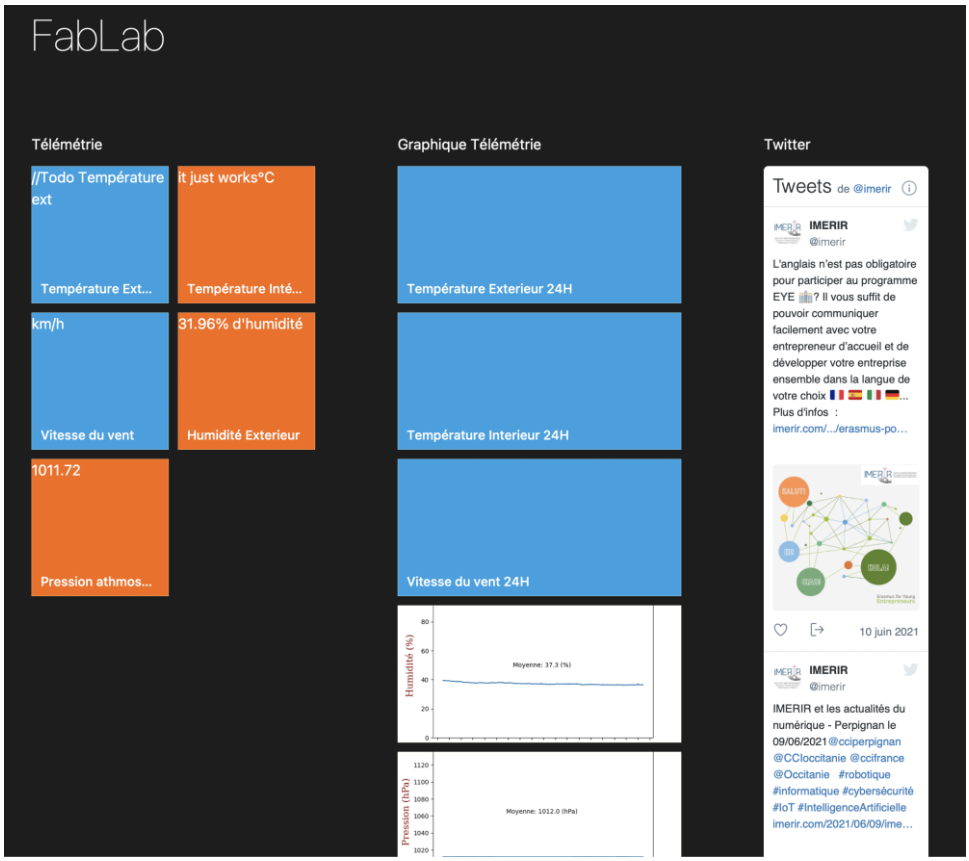
Corentin LEQUEUX
Mikel VIALE
Vini PRAION

INSTITUT MÉDITERRANÉEN
D'ÉTUDES ET DE RECHERCHE EN
INFORMATIQUE ET ROBOTIQUE

Rapport du projet

Ecran



Présenté par

Corentin LEQUEUX

Mikel VIALE

Vini PRAION

Corentin LEQUEUX
Mikel VIALE
Vini PRAION

Corentin LEQUEUX
Mikel VIALE
Vini PRAION

Introduction

Dans le cadre du projet maker de fin d'année, nous avons dû élaborer un projet de groupe en collaboration avec 3 autres groupes.

Notre projet consiste à mettre en place un écran qui affichera en temps réel des données, telles que la température, la force du vent, le taux d'humidité, etc... Que nous récupèrerons en collaboration avec 2 autres groupes, qui devront mettre en place un thermomètre ainsi qu'une station météo. On nous a également donné la tâche d'afficher les derniers tweets de l'IMERIR, ainsi que mettre à disposition un compteur du nombre de personnes présentes au sein du FabLab.

