_data = []
plot_data = []

# Open the CSV file once and keep it open for writing
csv_file = open(filename, mode='a', newline='')
writer = csv.writer(csv_file)

# Load existing data from csv into `incoming_data` list
def load_existing_data():
    global incoming_data, plot_data
    if os.path.exists(filename):
        df = pd.read_csv(filename)
        incoming_data = df.to_dict('records')
        plot_data = incoming_data.copy()  # Copy existing data to plot_data
        print(f"Loaded {len(incoming_data)} records from {filename}")

# Callback functions on value of change event
```

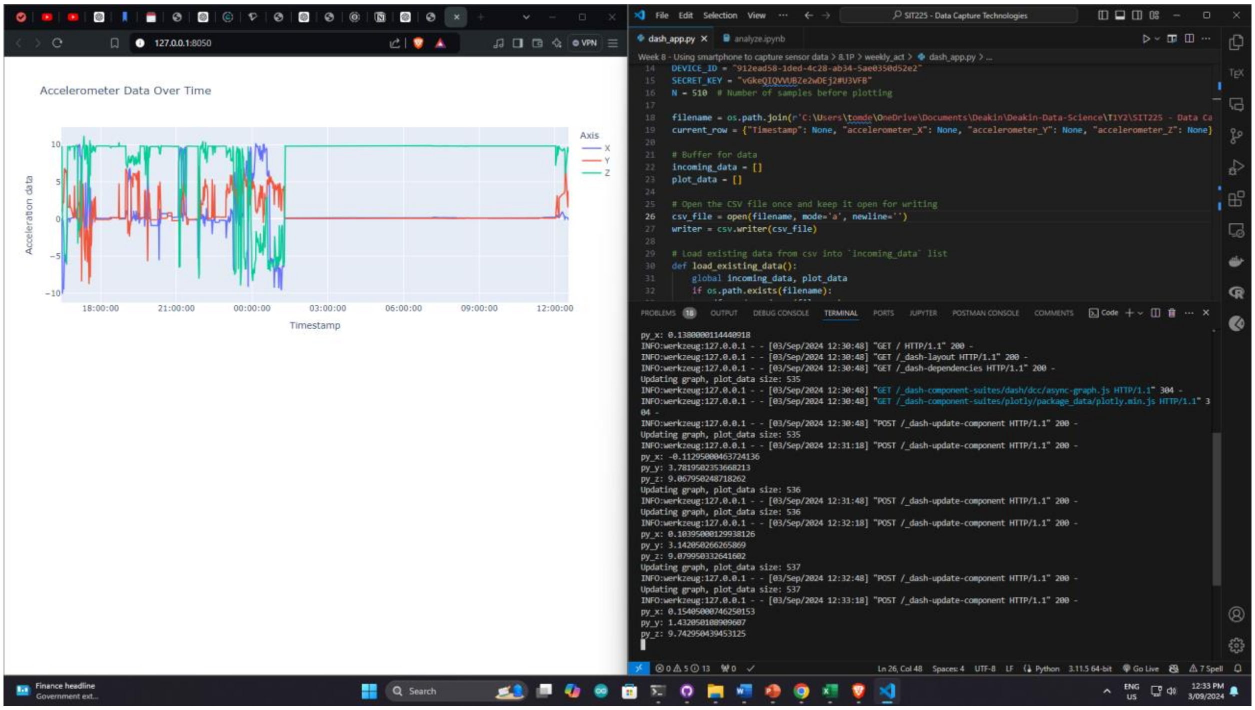
```
def on_X_changed(client, value):
    print(f"py_x: {value}")
    current_row["accelerometer_X"] = value
    current_row["Timestamp"] = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    save_data_to_csv()

def on_Y_changed(client, value):
    print(f"py_y: {value}")
    current_row["accelerometer_Y"] = value
    save_data_to_csv()

def on_Z_changed(client, value):
    print(f"py_z: {value}")
    current_row["accelerometer_Z"] = value
    save_data_to_csv()

# Save to CSV
def save_data_to_csv():
    global current_row, writer, incoming_data, csv_file

    if None not in (current_row["accelerometer_X"],
current_row["accelerometer_Y"], current_row["accelerometer_Z"]):
        writer.writerow([current_row["Timestamp"],
current_row["accelerometer_X"], current_row["accelerometer_Y"],
current_row["accelerometer_Z"]])
        csv_file.flush()
        incoming_data.append(current_row.copy())  # Add the current row to the
buffer
        plot_data.append(current_row.copy())  # Also add the row to plot_data for
live updates

        # Reset current_row for the next set of values
        current_row["accelerometer_X"] = current_row["accelerometer_Y"] =
current_row["accelerometer_Z"] = None

