

**TROISIÈME ANNÉE DE BUT  
GÉNIE ÉLECTRIQUE ET  
INFORMATIQUE INDUSTRIELLE**

**IUT D'ÉVRY VAL D'ESSONNE**

**PROJET FIL ROUGE**

**SYSTÈME DE SUPERVISION**  
**CONNECTÉ**

**Année 2024 - 2025**

Préparé par :

**Aubin TOURAIS**

**&**

**Melchior DIEGUEZ**

Enseignant :

**Aylen RICCA**

# SOMMAIRE

<b>Introduction :</b>	<b>3</b>
<b>Cahier des charges :</b>	<b>4</b>
a) Objectifs du projet :	4
b) Développement d'un système pour :	4
c) Composantes clés du projet :	5
 <b>Séance 1 - 11.09.2024.....</b>	 <b>6</b>
I. Description du plan pour le projet :	6
II. Liste des Capteurs :	10
III. Liste des Actionneurs :	10
IV. Liste de composants supplémentaires :	12
V. Liste des Microcontrôleurs :	13
VI. Aperçu du projet "envisagé" :	14
Conclusion de la séance :	15
 <b>Séance 2 - 18.09.2024.....</b>	 <b>16</b>
● Capteur SEN 0193 :	16
● Capteur DHT11 :	19
● Capteur LDR NSL 19M51 :	21
● Câblage sur TinkerCad :	23
● Test - Programme final sur TinkerCad :	24
● Type de scénario possible : (avec les actionneurs)	26
Conclusion de la séance :	28
 <b>Séance 3 - 25.09.2024.....</b>	 <b>29</b>
● Réflexion - Création Pages IHM (Interface Homme Machine)	29
● Réalisation de l'IHM :	29
Page ACCUEIL :	30
Page TEMPÉRATURE / HUMIDITÉ / LUMINOSITÉ :	31
Page HISTORIQUE :	32
Conclusion de la séance :	33
 <b>Séance 4 - 09.10.2024.....</b>	 <b>34</b>
● Communication Grove WiFi 8266 et Arduino	34
Conclusion de la séance :	38
 <b>Séance 5 - 16.10.2024.....</b>	 <b>39</b>
● LED RGB :	39
Conclusion :	45
Conclusion de la séance :	53

<b>Séance 6 - 23.10.2024.....</b>	<b>54</b>
1) Recherche des informations : .....	54
2) Etablir une connexion entre E&R (PTP) depuis l'Arduino : .....	55
3) Re-connecter vos capteurs et définir le format des messages pour récupérer : .....	57
4) Conclusion : .....	63
Conclusion de la séance : .....	64
<b>Séance 7 - 12.11.2024.....</b>	<b>65</b>
Relay, Seleroid valve, battery 6V, diode.....	65
TEST EN RÉSEAU.....	68
Conclusion de la séance : .....	74
<b>Séance 8 - 13.11.2024.....</b>	<b>75</b>
Conclusion de la séance : .....	83
<b>Séance 9 - 19.11.2024.....</b>	<b>84</b>
Conclusion de la séance : .....	99
<b>Conclusion.....</b>	<b>100</b>

## Introduction :

L'objectif du projet "Sauver BERRY" est de concevoir et réaliser un système permettant de subvenir à tous les besoins d'un plan de fraise c'est-à- dire l'humidité du sol et de l'air, la température et l'éclairage.

Notre but est de récupérer les données de nos capteurs positionnés sur notre fraisier avec une carte ESP8266 qui communique en réseau avec notre carte arduino qui commande les actionneurs à l'aide de notre code le tout en stockant les valeurs récupérer afin de pouvoir analyser notre fraisier avec détail.

Une fois le tout fait, sur grafana on affiche les graphiques des différents capteurs en temps réel. Ces informations doivent être accessibles en réseau wifi avec le routeur fournit pour ce projet.

## Cahier des charges :

### a) Objectifs du projet :

Nous avons devant nous un défi captivant pour mettre en pratique tout ce que nous allons apprendre en Réseaux et Supervision Avancés : **sauver "Berry"**, notre fraisier, de ses caprices quotidiens ! Un jour trop arrosé, le lendemain asséché, ou encore mal éclairé... Berry mérite une gestion plus intelligente.

Notre mission ? Concevoir un Système de Supervision pour Berry en utilisant nos compétences en réseaux et supervision. Nous devrons déployer des capteurs pour surveiller en temps réel l'humidité du sol, la température, la lumière, et les nutriments, puis développer un système capable de collecter ces données, de les analyser et de prendre les bonnes décisions pour automatiser les soins de Berry. Tout est question de stratégie : comment organiser le réseau, gérer les flux de données, et garantir une supervision efficace ?

Le but ? Faire en sorte que notre équipe soit celle qui maintient Berry en meilleure santé et qui produit les plus belles fraises. Et pourquoi ne pas couronner le tout avec une récompense délicieuse ? Imaginons le plaisir de déguster un gâteau ou une glace à la fraise lors de notre remise de diplôme, préparé avec les fruits de notre réussite technologique !

### b) Développement d'un système pour :

- Collecter des données en temps réel (humidité du sol, température, humidité de l'air, intensité lumineuse, ...).
- Communiquer ces données à l'aide de protocoles de réseau appropriés.
- Analyser les données pour prendre des décisions éclairées et Automatiser les actions (arrosage, éclairage, ...).
- Visualiser les données pour le suivi et contrôle, ainsi que pour la présentation des rapports.

**c) Composantes clés du projet :**

1. Collecte de données en Temps Réel : Utilisation de capteurs pour surveiller les conditions environnementales et la santé des plantes.
2. Communication des Données : Transfert des données vers un serveur central.
3. Analyse des Données et Prise de Décision : Développement d'algorithmes pour analyser les données et automatiser les actions.
4. Automatisation : Contrôle des actionneurs pour maintenir des conditions optimales.
5. Visualisation : Création d'un tableau de bord pour visualiser les données en temps réel.
6. Rédaction Technique et Présentation : Documenter le processus et présenter les résultats.

# Séance 1 - 11.09.2024

## I. Description du plan pour le projet :

Afin de pouvoir permettre à notre plant de fraise de pousser en toute sérénité et en bonne santé, nous construirons notre projet en se posant avant tout, une question primordiale sur ses conditions de vie. **De quoi a-t-elle besoin ?**

**Pour grandir et se développer, notre plantation a besoin de :**  
**lumière, d'eau et de chaleur.**

### - De lumière :

Nous utiliserons un capteur de lumière qui sera relié à notre carte microcontrôleur afin de pouvoir savoir si notre plantation est en manque ou en surconsommation de lumière. En plus de notre capteur, nous aurons besoin de lampes, qui seront nos actionneurs, afin que si notre plant de fraise soit justement en manque de lumière, de pouvoir l'éclairer jusqu'à ce que notre capteur de luminosité nous indique que celui-ci contient une luminosité supérieure à un seuil que nous lui aurons fixé.

Pour résumer :

- Luminosité < au seuil fixé = Allumer les lampes
- Luminosité > au seuil fixé = Éteindre les lampes

### - D'eau :

Dans le même principe que celui de la lumière, nous aurons un capteur d'humidité du sol, afin de savoir si notre plantation est en manque ou non d'eau. Afin de pouvoir humidifier notre plant de fraise, nous utiliserons une pompe à eau qui se trouvera dans un bac à côté de notre plantation afin d'envoyer de l'eau (sous forme de série) dans un tuyau souple jusqu'à celle-ci.

De plus, afin de pouvoir encore mieux automatiser le tout, nous aurons également un capteur de niveau d'eau dans notre bac, afin de savoir si nous avons suffisamment d'eau ou non pour permettre d'utiliser la pompe.

Si le niveau est trop bas, alors nous allumons une LED rouge, et envoyons une information via bluetooth sur le téléphone, car la plante sera donc en manque d'eau étant donné que la pompe sera incapable de répondre à cette demande, sinon, on pompe l'eau si besoin.

Pour résumer :

- Humidité sol < au seuil fixé = Activation de la pompe à eau(ex : 10 petites séries)
- Humidité sol > au seuil fixé = Ne pas activer la pompe à eau
- Niveau eau réservoir < au seuil fixé = Allumée LED ROUGE - Manque eau
- Niveau eau réservoir > au seuil fixé = Allumée LED VERTE - Niveau eau OK

- De chaleur :

Tout comme les besoins précédents, un niveau de seuil sera comparé à celui du capteur afin de savoir si le plant de fraise à froid, trop chaud ou se trouve dans un intervalle de température satisfaisant.

L'une des solutions serait d'utiliser une plaque que l'on fait chauffer proche de notre plant de fraise afin de lui permettre d'atteindre une température plus élevée (hiver / temps frais).

Afin de d'avoir plusieurs solutions, nous pourrions également utiliser une serre, afin de protéger notre plantation en cas de pluie si elle est dehors, de courant frais, de chaleur importante etc. Puis, selon les cas possibles, nous pourrions ouvrir la serre et la rafraîchir à l'aide de notre système d'arrosage automatique s'il fait chaud, la laisser fermée si la température dans la serre correspond à un intervalle de température prédéfini, correspondant aux besoins de notre plant de fraise, ou bien, fermer la serre ET allumer une plaque chauffante en cas de température trop faible afin de pouvoir remonter la température ambiante de notre plant de fraise.

Pour résumer :

- Température < au seuil fixé = Fermer serre / Allumer la plaque chauffante
- Température : intervalle de température OK = Fermer serre / Éteindre la plaque chauffante
- Température > au seuil fixé = Ouvrir serre / Éteindre la plaque chauffante

- **Suivi de la croissance :**

Afin de pouvoir lire les informations sur notre plantation en direct, soit comme ci-dessous avec le potentiel capteur 4 en 1, ou tout simplement à l'aide d'un afficheur LCD qui se trouverait à côté de notre plant de fraise, qui affichera la température ambiante, l'humidité du sol, le lux (pour la luminosité) etc. Pour l'afficheur on hésite entre deux, le SunFounder LCD1602 Display qui est un afficheur que l'on a déjà utilisé l'année dernière lors d'un projet, qui a deux lignes d'affichage, ou, le ERM2004FS-2-4832 qui est un autre afficheur que l'on a jamais utilisé mais qui nous paraît plus intéressant car il possède 4 lignes d'affichage ce qui nous permettra d'afficher plus d'informations/données sur notre fraisier.

L'objectif est de suivre le fonctionnement à distance mais aussi en étant à côté. De plus, afin de savoir quel actionneur est activé, nous activerons une LED afin de savoir que nous avons une automatisation qui se fait. Par exemple, si notre plant manque d'eau, alors la pompe à eau sera mise en route suite à la lecture du capteur, avec une LED correspondant à l'activation de la pompe à eau sera allumée à côté.

Pour pouvoir suivre sa croissance, ainsi que sa bonne forme à distance, nous envisageons de pouvoir utiliser un capteur tout en un, qui serait connecté à une application mobile "Home Smart" en bluetooth.

Ce capteur permet de mesurer : *Température - Luminosité - Oxygène - Humidité*

Nous nous posons également comme question, pourquoi ne pas utiliser que ce capteur, celui-ci répond à nos attentes, peut être connecté à un téléphone et ainsi nous pourrions utiliser une carte Raspberry Pi 4 pour y lire les données et ainsi plus qu'à utiliser nos actionneurs pour répondre aux manques de notre plant de fraise. Il faudrait voir si la Raspberry Pi en est capable et surtout, comment ferions-nous pour stocker nos valeurs dans un tableau de bord ? Arduino ? Raspberry ?

Pour le moment, nous partons sur des idées simples, en nous disant que nous faisons juste en sorte de lui permettre de répondre à ses propres besoins automatiquement, sans suivre les données des capteurs sur un tableau. Nous chercherons une solution aux stockages des données automatiquement par la suite.

Mais un problème se pose, ce capteur ne sera pas utilisable pour le niveau d'eau dans le bac, il faut donc trouver une autre solution pour suivre à distance le niveau d'eau à distance.

- **Bonus : APPLICATION BLYNK**

Nous avons trouvé une application “BLYNK”, qui permet de suivre en direct les données que lit une carte (ESP32, Arduino, Raspberry Pi etc) en rajoutant dans notre cas un module Ethernet pour la carte Arduino Uno. En utilisant cela, il ne serait pas nécessaire d'utiliser le capteur 4 en 1, nous choisissons nos capteurs selon nos demandes, et connectons le tout dans le code du microcontrôleur à l'application BLYNK pour suivre l'entièreté du fonctionnement de notre projet.

Il faudrait voir si cette application pourrait nous permettre de faire un tableau des données, mais si cette application lit bien les informations envoyées par la carte selon le code, il serait possible ainsi de pouvoir créer un tableau et y stocker chaque valeur selon les intervalles souhaités. Ainsi, nous pourrions répondre à la demande : Visualisation : Création d'un tableau de bord pour visualiser les données en temps réel.

Dans le document suivant : <https://booteille.github.io/blynk-docs-fr/>, nous pouvons voir que le logiciel marche en Wifi, Bluetooth et peut être connecté à nos cartes donc potentiellement une solution pour lire toutes les données simultanément, et les stocker dans un tableau.

**BLYNK => Solution la plus probable pour notre projet.**

## **II. Liste des Capteurs :**

- **Capteur de température :**

<https://www.lextronic.fr/sonde-de-temperature-et-humidite-sht20-sen0257-40695.html>

- **Capteur de lumière :**

<https://www.gotronic.fr/art-capteur-de-lumiere-sen0097-19372.htm>

- **Capteur de niveau d'eau (réservoir) :**

[https://www.otronic.nl/fr/capteur-deau-pour-arduino-esp32-esp8266.html?source=googlebase&gad\\_source=1&gclid=CjwKCAjw\\_4S3BhAAEiwA\\_64Yhnas-ODIzjLn5sf7fUyRAy0nxGILL2lnXLSRiMBFIfMIFvt4trwyRoCjE4QAvD\\_BwE](https://www.otronic.nl/fr/capteur-deau-pour-arduino-esp32-esp8266.html?source=googlebase&gad_source=1&gclid=CjwKCAjw_4S3BhAAEiwA_64Yhnas-ODIzjLn5sf7fUyRAy0nxGILL2lnXLSRiMBFIfMIFvt4trwyRoCjE4QAvD_BwE)

- **Capteur d'humidité du sol :**

<https://www.amazon.fr/dp/B07HJ6N1S4?tag=boilingbrains-21>

- **Capteur connecté via application (4en1 -> Home Smart) :**

[https://www.amazon.fr/Hygrometre-Intelligente-Surveillance-Automatique-Temp%C3%A9ratu re/dp/B096ZBQ1CG/ref=asc\\_df\\_B096ZBQ1CG/?tag=googshopfr-21&linkCode=df0&hvadid=701608457753&hvpos=&hvnetw=g&hvrand=1296421704938554917&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9056476&hvtargid=pla-1395540643852&psc=1&mcid=90677956313734959397ee94ad9c25ab&gad\\_source=1](https://www.amazon.fr/Hygrometre-Intelligente-Surveillance-Automatique-Temp%C3%A9ratu re/dp/B096ZBQ1CG/ref=asc_df_B096ZBQ1CG/?tag=googshopfr-21&linkCode=df0&hvadid=701608457753&hvpos=&hvnetw=g&hvrand=1296421704938554917&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9056476&hvtargid=pla-1395540643852&psc=1&mcid=90677956313734959397ee94ad9c25ab&gad_source=1)

## **III. Liste des Actionneurs :**

- **Pompe à eau :**

<https://www.amazon.fr/dp/B01FTL6E7M?tag=boilingbrains-21> OU

[https://www.amazon.fr/dp/B08SQGXZLG/ref=sspa\\_dk\\_detail\\_2?psc=1&pd\\_rd\\_i=B08SQGXZLG&pd\\_rd\\_w=jS0jk&content-id=amzn1.sym.13198011-2862-4d39-8574-319cce927b66&pf\\_rd\\_p=13198011-2862-4d39-8574-319cce927b66&pf\\_rd\\_r=ZWJYAZY9ARW64E3Y2W2Z&pd\\_rd\\_wg=fN5HD&pd\\_rd\\_r=e5c85466-bdd6-49ab-9c39-88d6adf896ce&sp\\_csd=d2lkZ2V0TmFtZT1zcF9kZXRhaWxfdGhlbWF0aWM](https://www.amazon.fr/dp/B08SQGXZLG/ref=sspa_dk_detail_2?psc=1&pd_rd_i=B08SQGXZLG&pd_rd_w=jS0jk&content-id=amzn1.sym.13198011-2862-4d39-8574-319cce927b66&pf_rd_p=13198011-2862-4d39-8574-319cce927b66&pf_rd_r=ZWJYAZY9ARW64E3Y2W2Z&pd_rd_wg=fN5HD&pd_rd_r=e5c85466-bdd6-49ab-9c39-88d6adf896ce&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhaWxfdGhlbWF0aWM)

- **Plaque Chauffante :**

[https://www.amazon.fr/%C3%89%C3%A9ment-Chauffant-Applicable-Miniature-Chauffage/dp/B07FY4PCL/ref=asc\\_df\\_B07FY4PCL/?tag=googshopfr-21&linkCode=df0&hvadid=701535511870&hvpos=&hvnetw=g&hvrand=5683367153374620125&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcndl=&hvlocint=&hvlocphy=9056476&hvtargid=pla-664806488325&psc=1&mcid=62ba3df24f9f30b0a9145fe73e2386be&gad\\_source=1](https://www.amazon.fr/%C3%89%C3%A9ment-Chauffant-Applicable-Miniature-Chauffage/dp/B07FY4PCL/ref=asc_df_B07FY4PCL/?tag=googshopfr-21&linkCode=df0&hvadid=701535511870&hvpos=&hvnetw=g&hvrand=5683367153374620125&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcndl=&hvlocint=&hvlocphy=9056476&hvtargid=pla-664806488325&psc=1&mcid=62ba3df24f9f30b0a9145fe73e2386be&gad_source=1)

- **Afficheur LCD :**

[OU](https://www.amazon.fr/dp/B019K5X53O?tag=boilingbrains-21)  
<https://www.cafr.ebay.ca/itm/293187459755>

- **Kit de LEDs :**

[https://www.amazon.fr/AQSQWQ-%C3%A9mettant-%C3%A9lectroluminescente-Lumi%C3%A8re-Emettant/dp/B0CXDS8LDL/ref=sr\\_1\\_5?\\_mk\\_fr\\_FR=%C3%85M%C3%85%C5%B0%C3%95%C3%91&dib=eyJ2ljoIMsj9.ztUFn4k\\_haJHtlSzsYzVSzyBTqf9zdb1ic0rlSOAkx0tTVQnCw3hzHmc57hz7JruCU1jEMvkPsE\\_lyUVm9DxFz1I1epKC8jrEGSXRsI5\\_JBW5hEW\\_HWefDsURboPFZzyLY\\_DpbEiz5xpI6EbhFL5twzzEctgpyG9DBEOGWaKQFqs1Bit1cgTC0TKGnTk-bXbQtXzx4G2nqjlPyZxrJnoTKiV2ClrElhSyuME99U7\\_n765n4bvQ7sRdbG97AucuwXzQaLgoOCWHueTVIF2EmCgz5tU8TE9i\\_3UUo1nellF\\_U.UvH-nfySfdEkNHKiz4X-s90GDFY1mPX\\_N0AsrXOPIAs&dib\\_tag=se&keywords=LED+arduino&qid=1726404287&sr=8-5](https://www.amazon.fr/AQSQWQ-%C3%A9mettant-%C3%A9lectroluminescente-Lumi%C3%A8re-Emettant/dp/B0CXDS8LDL/ref=sr_1_5?_mk_fr_FR=%C3%85M%C3%85%C5%B0%C3%95%C3%91&dib=eyJ2ljoIMsj9.ztUFn4k_haJHtlSzsYzVSzyBTqf9zdb1ic0rlSOAkx0tTVQnCw3hzHmc57hz7JruCU1jEMvkPsE_lyUVm9DxFz1I1epKC8jrEGSXRsI5_JBW5hEW_HWefDsURboPFZzyLY_DpbEiz5xpI6EbhFL5twzzEctgpyG9DBEOGWaKQFqs1Bit1cgTC0TKGnTk-bXbQtXzx4G2nqjlPyZxrJnoTKiV2ClrElhSyuME99U7_n765n4bvQ7sRdbG97AucuwXzQaLgoOCWHueTVIF2EmCgz5tU8TE9i_3UUo1nellF_U.UvH-nfySfdEkNHKiz4X-s90GDFY1mPX_N0AsrXOPIAs&dib_tag=se&keywords=LED+arduino&qid=1726404287&sr=8-5)

- **Têtes de lampes pour plantes :**

[https://www.amazon.fr/Niello-Plantes-160LED-Croissance-Horticole/dp/B0BZS1VHHL/ref=sr\\_1\\_1\\_sspa?\\_mk\\_fr\\_FR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=19AQYLYQ78PKW&dib=eyJ2ljoIMsj9.zUNwnMJq2JzrW7rzjLCBEoINQIMVIxF3uPrKJqvUUy9U6OMAA6Gi7wzOrfLSZcmstxuctPe50lHiY0ig3H\\_HeLsCNw1zkpzN1A76YIEqaki6HcH-\\_BFUwZJtGOtAS7QXg2ioeUDI3Z4zhR8f\\_bKGkK5wwd2scPMSSd\\_BxvMINjggBsUC884bTm8-ZvGZrE48XtWXFd8qAvskuiKFHEO6S6t4dbG5eQuMD6yhaWyxecXB9\\_I8GTztLt\\_BmglyTvi-j6DLH6VVv1UN6\\_W1p6Qfq4A8\\_h9gv5INI7r6FPC\\_HCQ.4\\_Xqbs6qPjrNjG88IrisKUPLoDfSLECMVu7zucJ6uB0&dib\\_tag=se&keywords=lampe+plante+arduino&qid=1726405007&s=lawn-garden&sprefix=lampe+plante+a%2Cgarden%2C718&sr=1-1-spons&sp\\_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1](https://www.amazon.fr/Niello-Plantes-160LED-Croissance-Horticole/dp/B0BZS1VHHL/ref=sr_1_1_sspa?_mk_fr_FR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=19AQYLYQ78PKW&dib=eyJ2ljoIMsj9.zUNwnMJq2JzrW7rzjLCBEoINQIMVIxF3uPrKJqvUUy9U6OMAA6Gi7wzOrfLSZcmstxuctPe50lHiY0ig3H_HeLsCNw1zkpzN1A76YIEqaki6HcH-_BFUwZJtGOtAS7QXg2ioeUDI3Z4zhR8f_bKGkK5wwd2scPMSSd_BxvMINjggBsUC884bTm8-ZvGZrE48XtWXFd8qAvskuiKFHEO6S6t4dbG5eQuMD6yhaWyxecXB9_I8GTztLt_BmglyTvi-j6DLH6VVv1UN6_W1p6Qfq4A8_h9gv5INI7r6FPC_HCQ.4_Xqbs6qPjrNjG88IrisKUPLoDfSLECMVu7zucJ6uB0&dib_tag=se&keywords=lampe+plante+arduino&qid=1726405007&s=lawn-garden&sprefix=lampe+plante+a%2Cgarden%2C718&sr=1-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1)

## **IV. Liste de composants supplémentaires :**

- **Tuyau souple :**

[https://www.amazon.fr/transparent-alimentaire-r%C3%A9sistante-flexible-installations/dp/B0C58CM9DP?source=ps-sl-shoppingads-lpcontext&ref\\_=fplfs&smid=A1M0FZMR9RV2FP&th=1](https://www.amazon.fr/transparent-alimentaire-r%C3%A9sistante-flexible-installations/dp/B0C58CM9DP?source=ps-sl-shoppingads-lpcontext&ref_=fplfs&smid=A1M0FZMR9RV2FP&th=1)

- **Transistor MOSFET (arrosage plantation) :**

<https://www.amazon.fr/dp/B08H56RJY9?tag=boilingbrains-21>

- **Diode redresseur (arrosage plantation) :**

<https://www.amazon.fr/dp/B08H56RJY9?tag=boilingbrains-21>

- **Alimentation 12V (pour Carte arduino + pompe) :**

<https://www.amazon.fr/dp/B07PGLXK4X?tag=boilingbrains-21>

- **Kit de Résistances :**

[https://www.amazon.fr/BOJACK-valeurs-r%C3%A9sistances-dassortiment-r%C3%A9sistances/dp/B08FD1XVL6/ref=sr\\_1\\_5?dib=eyJ2ljojMSJ9.A4IFvxke1cXmjxCmlG9NvehxK7A3C2tf3Baex1JCjtsGl7EIfHQ3d-JjGpITVaMGE7WOoMwYOs-QOaNrXk2GBxWSPlwV2TH\\_ZI9ah3XuxbD\\_NX0ehQqcVLXWVUrRla4e3Dni8mYRYILfOPjGYx3ZlgwKw69E5HPvWtI3dBWZmeL-QfMfMzTiTliSRSSpVFtFnTtgQipAejdECntGwDAhjngOBPNXM5yeMFfKjnidP98FnOSyxZAoUKDkmtYxKdJRT4FmCRgSR8Rum4a40Wk-Fnes1nCZqfpCBByk73XQ.SZCvTiOj8bhJKZrj1Z0Xw2M32PRBgh2c-Kq1ZohKTTI&dib\\_tag=se&keywords=resistance%2Barduino&qid=1726403935&sr=8-5&th=1](https://www.amazon.fr/BOJACK-valeurs-r%C3%A9sistances-dassortiment-r%C3%A9sistances/dp/B08FD1XVL6/ref=sr_1_5?dib=eyJ2ljojMSJ9.A4IFvxke1cXmjxCmlG9NvehxK7A3C2tf3Baex1JCjtsGl7EIfHQ3d-JjGpITVaMGE7WOoMwYOs-QOaNrXk2GBxWSPlwV2TH_ZI9ah3XuxbD_NX0ehQqcVLXWVUrRla4e3Dni8mYRYILfOPjGYx3ZlgwKw69E5HPvWtI3dBWZmeL-QfMfMzTiTliSRSSpVFtFnTtgQipAejdECntGwDAhjngOBPNXM5yeMFfKjnidP98FnOSyxZAoUKDkmtYxKdJRT4FmCRgSR8Rum4a40Wk-Fnes1nCZqfpCBByk73XQ.SZCvTiOj8bhJKZrj1Z0Xw2M32PRBgh2c-Kq1ZohKTTI&dib_tag=se&keywords=resistance%2Barduino&qid=1726403935&sr=8-5&th=1)

- **Kit de Transistor (précisément après schématisation et calcul) :**

[https://www.amazon.fr/Transistor-24Valeur-600Pcs-Transistors-2N2907/dp/B0BW9BMW1F/ref=asc\\_df\\_B0BW9BMW1F/?tag=googshopfr-21&linkCode=df0&hvadid=701622263954&hvpos=&hvnetw=g&hvrand=7814946978257331067&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9056476&hvtargid=pla-2193717978220&psc=1&mcid=97490d6df4673e1f81672d527bdee8fd&gad\\_source=1](https://www.amazon.fr/Transistor-24Valeur-600Pcs-Transistors-2N2907/dp/B0BW9BMW1F/ref=asc_df_B0BW9BMW1F/?tag=googshopfr-21&linkCode=df0&hvadid=701622263954&hvpos=&hvnetw=g&hvrand=7814946978257331067&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9056476&hvtargid=pla-2193717978220&psc=1&mcid=97490d6df4673e1f81672d527bdee8fd&gad_source=1)

- **Ethernet Shield 2 :**

<https://www.gotronic.fr/art-ethernet-shield-2-a000024-23299.htm>

## **V. Liste des Microcontrôleurs :**

- **Carte Arduino :**

<https://www.gotronic.fr/art-arduino-uno-cms-a000073-19380.htm>

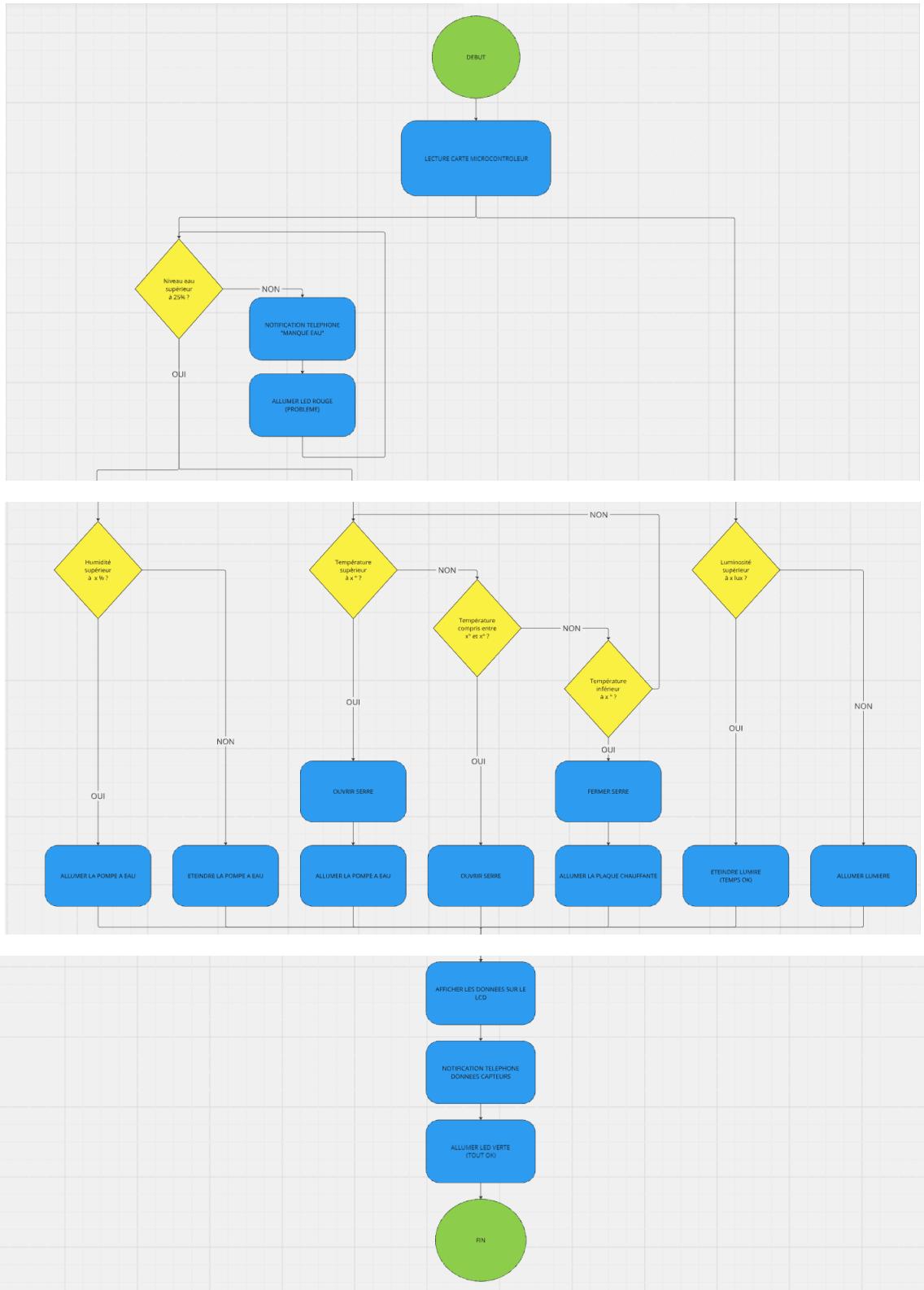
- **Carte Raspberry Pi 4:**

[https://www.amazon.fr/Raspberry-Pi-4595-mod%C3%A8les-Go/dp/B09TTNF8BT/ref=asc\\_df\\_B09TTNF8BT/?tag=googshopfr-21&linkCode=df0&hvadid=701511851267&hvpos=&hvnetw=g&hvrand=6850354763783709687&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9056476&hvtargid=pla-1679572971964&psc=1&mcid=e9453b632e653cdca350de98083c3cb0&gad\\_source=1](https://www.amazon.fr/Raspberry-Pi-4595-mod%C3%A8les-Go/dp/B09TTNF8BT/ref=asc_df_B09TTNF8BT/?tag=googshopfr-21&linkCode=df0&hvadid=701511851267&hvpos=&hvnetw=g&hvrand=6850354763783709687&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9056476&hvtargid=pla-1679572971964&psc=1&mcid=e9453b632e653cdca350de98083c3cb0&gad_source=1)

## VI. Aperçu du projet “envisagé” :

- Algorigramme du projet :

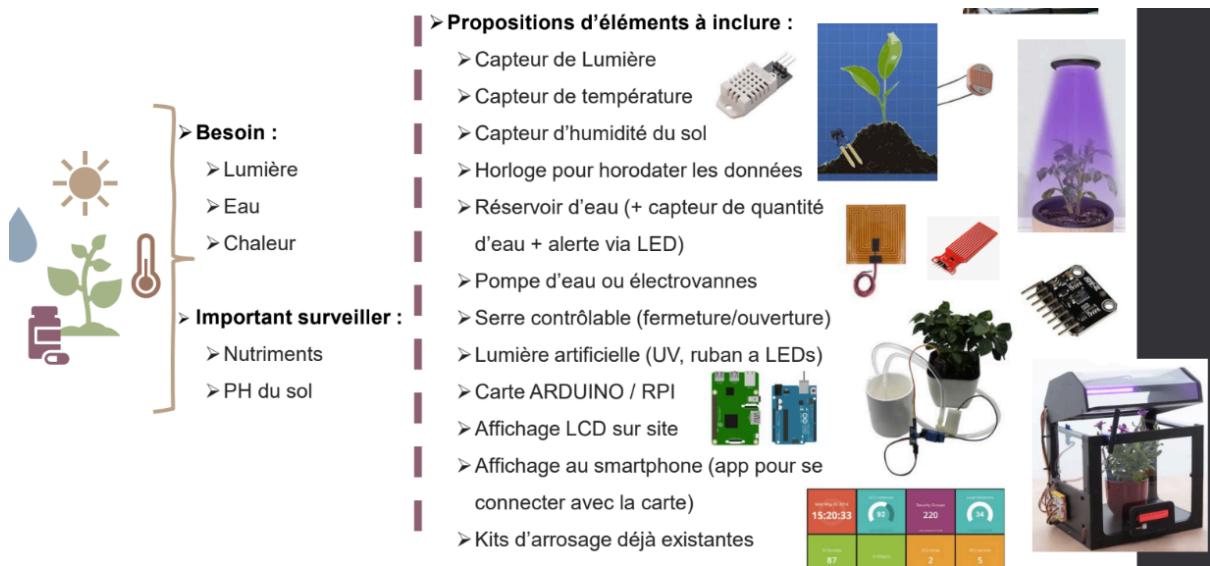
<https://miro.com/app/board/uXjVKgElaOk=/?diagramming=>



## **Conclusion de la séance :**

Lors de cette séance, nous avons détaillé le plan de notre projet pour automatiser la gestion des conditions de croissance d'une plante de fraise. Nous avons analysé les besoins essentiels de la plante, à savoir la lumière, l'eau et la chaleur, et avons identifié les capteurs et actionneurs nécessaires pour répondre à ces besoins. En particulier, nous avons choisi des capteurs de luminosité, d'humidité du sol, de température et de niveau d'eau, ainsi que des actionneurs comme des lampes, une pompe à eau et une plaque chauffante. Nous avons également exploré la possibilité d'utiliser une application mobile pour suivre les données et interagir avec notre système à distance, avec Blynk comme solution privilégiée. Enfin, nous avons discuté des différentes options pour l'affichage des informations, y compris des afficheurs LCD et la possibilité d'utiliser une carte Raspberry Pi pour centraliser les données. Ce travail a permis de poser les bases d'une solution automatisée et de définir les composants clés pour le bon fonctionnement du système.

## Séance 2 - 18.09.2024

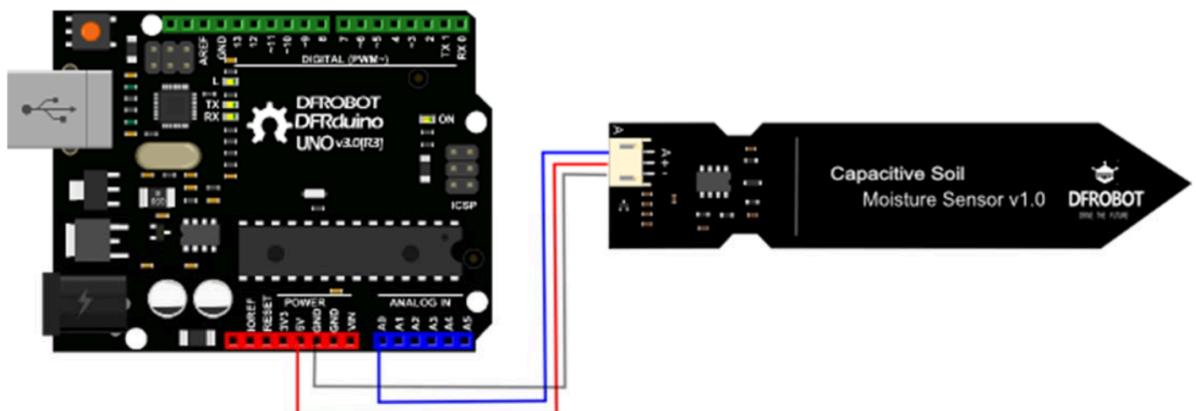


Composants pour notre projet :

- ARDUINO UNO
- Humidité : **SEN 0193**
- Température + humidité : **DHT 11**
- Luminosité : **LDR NSL 19M51**