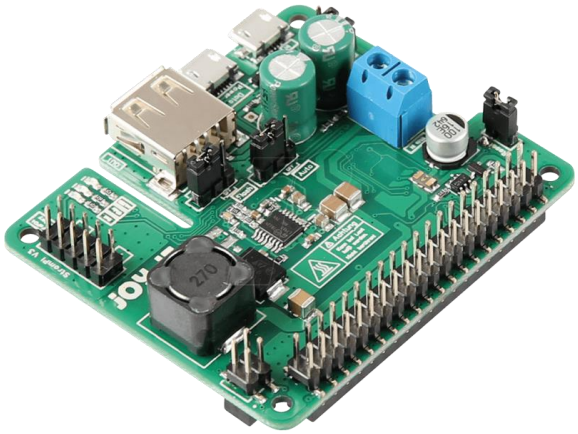
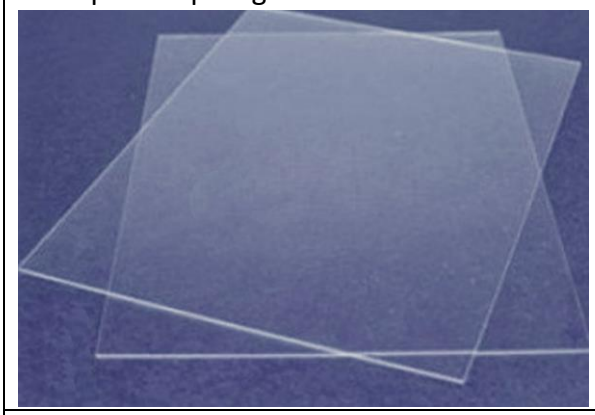
Matériel fourni

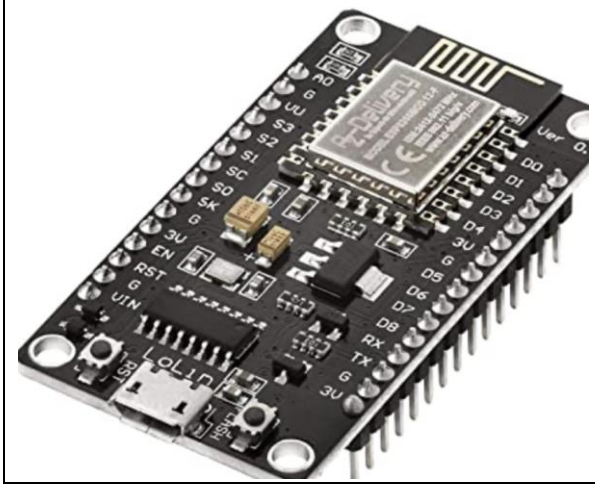
Pour réaliser ce projet, on nous a fourni un Raspberry Pi 3, qui communiquera avec le matériel des 2 groupes (thermomètre et station météo), ainsi qu'un moniteur 5/4 (1280x1024).

Nous avons pu également commander du matériel pour subvenir à nos besoins, qu'il s'agisse de la conception d'un châssis tout en un pour l'écran et le Raspberry, ou encore de compter le nombre de personnes présentes au sein du FabLab. Voici la liste du matériel ainsi que son prix :

Ecran 1280x1024 	Non trouvable neuf, environ 20€ d'occasion

Corentin LEQUEUX
Mikel VIALE
Vini PRAION

Raspberry PI 3 	32€
2 Plaques de plexiglass 	69€
Bombe de peinture 	2€

2 ESP (ESP12F et ESP8266) 	2€

Le matériel fourni et commandé représente un coût total d'environ 125€.

Choix de la technologie

Afin de communiquer de façon sans fil avec les périphériques des 2 autres groupes, nous avons choisi d'utiliser le protocole MQTT (Message Queuing Telemetry Transport). Ce protocole a l'avantage de convenir aux faibles bandes passantes, et dans le cas où nous souhaitons juste envoyer la valeur mesurée du périphérique ainsi que la date de mesure, ça répond parfaitement au besoin. Pour l'affichage, nous afficherons ces valeurs au sein d'une page web.

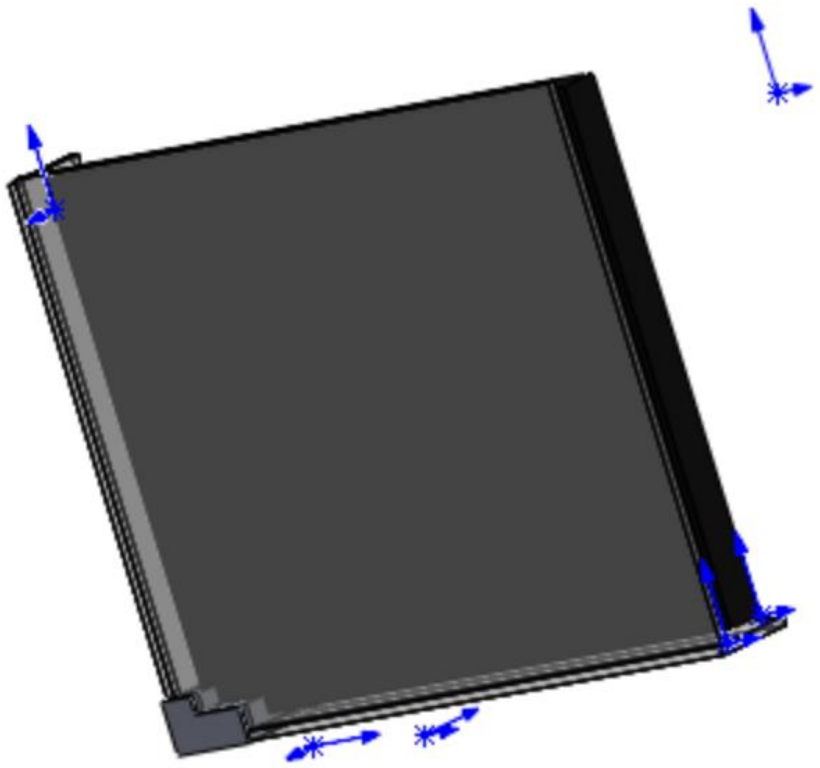
Principe de fonctionnement

Nous traitons la partie communication MQTT entre les périphériques par Python, pour enregistrer les informations données au sein d'un fichier CSV. Deuxièmement, à base d'une certaine quantité de données au sein des CSV, nous générons un fichier image représentant un graphique sur la durée de ces valeurs (température, pression atmosphérique...), toujours par Python et à l'aide de la Library mathplotlib.

Finalement, avec ces fichiers CSV et images générés correspondant à chaque périphérique de mesure, nous les affichons au sein d'une page Web à l'aide de PHP, et de Javascript. La page s'actualisera toutes les minutes afin de venir charger les dernières mesures et graphiques.

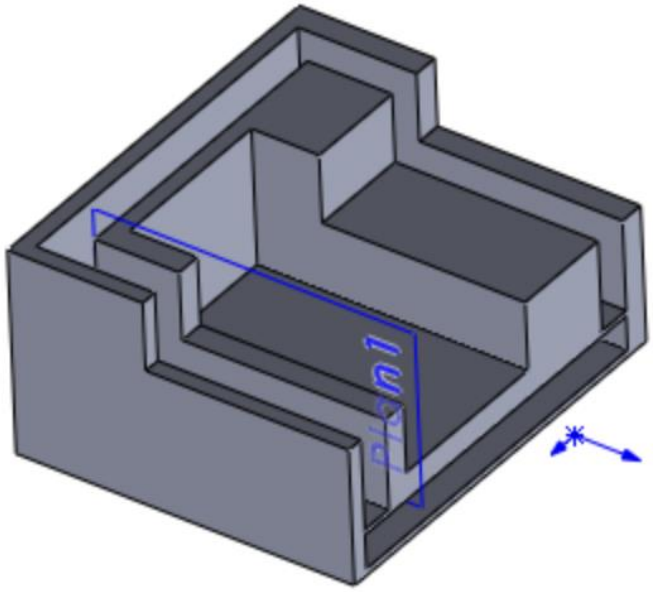
Schématisation et conception du châssis

Afin de concevoir un châssis tout en un pour l'écran et le raspberry, nous avons démonté l'actuel châssis de l'écran. Nous avons par la suite envisagé d'encastrer le moniteur entre 2 plaques de plexiglass, dont voici le prototypage :