# Dash app
def create_app():
    app = Dash(__name__)

    app.layout = html.Div([
        dcc.Graph(id='live-graph'),
        dcc.Interval(
            id='graph-update',
            interval=30000,  # Update every 30s
            n_intervals=0
```

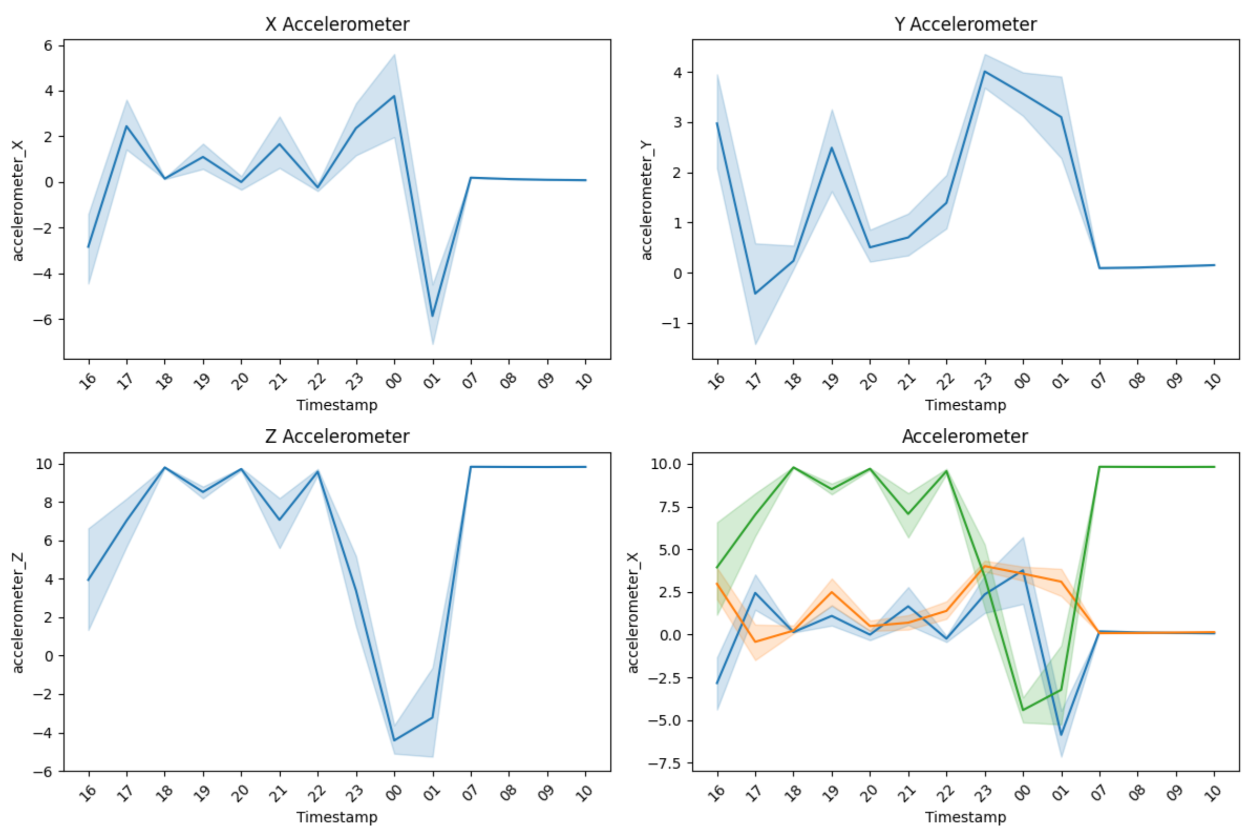
```
        )
    ])

    @app.callback(Output('live-graph', 'figure'), [Input('graph-update', 
'n_intervals')])
    def update_graph(n):
        global plot_data
        print(f"Updating graph, plot_data size: {len(plot_data)}")
        if len(plot_data) > 0:
            # Convert plot_data to DataFrame
            df = pd.DataFrame(plot_data)

            # Convert 'Timestamp' column to datetime dtype
            df['Timestamp'] = pd.to_datetime(df['Timestamp'])

            fig = go.Figure()
            fig.add_trace(go.Scatter(x=df['Timestamp'], y=df['accelerometer_X'], 
mode='lines', name='X'))
            fig.add_trace(go.Scatter(x=df['Timestamp'], y=df['accelerometer_Y'], 
mode='lines', name='Y'))
            fig.add_trace(go.Scatter(x=df['Timestamp'], y=df['accelerometer_Z'], 
mode='lines', name='Z'))

            # Customize the layout
            fig.update_layout(
                title="Accelerometer Data Over Time",
                xaxis_title="Timestamp",
                yaxis_title="Acceleration data",
                legend_title="Axis",
                xaxis=dict(
                    tickformat="%H:%M:%S"
                )
            )

            return fig
        else:
            print("Plot data is empty")
            return go.Figure()

    return app

def start_dash():
    app = create_app()
    app.run_server(debug=False)
```

```python
def main():
    print("Starting data collection...")

    # Load existing data from CSV
    load_existing_data()

    # Instantiate Arduino cloud client
    client = ArduinoCloudClient(
        device_id=DEVICE_ID, username=DEVICE_ID, password=SECRET_KEY
    )

    # Register callbacks
    client.register("py_x", value=None, on_write=on_X_changed)
    client.register("py_y", value=None, on_write=on_Y_changed)
    client.register("py_z", value=None, on_write=on_Z_changed)

    # Start the client
    client.start()

if __name__ == "__main__":
    try:
        dash_thread = threading.Thread(target=start_dash)
        dash_thread.start()
        main()
        dash_thread.join()
    except Exception as e:
        exc_type, exc_value, exc_traceback = sys.exc_info()
        traceback.print_exception(exc_type, exc_value, exc_traceback)
    finally:
        csv_file.close()
```

I will just explain the code. To avoid drawing the same graph twice, I have 2 lists `incoming_data` and `plot_data`. The `load_existing_data` loads data from csv into the `incoming_data` list and copy those data into the `plot_data` list. I also have a function to write the newly gathered data into the csv file. Using the `plot_data` variable to plot the time series plot.

Here is the plot from plotly dash and my analysis in jupyter notebook.





The two plots show the same data I gathered. There are quite a few activities I used with my phone. From 4PM to 12PM, I moved my phone a lot, whether it is in my kitchen, on my table or on my bed, the phone was being moved from places to places. From 1AM onwards, I left my phone stationery on my desk.

Q3.

https://www.youtube.com/watch?v=qK5mTlWffHo

Q4.

https://github.com/tomadonna1/SIT225_2024T2/tree/main/Capture%20smartphone%20sensor%20data