

Student name: Rammaka Iddamalgoda

Student ID: s223496576

SIT225: Data Capture Technologies

Video link:

<https://deakin.au.panopto.com/Panopto/Pages/Sessions/List.aspx?folderID=06a5752e-8f42-48e8-8d67-b1ff00e2a009>

Files included: https://github.com/rammakablecode/SIT225_2024T2

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

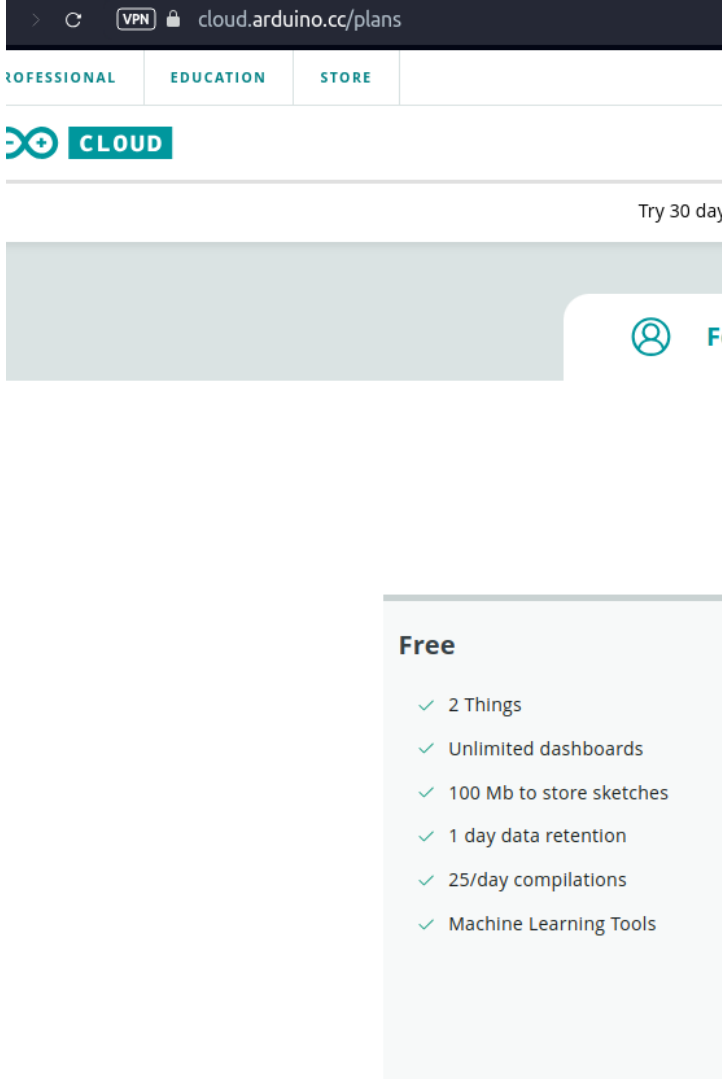
NOTE: *The IoT Remote app requires iOS 12.4 or later for iOS the version. If you are using Android, version 8.0 or later is required. Make sure the iOS or Android version on your device is up to date before downloading the app.*

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: 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: 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>

	<p>Question: Take a screenshot of your Arduino Cloud Dashboard where smartphone data is streaming and paste it here.</p> <p>Answer: <Your answer></p>
4	<p>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.</p> <p>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.</p> <p>Answer: <Your answer></p>
5	<p>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.</p> <p>Answer: <Your answer></p>

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	<p>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.</p> <p>Answer:</p> <pre>import time import csv from datetime import datetime import random # To simulate sensor readings, replace with real sensor data # Simulate fetching accelerometer data for x, y, z def get_sensor_data(): # Replace this with real sensor readings from Arduino Cloud return random.uniform(-10, 10), random.uniform(-10, 10), random.uniform(-10, 10) # File paths for each axis x_file = "x_data.csv" y_file = "y_data.csv" z_file = "z_data.csv" # Continuously append data to separate CSV files while True: # Fetch current timestamp and sensor data timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S') x, y, z = get_sensor_data()</pre>