Le nouveau châssis, composé de deux plaques et 2 équerres

Corentin LEQUEUX
Mikel VIALE
Vini PRAION



L’équerre qui sera entre les 2 plaques

Le résultat du prototypage 3D

Programmation

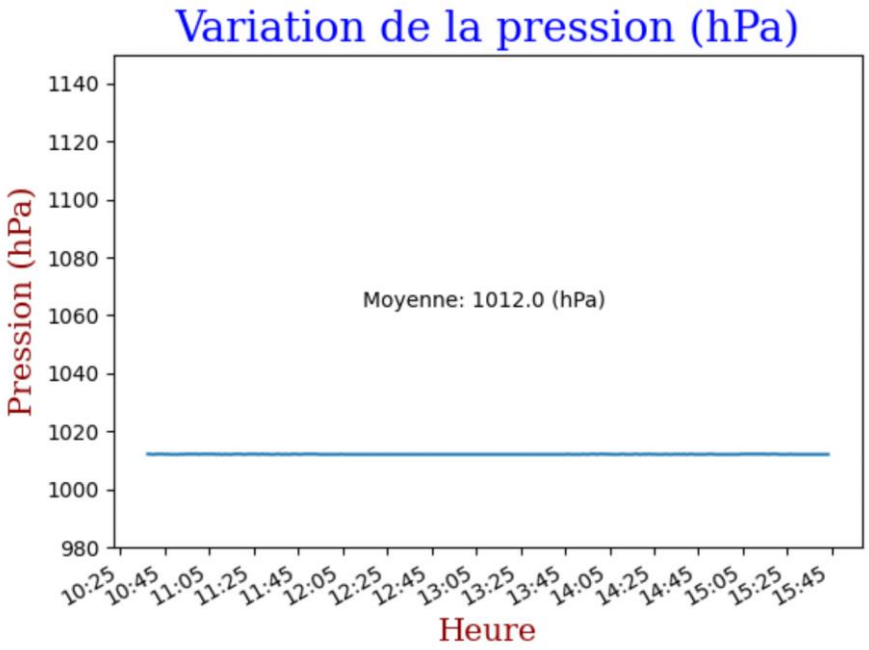
Lors du démarrage du Raspberry, les scripts suivant vont se lancer :

- Traiter le protocole MQTT
- Générer un graphique toutes les 60 secondes.

```python
import logging
import threading
import time
import schedule
import os
import mosquittosub

def mosqui():
        mosquittosub.run()

def rune():
        while True:
                os.system('sudo python /home/pi/Mosquitto/hydrograph.py')
                print("Run Hydro Graph")
                os.system('sudo python /home/pi/Mosquitto/pressiongraph.py')
                print("Run Pression Graph")
                time.sleep(60)

threading.Thread(target=rune).start()
mosqui()
```

Exemple des fichiers CSV générés, où figure un “timestamp” et la valeur de mesure

```
1623399111.05,  26.35
1623399115.4,  26.36
1623399120.33,  26.34
1623399125.53,  26.34
1623399130.55,  26.35
1623399135.45,  26.35
1623399140.34,  26.37
1623399145.35,  26.36
1623399150.5,  26.36
1623399155.47,  26.38
1623399160.37,  26.38
1623399165.37,  26.40
1623399170.56,  26.42
```



Graphique généré issu d'un CSV

Un extrait du code pour le compteur de personnes, qui se base sur 2 capteurs lasers, on émet une impulsion puis on attend de recevoir ce signal, et selon le temps écoulé, on en déduit la distance.

```
int visitor = 0;
String visiteur;

void read_dual_sensors() {

  lox1.rangingTest(&measure1, false); // pass in 'true' to get debug data printout!
  lox2.rangingTest(&measure2, false); // pass in 'true' to get debug data printout!

  distance1 = measure1.RangeMilliMeter;
  distance2 = measure2.RangeMilliMeter;

  if(distance1 < 800 && distance2 > 800) { // si le capteur 1 reagit en premier
    visitor = visitor + 1; // ALORS on ajoute +1 a la variable visitor
    visiteur = String(visitor);
  }
  if(distance1 > 800 && distance2 < 800) { // si le capteur 2 reagit en second
    visitor = visitor - 1; // ALORS on enleve 1 a la variable visitor
    visiteur = String(visitor);
  }
```

Démonstration d'une fonction qui permet d'afficher la dernière valeur au sein d'un CSV, où on lui passe en paramètre le chemin du fichier et le type d'unité à afficher.

Corentin LEQUEUX
Mikel VIALE
Vini PRAION

```php
<?php
function csvLastValue($path, $unitType)
{
    $dateTimeFormat = 'Y-m-d H:i:s';
    $handle = fopen($path, "r");
    $data = fgetcsv($handle);
    $value = $data[count($data) - 1];
    echo ($value . $unitType);
    fclose($handle);
};
?>
```