Situation d'apprentissage et d'évaluation 24 - Projet Intégratif (partie IOM)

Revue de projet faite par Yassine EL HAMIOUI, Thomas Raynaud et Thomas Mirbey.

Introduction

L'acronyme **Saé** est **« Situation d'Apprentissage et d'Evaluation** ». Son objectif est de **valider** les nombreuses compétences acquises au fur et à mesure de l'année dans différentes ressources.

Cette Saé nommée « **Projet Intégratif** », partie **Internet des Objets et Mobilités** (IOM) a débuté le mardi 7 juin à l'IUT Réseaux et Télécommunications de Montbéliard, et c'est fini le mercredi 8 juin 2022.
Cette Situation d'Apprentissage et d'Evaluation nous a permis de **mobiliser les compétences** pour la mise en place de serveur Apache, l'utilisation de PHP et de développement Arduino. Pendant cette Saé, nous avons aussi eu l'occasion de manipuler du matériel réseaux et du matériel nous permettant de mener à bien notre projet dans le domaine de l'Internet des Objets et Mobilités.

Durant ce jour et demi, des **professeurs nous ont accompagnés** afin de nous aiguiller en cas de problèmes, mais aussi pour vérifier les bonnes configurations matérielles et logicielles. Les professeurs étaient Monsieur GIVRON et Monsieur BOUILLET.

SOMMAIRE

1. Matériel mis à notre disposition

Le matériel mis à notre disposition était très utilisé dans le domaine de l'Internet des Objets et Mobilités. Nous pouvons séparer celui-ci en trois catégories.

La **première** catégorie correspondrait aux équipements réseaux. Nous avons eu recours à :

- Un **routeur Linksys**, qui nous sert de serveur DHCP et de passerelle vers le portail captif. Il permet de relier nos ordinateurs sur un même réseau afin de faciliter l'utilisation du serveur Apache qui sera présenté ultérieurement. Pour mettre en place ce routeur, nous devions réaliser un reset de la borne, réaliser les branchements, puis se connecter au **Firmware** du routeur en rentrant l'adresse IP du routeur dans une page web (192.168.1.1). Dans ce *Firmware*, il fallait faire plusieurs configurations comme la mise en place du bon nom ssid, sécuriser le réseau en mode *wpa2 personal*, mettre le mot de passe *tpRT9025*. Puis vérifier le bon fonctionnement en s'y connectant avec nos appareils.
- Des **câbles réseaux** pour réaliser les branchements entre les ordinateurs et le routeur. Et entre routeur et armoire de brassage.
- L'**armoire de brassage** permettant de se connecter au portail captif.

La **deuxième** catégorie serait le matériel de bureautique tel que :

- Les **trois ordinateurs** ;
- Un **adaptateur de carte SD** ;
- Le **matériel de stockage** avec une carte micro-SD de 16go de classe 10 contenant une image de type BUSTER. Pour mettre l'image ISO sur la carte SD, nous avons utilisé le logiciel Rufus qui nous est assez familier ;
- Un **Raspberry Pi 4** utilisant la carte SD citée précédemment ;
- Une **alimentation** USB-C 5V pour le Raspberry Pi ;
- Un **écran** ainsi que le **câble HDMI** vers micro HDMI, pour voir l'interface graphique du Raspberry Pi.

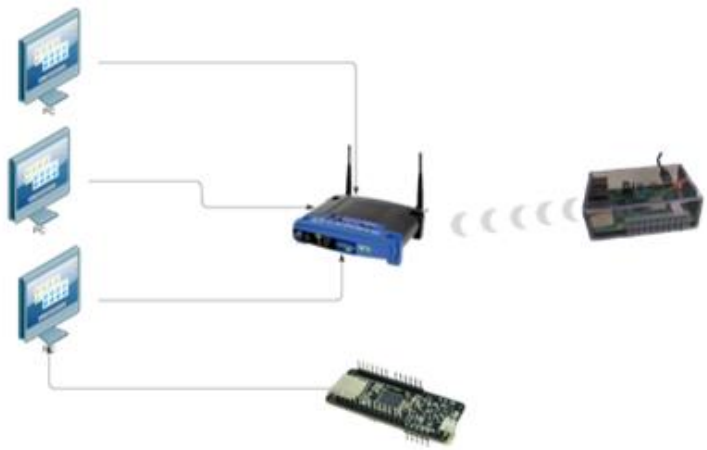
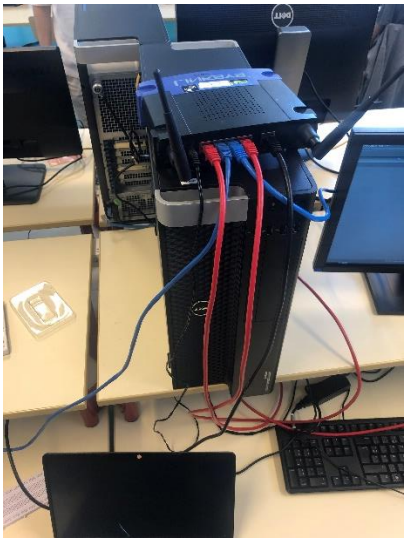


Figure 1 : Topologie physique du réseau mis en place.



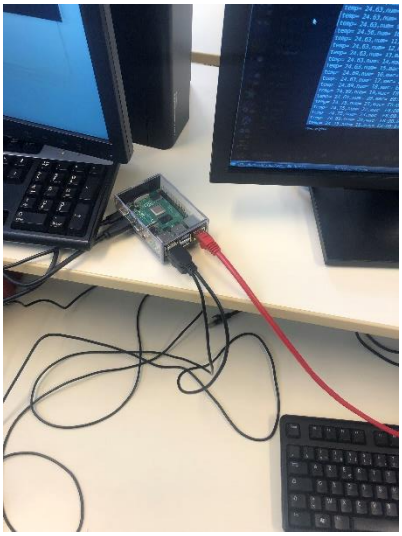


Figure 2 et 3 : A gauche branchements au niveau du routeur Linksys ; à droite branchements du Raspberry Pi

Enfin, la **troisième** catégorie correspondrait aux équipements électroniques couramment utilisés dans l'Internet des Objets et Mobilités, tels que :

- Un **microcontrôleur ESP8266** ;
- **Cable d'alimentation** USB-microUSB pour le microcontrôleur ;
- Une **plaque de prototypage** ;
- **Fils Dupond** ;
- Une **résistance** de 4,7kOhms ;
- Un **capteur de température** OneWire DS18B20.

Figure 3 : Montage du microcontrôleur ESP8266 pour capturer la température.

2. Organisation du projet

- Afin de mener à bien ce projet, nous avons utilisé plusieurs **outils d'organisation** :
  - Dans un premier temps, **Discord**. Grâce à cet outil, nous avons pu créer un serveur de discussion permettant le travail collaboratif et la répartition des tâches.
    Dans ce serveur, plusieurs salons ont été créés. Comme vous pouvez le voir dans l'image ci-dessous, nous avons réparti les jalons par numéros. Dans ceux-ci, nous avons mis les différentes traces afin de ne pas les perdre et de pouvoir y accéder facilement, pour éviter un maximum de retourner en arrière dans notre travail.

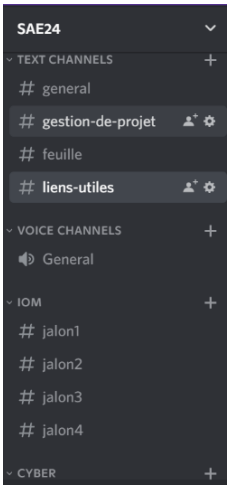


Figure 4 : Différents salons textuels et vocaux présent dans notre serveur Discord.

  - Nous avons également utilisé **Paint**, ce qui nous a permis de faire une répartition claire des tâches en associant à chacun d'entre nous une couleur, et surlignant les tâches sur la carte mentale. Comme vous pouvez le voir ci-dessous, nous avons pu avancer indépendamment les uns des autres pour rester efficace.

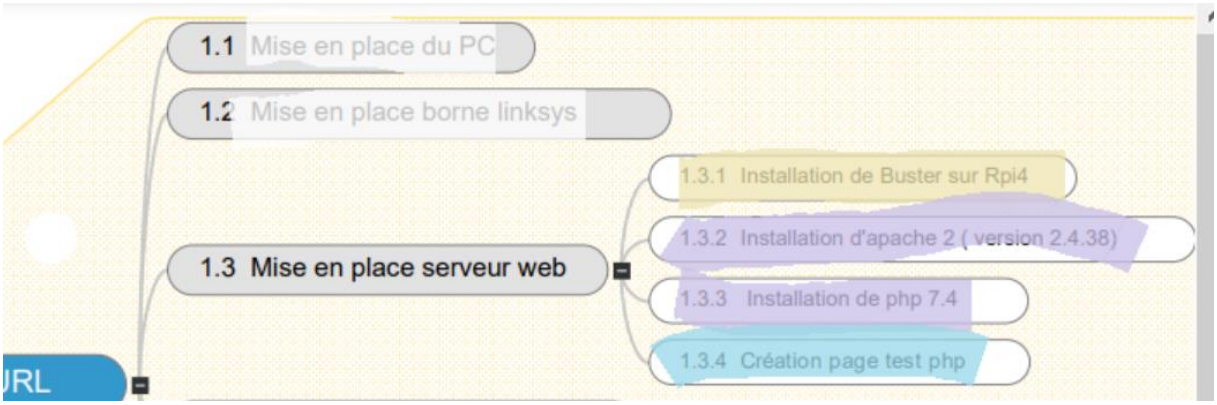


Figure 5 : Répartition des tâches grâce à Paint

Figure 6 : Légende de la figure 5 (sous discord)

- Nous avons aussi utilisé **Notion**, qui est un logiciel très adapté pour la gestion de projet et que nous utilisons très régulièrement. Avec ce logiciel, nous sommes en mesure d'actualiser l'état d'avancement du travail restant et y mettant des informations supplémentaires. Comme vous pouvez le voir, pour **chaque jalon** nous y renseignons :
  - Son état d'avancement ;
  - Quelles sont les personnes ayant réalisées la pratique ;
  - Heures de début/fin ;
  - Qui a rendu le jalon ;
  - Le rendu final est-il disponible sur Discord ;
  - Les difficultés rencontrées ;
  - Les différents rendus à faire ;
  - D'éventuelles photos afin d'alimenter le rapport.

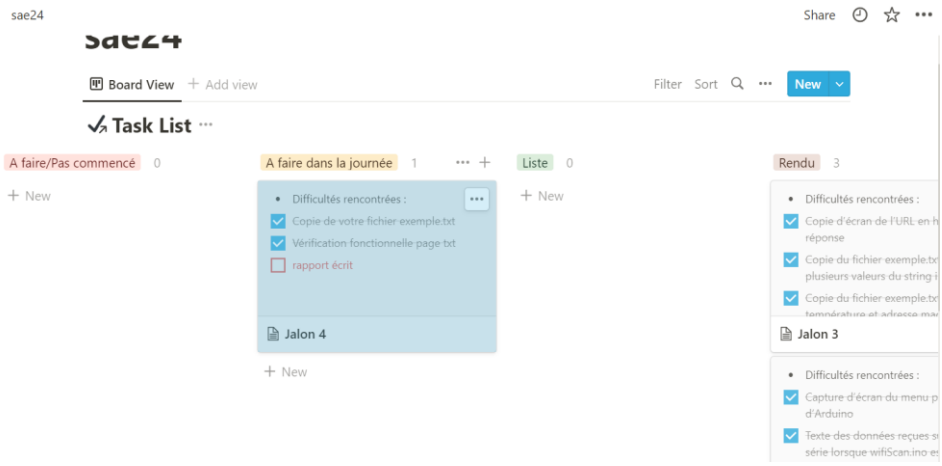


Figure 7 :

⇨ Dans notre groupe, **l'organisation** et la **répartition des tâches** a été très importante. Avancer indépendamment les uns des autres, en demandant de l'aide si besoin nous a permis d'obtenir un gain de temps conséquent dans un projet qui serait difficilement réalisable sans répartition équitable des tâches.

- Nous avons utilisé **Rufus**. Ce logiciel nous a permis de facilement créer des clés bootables sur les cartes SD qui étaient à notre disposition. L'utilisation de ce logiciel nous a permis un gain de temps considérable sachant que nous l'avions déjà utilisé auparavant.

3. Services mis en place

Pour pouvoir exploiter pleinement notre réseau, nous avons pu mettre en place plusieurs services, comme :

- **Service Secure Shell** aussi appelé **SSH** sur le Raspberry Pi. Cela nous permet de nous connecter à distance à ce dernier. En nous appuyant sur cet outil, nous avons pu optimiser notre travail, car plusieurs machines pouvaient se connecter au Raspberry Pi pour pouvoir travail en parallèle. L'avantage de ce protocole est qu'il est sécurisé, la communication est totalement cryptée par un algorithme de **chiffrement asymétrique** RSA (Rivest Shamir Adleman).
  Pour configurer SSH, il faut l'activer sur la machine (cliente).

```
tp@rt:~$ ssh tp@192.168.1.117
tp@192.168.1.117's password:
Linux rt 4.19.0-20-amd64 #1 SMP Debian 4.19.235-1 (2022-03-17) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jun  8 12:10:44 2022 from 192.168.1.108
```

Figure 8 : Connexion SSH au Raspberry Pi depuis un ordinateur fixe

- **Apache2** permet de créer serveur web. Pour le mettre en place, il fallait suivre plusieurs commandes renseignées dans le document « Fiche ressource 2 : TP_fiche4_mise_en_place_serveur_web_.pdf ».
  Les **étapes majeures** étaient :
  - L'installation des paquages « apache2 » ;
  - La gestion les droits d'accès au dossier d'apache, ce qui facilite l'administration des sites que nous vous présenterons dans la suite de ce rapport ;
  - La vérification du bon fonctionnement du serveur au niveau de l'interface graphique en se connectant à l'adresse localhost 127.0.0.1. On obtient le message *It works* ;
  - La vérification du bon fonctionnement du serveur sans interface graphique, nous avons pu récupérer le contenu des différentes pages avec la commande wget.

Figure 9 : Message « It works ! »

- **PHP version 7.4**. : il faut installer PHP, c'est un langage qui s'exécute sur le serveur. Sans son téléchargement, il serait impossible d'exécuter les programmes que nous avons réalisé. Nous avons pu vérifier sa version :

```
root@raspberrypi:/home/tp/Desktop# php -v
PHP 7.4.28 (cli) (built: Feb 17 2022 16:17:19) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.28, Copyright (c), by Zend Technologies
```

Figure 10 : Version de php 7.4.28

4. Utilisation de PHP et Arduino

Pour différents jalons, nous devions être capables de **comprendre** et de **réaliser** différents programmes.

- **Au niveau du PHP** :
  - Tout d'abord, nous avons créé une première page (**phpinfo.php**) permettant d'afficher toutes les informations concernant le serveur et les versions de PHP. Ces valeurs peuvent être très utiles pour s'assurer que le serveur fonctionne correctement mais peuvent être très dangereuses si un attaquant arrive à se les procurer.
    En effet, celui-ci pourrait être au courant de toutes les **failles** présentes sur la machine. Il est donc nécessaire de ne pas laisser cette page sur un serveur web une fois mis en production.

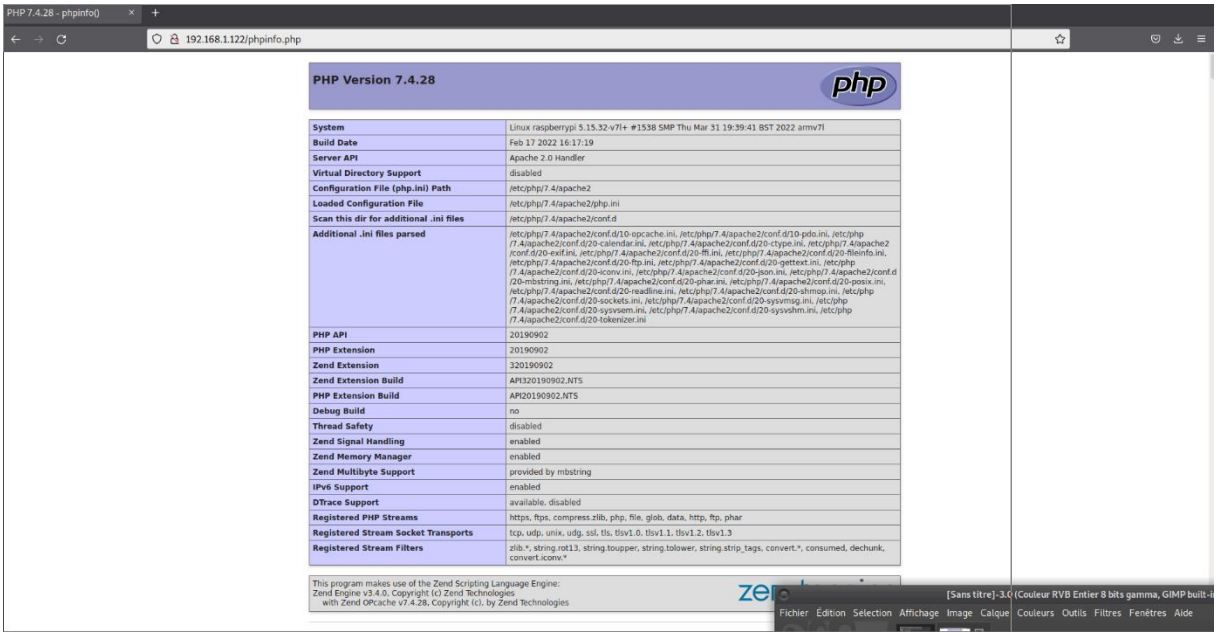


Figure 11 : Page PHPinfo

  - Nous avons ensuite créé une page intitulée **echo.php** permettant de récupérer des valeurs fournies par l'utilisateur dans un URL. Nous avons fait **évoluer** cette page de nombreuses fois au cours de cette SAé. Cette page permettait également de récupérer des valeurs envoyées pour les différents programmes Arduino qui seront présentés ci-dessous.

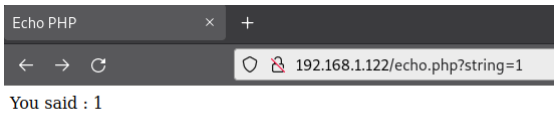


Figure 12 : Récupère la valeur dans l'url

- Nous avons réalisé la page **echo2.php** qui enregistrait les valeurs entrées dans l'url dans un fichier .txt (chez nous **exemple1.txt**). A la différence de la page précédente, les valeurs étaient stockées en plus d'être affichées. Dans la figure 16, on peut donc deviner que dans le fichier exemple1.txt, la valeur « 1 » serait ajoutée.

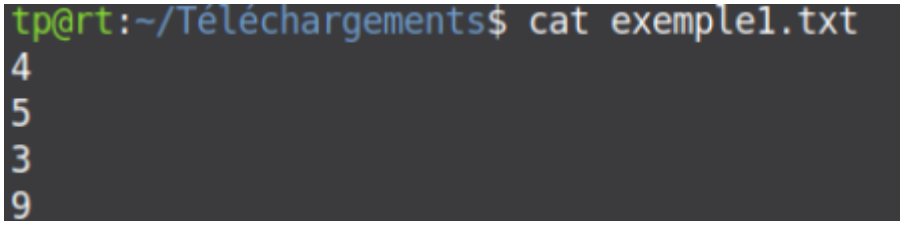


Figure 13 : Stockages des valeurs dans exemple1.txt

- Finalement, nous avons mis en place une page **echo_parametres.php** qui prenait en compte plusieurs valeurs renseignées dans l'url et qui les stockait dans un fichier .txt. Nous avons récupéré **l'adresse MAC** de la source ainsi qu'un **numéro de requête**. Nous vous en exposerons l'intérêt dans la partie concernant Arduino. Nous avons également stocké la température récupérée par un capteur.

➢ **Au niveau d'Arduino** :

- Tout d'abord, nous avons lancé le fichier **wifiScan.ino** afin d'observer l'ensemble des réseaux sans fils accessibles par notre ESP. Nous avons récupéré le résultat dans un fichier texte.

```
05: [CH 01] [A4:BE:2B:C0:0F:23] -79dBm * V ufc-vpn-wpa
06: [CH 01] [62:01:94:0F:1E:B6] -61dBm * V mariline
07: [CH 01] [A4:BE:2B:BF:F9:A1] -76dBm * V ufc-edu-portail
08: [CH 01] [C0:56:27:1B:02:C5] -65dBm * V binome_12
09: [CH 01] [A4:BE:2B:BF:F9:A2] -76dBm * V ufc-personnels-wpa
10: [CH 01] [A4:BE:2B:BF:F9:A3] -76dBm * V ufc-vpn-wpa
11: [CH 01] [A4:BE:2B:BF:F9:A0] -76dBm * V eduroam
12: [CH 06] [A4:BE:2B:BF:1B:E9] -61dBm * V ufc-wifi-invites
13: [CH 06] [A4:BE:2B:BF:1B:E0] -61dBm * V eduroam
14: [CH 06] [00:1E:E5:56:CB:4D] -57dBm * V binome_4
15: [CH 06] [A4:BE:2B:BF:1B:E1] -60dBm * V ufc-edu-portail
16: [CH 06] [A4:BE:2B:BF:1B:E2] -60dBm * V ufc-personnels-wpa
17: [CH 06] [A4:BE:2B:BF:1B:E3] -61dBm * V ufc-vpn-wpa
18: [CH 06] [C0:56:27:19:B5:21] -65dBm * V binome_6
19: [CH 06] [C0:56:27:1A:FE:3F] -82dBm * V dd-wrt
20: [CH 06] [C0:56:27:19:B3:F2] -51dBm * V binome_2
21: [CH 06] [C0:56:27:19:B4:DC] -52dBm * V binome_3
22: [CH 06] [C0:56:27:19:B3:EF] -48dBm * V trinome_1
23: [CH 06] [00:1E:E5:56:F8:7A] -74dBm * V binome_12
24: [CH 06] [C0:56:27:19:B3:B0] -60dBm * V binome_8
25: [CH 06] [00:1C:10:A4:51:04] -65dBm * V dd-wrt
26: [CH 06] [00:1C:10:A4:44:E9] -81dBm * V binome_10
27: [CH 06] [00:22:6B:48:98:7F] -77dBm * V binome_9
28: [CH 06] [C0:56:27:19:B6:A4] -87dBm * V binome_14
29: [CH 06] [C0:56:27:19:B3:CE] -75dBm * V binome_10
30: [CH 01] [A4:BE:2B:BF:F9:A9] -77dBm * V ufc-wifi-invites
31: [CH 10] [E6:0E:EE:09:21:C2] -63dBm * V la sainte connexion internet
32: [CH 11] [DA:4E:C5:AA:AE:8C] -86dBm * V Galaxy S10055f
33: [CH 11] [2E:6B:26:A2:AF:76] -65dBm * V Rogue Tah Phone
34: [CH 11] [4A:2C:CA:BE:1E:28] -84dBm * V Val
35: [CH 11] [A4:BE:2B:BE:9C:A0] -86dBm * V eduroam
36: [CH 11] [A4:BE:2B:BE:9C:A1] -88dBm * V ufc-edu-portail
37: [CH 11] [A4:BE:2B:BE:9C:A2] -87dBm * V ufc-personnels-wpa
38: [CH 11] [A4:BE:2B:BE:9C:A3] -88dBm * V ufc-vpn-wpa
39: [CH 11] [A4:BE:2B:BE:9C:A9] -87dBm * V ufc-wifi-invites
40: [CH 12] [C0:56:27:19:B4:C7] -58dBm * V binome_5
```

Figure 14 : Scan des réseaux capté par notre microcontrôleur ESP (grâce à wifiScan.ino)

- Nous avons importé le programme **wifiClientBasic.ino** depuis les menus Exemple => ESP8266Wifi=>**wifiClientBasic**.
  Nous avons dû changer certains paramètres afin d'adapter le programme à notre environnement de travail. Nous avons modifié le nom du SSID et le mot de passe de notre wifi afin de correspondre avec celui de notre routeur. Par la suite, nous avons également supprimé le contenu de la partie loop du programme ainsi que toutes les variables inutilisées. Finalement, nous avons ajusté le débit d'envoi des données sur le port série, nous avons fixé cette valeur à 115200 kbits/s au niveau du code et de la fenêtre terminale. Nous pouvons connecter notre ESP au réseau et voir **l'adresse IP qui lui est attribuée**.

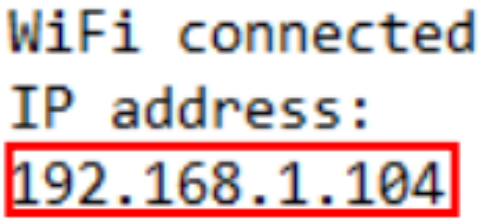


Figure 15 : Attribution d'un adresse IP à notre microcontrôleur ESP

- Nous avons ensuite utilisé un programme (**BasicHttpClient.ino**) permettant d'envoyer des requêtes GET à un serveur web pour récupérer des pages HTTP depuis notre microcontrôleur ESP. Cela peut être **utile** notamment pour vérifier l'état d'un serveur distant. Nous obtenions :

```
[SETUP] WAIT 4...
[SETUP] WAIT 3...
[SETUP] WAIT 2...
[SETUP] WAIT 1...
[HTTP] begin...
[HTTP] GET...
[HTTP] GET... code: 200
<!DOCTYPE html>
<html lang="fr">
<head>
        <title>Echo PHP</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
    <body>

    <!--DÉBUT affichage PHP -->

    You said :1
    <!--FIN affichage PHP -->

    </body>
</html>
```

Figure 16 : Affichage dans le terminal Arduino avec le programme BasicHttpClient.ino

- Nous avons utilisé un programme similaire (**BasicHttpsClient.ino**) nous permettant d'envoyer des requêtes GET à un serveur web pour **récupérer des pages HTTPS**. Ces pages utilisent un certificat permettant de sécuriser les données. Il était donc nécessaire de renseigner le **fingerprint** du certificat dans notre programme. Le fingerprint est un hash du certificat. Dans notre cas, le hash était l'empreinte de l'algorithme de **cryptage SHA-1**.

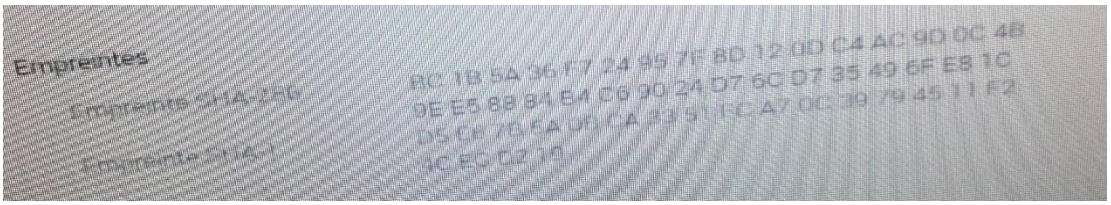


Figure 17 : Le fingerprint de l'algorithme de cryptage SHA-1

- Nous avons réalisé une version plus avancée du programme **BasicHttpsClient.ino** en ajoutant divers champs utiles comme la température détectée par le capteur, le numéro de la requête est très utile pour permettre de différencier les différentes requêtes envoyées. L'adresse MAC est complémentaire puisqu'elle permet de différencier la source de la requête. Combinées, ces deux valeurs permettent d'identifier chaque requête et sa source. Cela peut être utile lors de l'utilisation de plusieurs sondes météo. Ces requêtes étaient destinées à une page web qui stockait ces valeurs dans un fichier .txt. C'est pour cela qu'après la validation de notre programme avec des champs statique nous sommes passés aux champs variables en ajoutant le numéro de requête qui sont intéressant pour avoir une **chronologie** dans nos mesures.

```
[HTTPS] GET...
[HTTPS] GET... code: 200
<!DOCTYPE html>
<html lang="fr">
<head>
        <title>Echo PHP</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
    <body>
    <p>Adresse MAC comprise</p>
        La temperature correspondante a la requete 1 est de 32, la mac source est  E8:DB:84:95:DF:49


    </body>
</html>

Wait 10s before next round...
ESP Board MAC Address: E8:DB:84:95:DF:49[HTTPS] begin...
[HTTPS] GET...
[HTTPS] GET... code: 200
<!DOCTYPE html>
<html lang="fr">
<head>
        <title>Echo PHP</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
    <body>
    <p>Adresse MAC comprise</p>
        La temperature correspondante a la requete 2 est de 32, la mac source est  E8:DB:84:95:DF:49


    </body>
</html>

Wait 10s before next round...
```

Figure 18 : Affichage dans le terminal Arduino avec le programme BasicHttpClient.ino avec plusieurs arguments

- Par la suite nous y avions aussi ajouté un autre **champ variable** qui est la température. Nous présentons cette variable plus en détail dans la partie suivante.

⇨ Pour pouvoir extraire toutes les valeurs utiles, nous avons dû installer des **libraires Arduino** que nous avons importé dans nos programmes.

5. Mesures de température

Afin de mesurer les températures, il fallait d'abord mettre en place un **montage spécifique** au niveau du microcontrôleur :

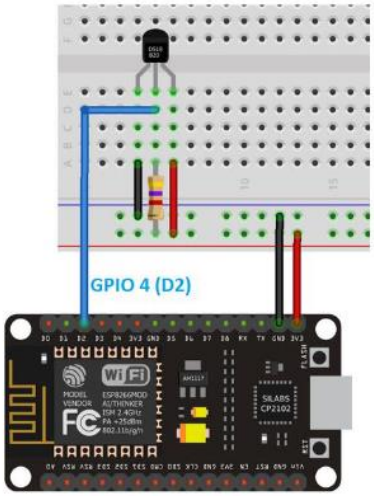


Figure 19 : Montage du microcontrôleur et de la sonde à température

En réalité, les branchements ressemblaient à :

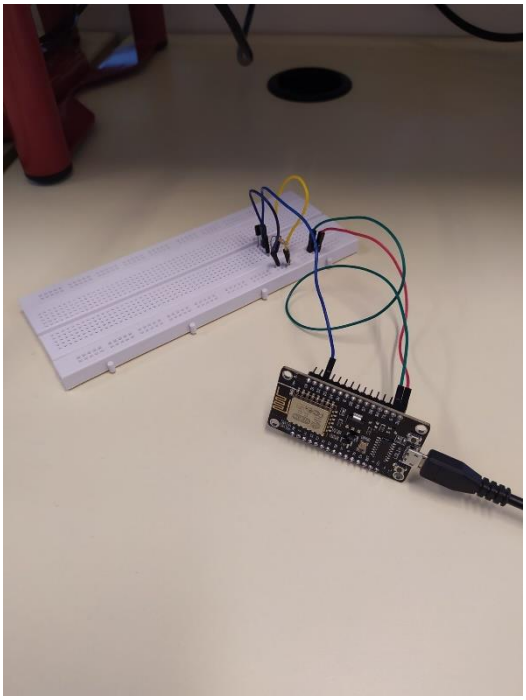


Figure 20 : Montage physique

Il fallait aussi penser à installer les **librairies** afin d'exploiter notre capteur de température « one wire ».

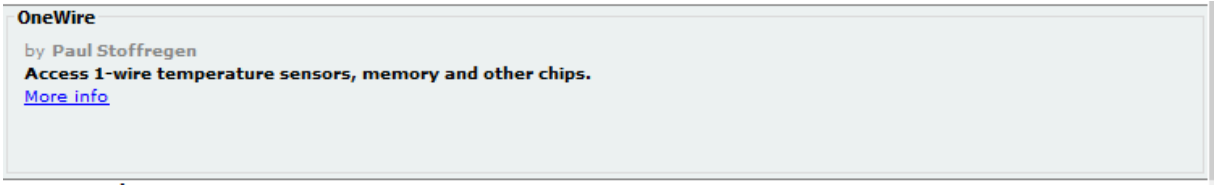


Figure 21 : Différentes librairies à installer

Puis nous avons récupéré le programme Arduino présent sur Moodle, nous permettant de vérifier la **bonne récupération** des mesures de température. Après la validation des températures captées, nous avons alors pu le combiner avec le programme qui renseigne des informations dans l'URL se nommant **BasicHttpsClient.ino**. Nous avons alors **deux valeurs dynamiques** : le numéro de requête et la température.
Nous avons pu vérifier si le fichier **exemple.txt** se remplissait bien, sur le serveur, cela donnait :

```
tp@rt:~/Bureau$ cat exemple3.txt
temp= 24.56,num= 3,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 4,mac= E8:DB:84:95:DF:49
temp= 24.56,num= 5,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 6,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 7,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 8,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 9,mac= E8:DB:84:95:DF:49
temp= 24.56,num= 10,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 11,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 12,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 13,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 14,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 15,mac= E8:DB:84:95:DF:49
temp= 24.69,num= 16,mac= E8:DB:84:95:DF:49
temp= 24.63,num= 17,mac= E8:DB:84:95:DF:49
temp= 24.69,num= 18,mac= E8:DB:84:95:DF:49
temp= 24.69,num= 19,mac= E8:DB:84:95:DF:49
temp= 24.69,num= 20,mac= E8:DB:84:95:DF:49
temp= 24.75,num= 21,mac= E8:DB:84:95:DF:49
temp= 24.75,num= 22,mac= E8:DB:84:95:DF:49
temp= 24.75,num= 23,mac= E8:DB:84:95:DF:49
temp= 24.69,num= 24,mac= E8:DB:84:95:DF:49
temp= 24.75,num= 25,mac= E8:DB:84:95:DF:49
```

Figure 22 : Fichier exemple.txt qui se remplit selon les informations du code Arduino

6. Sécurisation

Au cours de cette SAé, nous avons pu nous rendre compte que la sécurité occupe une place plus importante dans **l'Internet des Objets** que nous ne le pensions.
En effet, une des problématiques importantes de **l'IOT** (Internet des Objets) est de sécuriser les appareils et leurs données afin que celles-ci ne soient pas interceptées par des individus malveillants.
Au cours de cette SAé, nous avons pu nous rendre compte de plusieurs failles de sécurité.
Premièrement, malgré le cryptage des données sur un site en **HTTPS**, celles-ci naviguent en clair dans l'url lorsque l'on utilise la méthode **GET**. Cela pose un problème en termes de **confidentialité** et **d'intégrité des données**. Une attaque du type « **Man in the middle** » pourrait compromettre ces données, un utilisateur pourrait récupérer les données ou bien servir d'intermédiaire en modifiant les données présentes dans un url lors d'une communication. Nous avons pu nous en rendre compte en analysant les trames http via l'outils **Wireshark**, nous avons observé les url que nous avions écrit précédemment.