	<pre> # Write each axis data to its respective file with open(x_file, 'a', newline='') as xf: csv.writer(xf).writerow([timestamp, x]) with open(y_file, 'a', newline='') as yf: csv.writer(yf).writerow([timestamp, y]) with open(z_file, 'a', newline='') as zf: csv.writer(zf).writerow([timestamp, z]) print(f"Data written to files: X={x}, Y={y}, Z={z}") time.sleep(1) # Simulate 1-second delay between sensor readings </pre>
4	<p>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.</p> <p>Answer:</p> <pre> import time import csv from datetime import datetime import random # To simulate sensor readings, replace with real sensor data # Simulate fetching accelerometer data for x, y, z def get_sensor_data(): # Replace this with real sensor readings from Arduino Cloud return random.uniform(-10, 10), random.uniform(-10, 10), random.uniform(-10, 10) # File path for combined data combined_file = "sensor_data.csv" # Continuously append data to the combined CSV file while True: # Fetch current timestamp and sensor data timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S') x, y, z = get_sensor_data() # Write the timestamp and all sensor data to the combined file with open(combined_file, 'a', newline='') as cf: csv.writer(cf).writerow([timestamp, x, y, z]) print(f"Data written to combined file: Timestamp={timestamp}, X={x}, Y={y}, Z={z}") time.sleep(1) # Simulate 1-second delay between sensor readings </pre>

Py:

```

import time
import requests
import pandas as pd
import plotly.graph_objects as go
from dash import Dash, dcc, html
from dash.dependencies import Input, Output

```

```

import csv
from datetime import datetime

# Arduino Cloud API Information
THING_ID = 'your_thing_id'
DEVICE_ID = 'your_device_id'
TOKEN = 'your_auth_token'

# Initialize variables for accelerometer data
x_data = []
y_data = []
z_data = []
timestamps = []

# Initialize Dash app
app = Dash(__name__)

# Layout of the dashboard
app.layout = html.Div([
    html.H1("Accelerometer Data Visualization"),
    dcc.Graph(id="live-graph"),
    dcc.Interval(
        id='interval-component',
        interval=10000, # Update every 10 seconds
        n_intervals=0
    )
])

# Function to get data from Arduino Cloud
def get_accelerometer_data():
    global x_data, y_data, z_data, timestamps
    url = f"https://api2.arduino.cc/iot/v2/things/{THING_ID}/properties"
    headers = {
        'Authorization': f'Bearer {TOKEN}'
    }
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        data = response.json()
        # Assuming x, y, and z properties are available
        py_x = data['py_x']
        py_y = data['py_y']
        py_z = data['py_z']

        x_data.append(py_x)
        y_data.append(py_y)
        z_data.append(py_z)
        timestamps.append(datetime.now().strftime('%H:%M:%S'))

# Save data to CSV
with open('accelerometer_data.csv', 'a', newline="") as f:

```

```

writer = csv.writer(f)
writer.writerow([datetime.now(), py_x, py_y, py_z])

# Callback to update graph
@app.callback(
    Output('live-graph', 'figure'),
    [Input('interval-component', 'n_intervals')]
)
def update_graph(n):
    get_accelerometer_data()

    fig = go.Figure()

    fig.add_trace(go.Scatter(x=timestamps, y=x_data, mode='lines+markers', name='X-axis'))
    fig.add_trace(go.Scatter(x=timestamps, y=y_data, mode='lines+markers', name='Y-axis'))
    fig.add_trace(go.Scatter(x=timestamps, y=z_data, mode='lines+markers', name='Z-axis'))

    fig.update_layout(title="Accelerometer Data (X, Y, Z)",
                      xaxis_title="Timestamp",
                      yaxis_title="Accelerometer Values")

    return fig

if __name__ == '__main__':
    app.run_server(debug=True)

```



```
In [ ]: # Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt

# Load CSV file with accelerometer data
data = pd.read_csv('accelerometer_data.csv', names=['Timestamp', 'X', 'Y', 'Z'])

# Convert timestamp to datetime for easier plotting
data['Timestamp'] = pd.to_datetime(data['Timestamp'])

# Plot the data
plt.figure(figsize=(10, 6))
plt.plot(data['Timestamp'], data['X'], label='X-axis')
plt.plot(data['Timestamp'], data['Y'], label='Y-axis')
plt.plot(data['Timestamp'], data['Z'], label='Z-axis')
plt.xlabel('Time')
plt.ylabel('Accelerometer Readings')
plt.title('Accelerometer X, Y, Z Data Over Time')
plt.legend()
plt.show()

# Save filtered data
filtered_data = data[(data['X'] < 50) & (data['Y'] < 50)] # Example filter
filtered_data.to_csv('filtered_accelerometer_data.csv', index=False)
```

In []: