Deusto
Facultad de Ingeniería
Ingeniaritza Fakultatea

IoT Challenge, Final part

Naia Landaluce
Grupo 8

Deusto
Facultad de Ingeniería
Ingeniaritza Fakultatea

FINAL PROJECT PART ONE:

Collecting data and creating a Python script:

Step 1: Find an Application to Record GPS Tracks

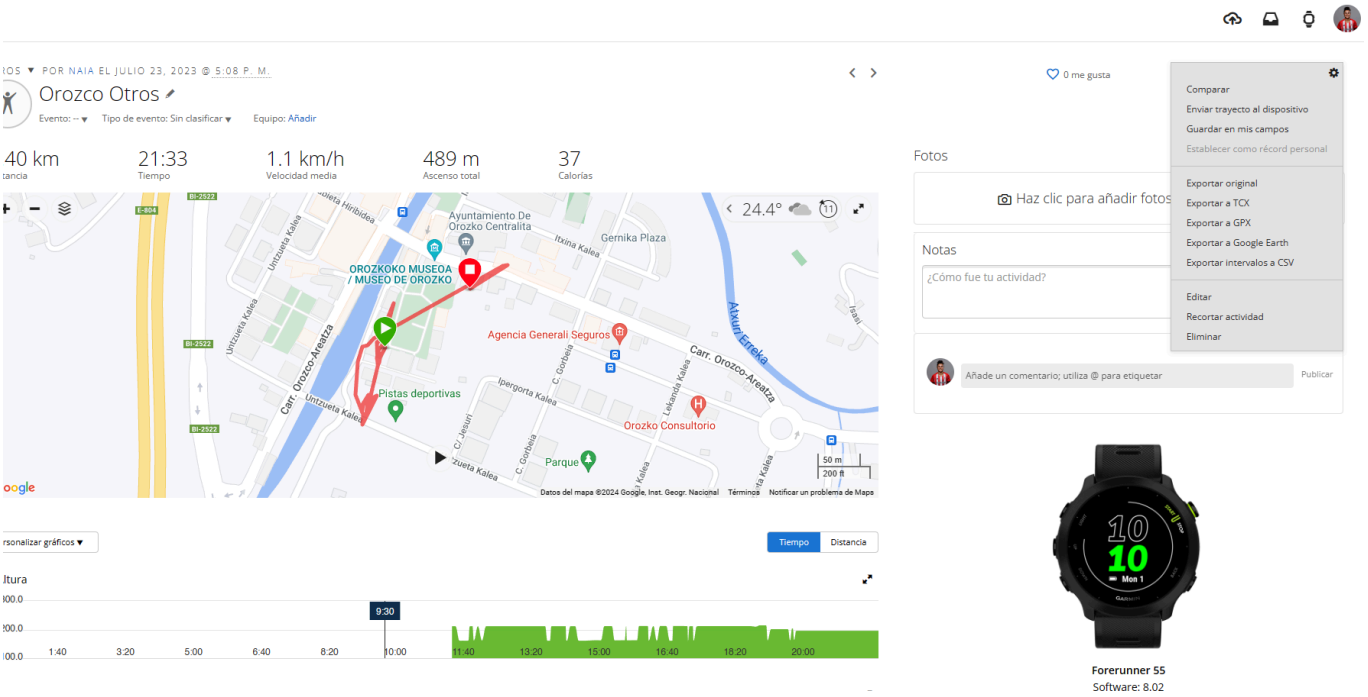
- **Research applications:** Looking for different GPS tracking applications that allow me to export the data in a format that I can manipulate, such as GPX or CSV. In this case, I found different ones like; Strava, Garmin Connect, and Google Maps.
- **Check the features:** In this case, I made sure that the Garmin app allowed me to export my travel data. Some applications offer this directly in the app, while in this case, I had to enter my account to login and download my data.

Step 2: Export Your GPS Tracks

- **Synchronize or save your route**: In my case, I had routes that I have been doing during this year, so it has been easier to search.
- **Export your data**: In the following image you can see how it could be exported for CSV, TCX, GPX, and GOOGLE EARTH.

Step 2: Export Your GPS Tracks

- **Synchronize or save your route:** In my case, I had routes that I have been doing during this year, so it has been easier to search.
- **Export your data:** In the following image you can see how it could be exported for CSV, TCX, GPX, and GOOGLE EARTH.

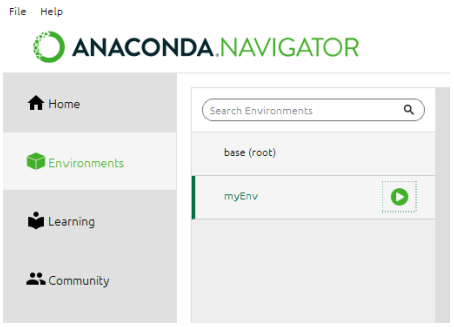


Step 3: Install the Folium and gpxpy Library

- Installing the folium library, which allows creating interactive maps. What I have done was to use my Anaconda environment and incorporate the following installation there to make it easier to execute.

```
pip install folium
```

```
pip install gpxpy
```



Step 4: Parse the GPX Files

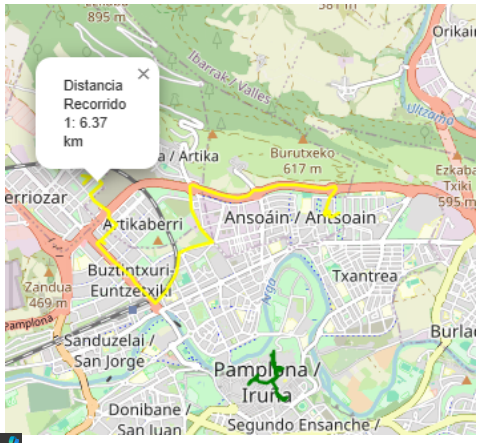
- I used gpxpy to read and parse the GPX files and extract the latitude and longitude coordinates.

Step 5: Create the Map with Folium

- I created a map using folium and added the eight GPX routes as separate lines on the map.

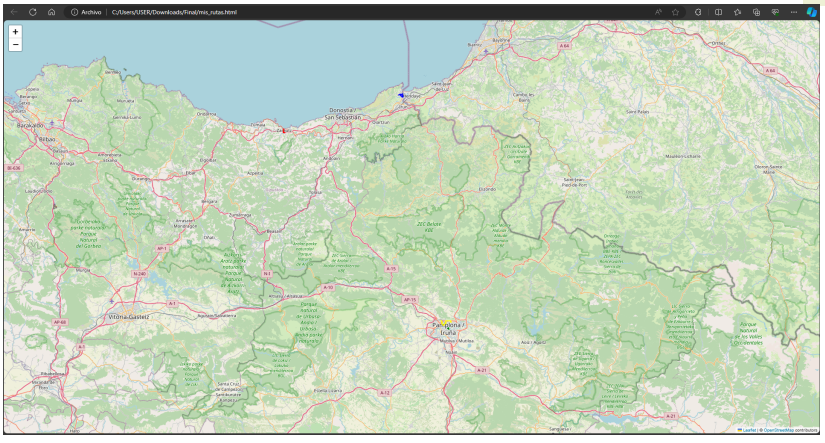
Step 6: Calculate the Distance of the Routes

- I had to add a function to the script to calculate the total distance of the route. Then calculating the distance for each set of coordinates, and finally, I added them as popups on the map. That way, when you visualize the HTML, if you click on the route, it tells you how many kilometers I have done on that route.



Step 7: Save and View the Map

- Saving the map as an HTML file and thus opening it in the browser so that you can see all the routes. In the following images, you have the process to follow so that it takes you to the HTML and there you can play with the different routes that you can find in the same HTML and see the distance of each one:



With this code, you enter the HTML and it takes you to all my routes.

```
>start mis_rutas.html
```

FINAL PROJECT PART TWO:

After getting information about some routes and being able to create an HTML with all of them together.

My idea from the beginning was to use these sensors Bruzzer, Red Led Button, Grove - 16 x 2 LCD, Humidity, and the pulse meter:

Functions of Each Sensor:

- **Pulse meter:** That the screen shows the pulse rate of the person using the smartwatch.
- **Humidity:** With this sensor, what I want is that before going out to run, they had the temperature and humidity at that moment so that they could decide if it was a good idea to go out at that time or not.
- **Bruzzer:** With this sensor, I wanted that when the pulses exceeded calculating that each person had their own range of pulses, when they were high it would sound to warn the person that they are with very high pulses.

RASPBERRY PI:

example1.py

- In this case, we find this .py file that prints on the screen the temperature and humidity. Since I wanted to put that by pressing the button, it would give the current time, but I don't know why I have had many problems with the button. I have not managed to make it work. I have tried it in all ways, so that they could change what they see on the screen for all the sensors I had put. Although as you see to get something I put first the kilometers of the route then the local time and finally the humidity and temperature.

example.py

- In this case, it is the one that I would have loved to finish, read_humidity_temperature(), read_pulses(), route information, button_pressed()... With all that information get a CSV that I could incorporate into a dashboard and from there get the information in graphs about all this information. Every night from 10 o'clock put at least 10 minutes this idea and know which days would be good to go running at those hours depending on all those factors. But it has been impossible for me to use all those sensors.

rutas.py

- The last idea to try to create something taking into account the created HTML was to directly display the routes on the screen. To at least be able to link them in some way.

PROBLEMS:

- **First Problem:**

  My Raspberry Pi would shut down every so often, which meant everything I was doing on the screen would restart and not save. Later, I got into the habit of saving absolutely everything, but each time it shut down, I had to go through the

whole process of connecting it again. I think the power cable might not be working well, you might want to check it.

- **Second Problem:**

  The red button, which seems so useful, was giving me a lot of trouble and wouldn't work, despite trying everything. I installed different imports to try, followed your instructions as well as the same ones I found on the internet, but it was impossible to get it to work.

- **Third Problem:**

  I couldn't display the pulse rate on the screen, nor the kilometers from the HTML.

Final Picture: