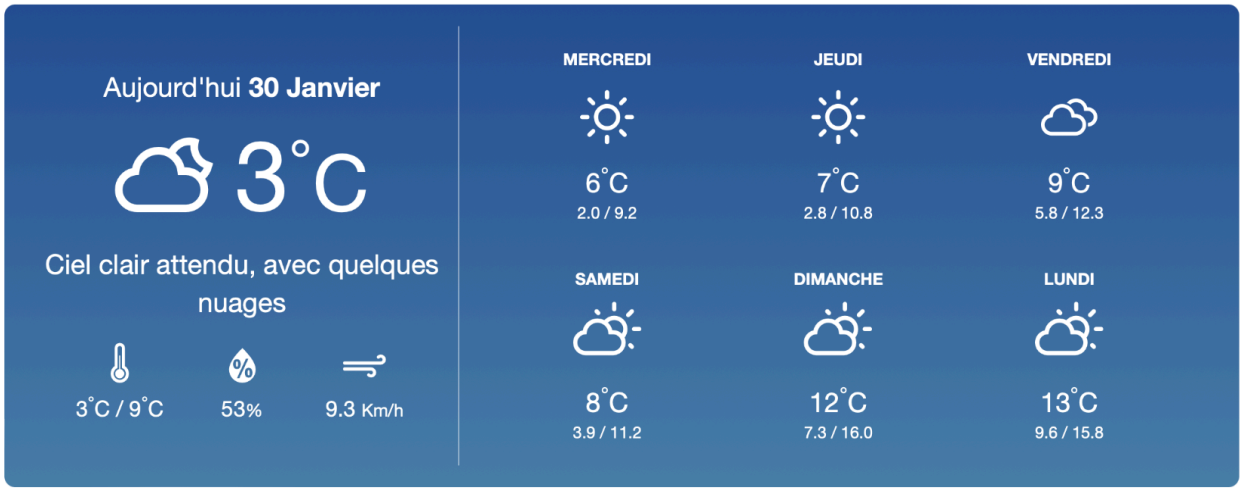
Project - Cloud and Advanced Analytics 2024



Weather affects our mood, our clothing, our interactions with people… even the stock market!

In this project, you will implement an indoor/outdoor weather monitor and dashboard using M5stack IoT devices and sensors. We will supply you with these devices and components, which must all be integrated in your solution.

M5stack Core2 IoT device	ENVIII sensor (humidity & temperature)	Air quality sensor	Motion sensor
			

You will gather both indoor measurements from the sensors and outdoor weather data from your location (a variable) using existing APIs and you will deploy this functionality on the cloud. Your interface on the M5stack IoT device should display:

a) outdoor temperature
b) indoor temperature and humidity
c) indoor air quality
d) Time and date (using NTP)
e) Weather forecast for upcoming days (*only this doesn't need to be stored in the cloud*)

You will also use the presence sensor to detect human activity and announce updates or critical information (e.g., very low air-humidity, storm coming up, take an umbrella, etc). All collected data will be stored in Google Cloud's BigQuery.

In addition to the interface on the device, you are required to develop a cloud-based dashboard using streamlit, so that you monitor the conditions of your house from a distance. This user interface/dashboard will retrieve the data already sent to BigQuery. Therefore, you will have two interfaces, one on the M5stack (home) device and one on the cloud. These can be different, since on the streamlit (cloud) interface you mostly care about accessing not only the current data but also historical data.

Your solution is expected to use the following services and have these functionalities:

1. Big query for storing both the inside temperature/humidity, and the historical weather information. Issue alerts on the local dashboard (M5stack) if the indoor humidity falls below 40% or if the indoor air quality is bad.
2. For outside weather you will use the openweathermap API. You should also use icons for the various weather conditions. Check in GitHub for various projects using Python and openweathermap, so that you don't have to implement this part from scratch.
3. Text-2-speech API from google, openAI or googleLLM (when presence is detected, announce the weather, but only one every hour, for example). In the morning if it is going to rain, remind me to take an umbrella, etc. Make sure you create several such actions.
4. The UI (both on device and on the cloud) should display in a simple format the historical indoor temp, humidity, air quality, etc.
5. It is important to consider that the device may be turned off. Upon turning on, the device should sync the last x values from BigQuery, and the device should display those values. This is something that we will test in the evaluation of the project, so make sure you have some values in your BigQuery table!

User Interface

To understand how to make a usable and easy UI/UX experience especially on the small screen estate of the IoT device, examine existing weather monitors. Analyze how they present the information. Below are a few examples to draw inspiration from, but ultimately, you must devise your own interface.

You can also consult the Teaching Assistants for the course "Interaction Design". They are aware of this and they are waiting eagerly to help you!

Code Quality - IMPORTANT!:

For all projects, the focus should be on clean and reusable code. Imagine that someone else other than you has to deploy your project. Generally, we expect a 3-tier architecture:

- data (bigquery)
- middleware/services (adding, retrieving data and logic)
- UI (streamlit)

These should/can be developed independently after the communication protocols, i.e., APIs, between them have been established.

Code length expectation:

- Your total code for the project is expected to be **~1000 lines** (i.e., 500 per person). This should include the middleware in Flask, and the streamlit visual dashboard. Try to push it to 1000 lines!

Evaluation criteria for code and program stability

- Is it clear how to deploy the application?
- Have you abstracted the project variables (e.g., Wifi username/pass, local zip code, google cloud end-points) so that if someone wants to redeploy the application and services it should be easy?
- If everything is on the same file and not decomposed in meaningful packages, points will be deducted.
- If the device is disconnected from the internet or switched off, what happens when you turn the device on? This is something that you should also consider and we will test it during your presentation.
- Is your program stable? Have you tested all the corner cases?

DELIVERABLES:

1. **VIDEO** (duration 3-10mins) in which you showcase the application: how to use it, which services you used etc. We don't expect you to show the code. Just your solution. Imagine that this is something that you will post on LinkedIn. It should be exciting and highlight the benefits of your solution. In the end of the video, you can also give the link to your GitHub url so that people can see your code and deploy it.
2. **Github url:** Include your YouTube video link in your git. The readme should provide a general overview of the project and describe what each folder of the git contains.

   On the top of your git, clearly declare the names of the group participants and **exactly what each person did for the project**. This should more or less be reflected in the git commits that we see for your project.
3. Submit a zip of your git in the **Moodle** submission.

EVALUATION CRITERIA FOR FINAL GRADE:

Coding/Deployment (60%)

- Google cloud URL for running application (if this is not working no points will be given)
- Google BigQuery
- Google Text-2-speech
- Google LLM/openAI (for generating realistic answer variations) [bonus points!]
- Clean-code and abstracted variables. Different modules/functionalities should be abstracted in different files, functions or classes.
- Error handling in case of reset or loss of Internet.
- OpenWeatherMap or other weather service
- User Interface. Should be appealing, intuitive and easy to use. Should not be slow.
- Weather forecast for the week, in addition to the current day.
- Appealing 3D printed enclosure for the M5stack and sensors [bonus points!]
- Innovative use of some aspects [bonus points!]. If you need more sensors from the M5stack ecosystem let us know! (e.g., cameras, LORA, etc)

Quality of Git (20%)

- Well-structured (images, link to video, etc)?
- Explains how to deploy the app? Is it easy to redeploy the app?

Quality of Video (20%)

- Is it captivating, exciting, fun? Is the audio clear? (Expected duration of 3-10mins)

And remember, it's the journey that matters, not the destination, so have fun while coding and deploying!

Some good projects that students did the previous years:

- Project time keeping device
- Language Translator
- Alcohol-level detector using C02 sensor
- GPS Logbook for Rowing Club
- Smart Dog-food dispenser