Figure 23 : Requête HTTPS que nous pouvons lire en clair sur Wireshark

Dans ce cas, il pourrait être intéressant de plutôt utiliser la **méthode POST** qui ne fait pas apparaître les données dans l'url, ce qui permet une plus grande **confidentialité des données**. Les données envoyées avec POST sont beaucoup plus compliquées à récupérer par une personne mal intentionnée. C'est ce que nous avions pu étudier dans la ressource « **Initiation au développement Web** ».

Deuxièmement, une **vérification est nécessaire** lorsqu'on reçoit les données d'un utilisateur ou d'un programme, surtout quand on utilise des bases de données, ce qui n'était pas notre cas ici. Lorsque l'on reçoit des données de la part d'utilisateur, il faut toujours vérifier si elles ne contiennent pas de commandes ayant pour but de faire fuiter des données par exemple. Il faut donc interdire certains **caractères/suites de caractères**.

Troisièmement, il est nécessaire de mettre en place **d'avantages de sécurités quant à l'accès au réseau**. Même si un **mot de passe** était nécessaire pour accéder au rester, n'importe quelle personne pouvait obtenir une adresse IP et donc accéder par exemple au serveur web qui pourrait contenir des informations sensibles. Nous aurions pu mettre en place des **réservations d'adresses IP** pour les machines que nous utilisions et seulement celles-ci en renseignant leurs adresses MAC. Cela aurait empêcher des personnes non autorisées d'avoir accès à des IP via DHCP. De plus, nous aurions pu **désactiver les ports de la borne wifi** qui n'étaient pas utilisés pour empêcher la connexion d'un appareil non autorisé sur celle-ci

7. Conclusion

Cette **Situation d'Apprentissage et d'Evaluation** nous a permis d'appréhender d'une nouvelle manière l'Internet des Objets. Pour le moment seule la programmation et l'utilisation de **robots Arduino** et l'étude de signaux à l'aide d'un **dongue USB** nous avaient été présenté.
Cette Saé a donc été une grande découverte dans le domaine.
C'est pour cela qu'elle a permis de **valider les acquis** de cette année de travail, notamment dans le domaine de la **programmation** pour la compréhension, la modification et l'écriture d'un programme Arduino. Mais également la **gestion de projet**, car dans une Saé de cette ampleur, il faut savoir s'organiser et répartir équitablement le travail afin de réaliser les rendus dans le temps attendus. Les ressources **d'Expression-Culture-Communication** et de **Projet Personnel et Professionnel** nous ont permis de réaliser des jalons clairs ainsi qu'un dossier compréhensible. Nous avons pu nous appuyer sur les connaissances acquises durant la ressource **Base des services réseaux** dans laquelle nous avions déjà eu l'occasion de mettre en place un serveur Apache et la « sécurisation » avec HTTPS. Et enfin la ressource **Signaux et Systèmes pour les Transmissions** qui nous a permis de comprendre le fonctionnement physique du capteur de température.
Certains problèmes nous ont empêchés d'avancer aussi rapidement que l'on ne le souhaitait, c'est pour cela qu'il a fallu faire preuve de flexibilité et de coordination dans le groupe. Cependant lors de cette Saé, les problèmes ont été moins importants que dans les précédentes Saé, ce qui vient d'une **meilleure méthodologie de travail**.
Nous avions eu une **situation imprévue** lors de la mise en place de Buster. La cause était la carte SD. Nous avons **contourné le problème** en la changeant tout en continuant sans interrompre les autres membres du groupe qui avançaient de leur côté.
Vis-à-vis du **respect de la méthodologie**, nous pensions l'avoir respecté. Nous avons découpé le projet en plusieurs étapes :

- Répartition et réalisation des tâches ;
- Dépôt des traces dans le Discord ;
- Modification du document Notion ;
- Aide des autres membres du groupe si besoin.

Ce découpage nous a été **favorable**. La facette méthodologie de la **conservation** de certaines traces nous aide aussi beaucoup, principalement pour la présentation lors de la soutenance. C'est pour cela

que cette Situation d'Apprentissage et d'Evaluation nous aura permis de travailler efficacement en équipe. Nous avions eu une première approche de l'option nommée « **IOM** », ce qui a permis de rediscuter de nos **choix de parcours** pour l'année prochaine.

Nous avons dû **chercher comment contourner** des problèmes en cherchant des informations par nous-même et en **travaillant avec d'autres groupes**. Ceci nous permet de capitaliser des connaissances qui pourrons être utilisées dans les projets à venir ainsi que dans notre **vie professionnelle**.