[https://www.tinkercad.com/things/9ZaTwqL5yM0-swanky-kup-tumelo/editel?sharecode=Gd85ensx52-uJJrfVxP7HmsNdcejIstBC57gfcRv\\_3l](https://www.tinkercad.com/things/9ZaTwqL5yM0-swanky-kup-tumelo/editel?sharecode=Gd85ensx52-uJJrfVxP7HmsNdcejIstBC57gfcRv_3l)

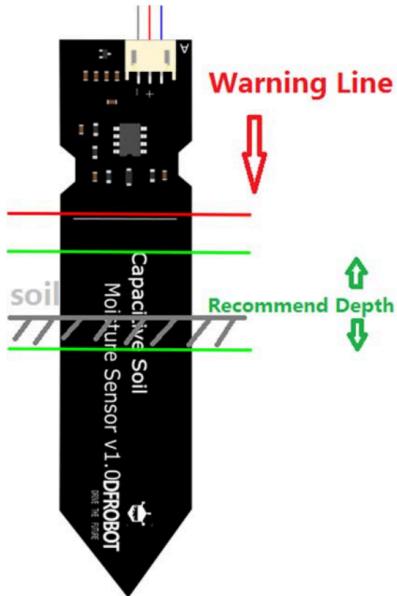
- Capteur SEN 0193 :



A = Analogique

+ = VCC

- = GND



### Caractéristique capteur :

- Type : Capteur capacitif d'humidité du sol
- Tension d'alimentation : 3,3V à 5,5V
- Sortie : Signal analogique (proportionnel à l'humidité du sol)
- Plage de mesure : De sec à très humide, sans être affecté par la corrosion (à la différence des capteurs résistifs)
- Caractéristiques spéciales :

Plus durable que les capteurs résistifs classiques, car il n'y a pas de contact direct avec l'eau ou le sol, évitant ainsi la corrosion.

Le capteur ne doit pas être trop dans le sol car les composants au-dessus de la ligne rouge ne doivent pas être mouillés ou humide car cela peut poser problème.

### **Test Programme :**

```
// Capteur d'humidité sur TinkerCard
#define Soil_Moisture_Sensor A0 // composante photoresistor sur la pin A0

unsigned int value;

void setup() {
    // initialise la communication avec le PC
    Serial.begin(9600);

    // initialise les broches
    pinMode(Soil_Moisture_Sensor, INPUT);
}

void loop() {
    value = analogRead(Soil_Moisture_Sensor);      // mesure la tension sur la broche A0
    float pourcentage = (value / 876.0) * 100;

    if(value < 400){ // Inférieur à 400 = Peu humide => Activer arrosage
        Serial.print("Sol Peu humide : ");
    }
    else if (value > 500){ // Supérieur à 500 = Trop humide => Désactiver arrosage
        Serial.print("Sol Trop humide : ");
    }
    else { // Bon intervalle pour la fraise = Bonne humidité du sol => Désactiver arrosage
        Serial.print("Sol bien humidifie : ");
    }

    Serial.print(value);
    Serial.print(" - En pourcentage : "); Serial.print(pourcentage); Serial.println("%");

    delay(200);
}
```

- **Capteur DHT11 :**
- **Type :** Capteur de température
- **Plage de mesure :** Température de 0 à 50 degrés.      +/- 2%  
Humidité de 20 à 80%      +/- 5%
- **Sortie :** Signal digital sur le pin numéro 2
- **Tension d'alimentation :** 3,3V à 5,5V

### 3. Typical Application (Figure 1)

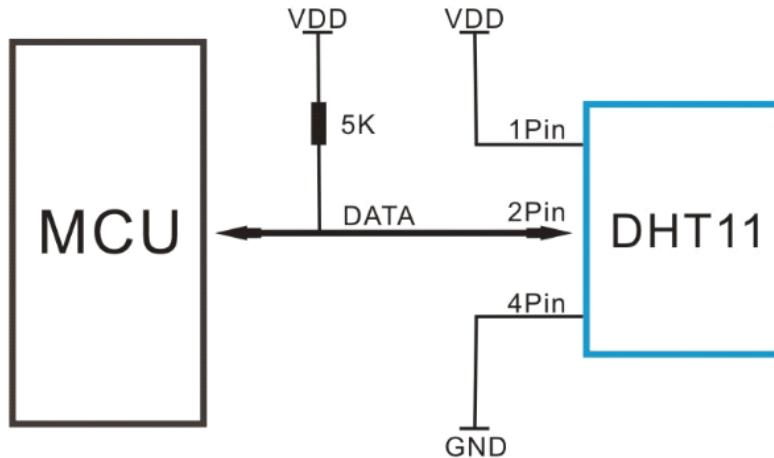


Figure 1 Typical Application

Note: 3Pin – Null; MCU = Micro-computer Unite or single chip Computer

#### Caractéristique capteur :

- **Type :** Capteur numérique de température et d'humidité
- **Tension d'alimentation :** 3,3V à 5,5V
- **Sortie :** Signal digital (40 bits of data (H, T, checksum))
  - Le capteur envoie des impulsions spécifiques basées sur le temps qui représentent des données binaires (0 et 1).
  - interprétées par l'Arduino pour récupérer les données de température et d'humidité
- **Plage de mesure :**
  - Température : 0°C à 50°C (précision :  $\pm 2^\circ\text{C}$ )
  - Humidité : 20% à 90% (précision :  $\pm 5\%$ )

### **Test Programme :**

```
#include "DHT.h"
DHT dht(2, DHT11);

void setup() {
    dht.begin();
    Serial.begin(9600);
}

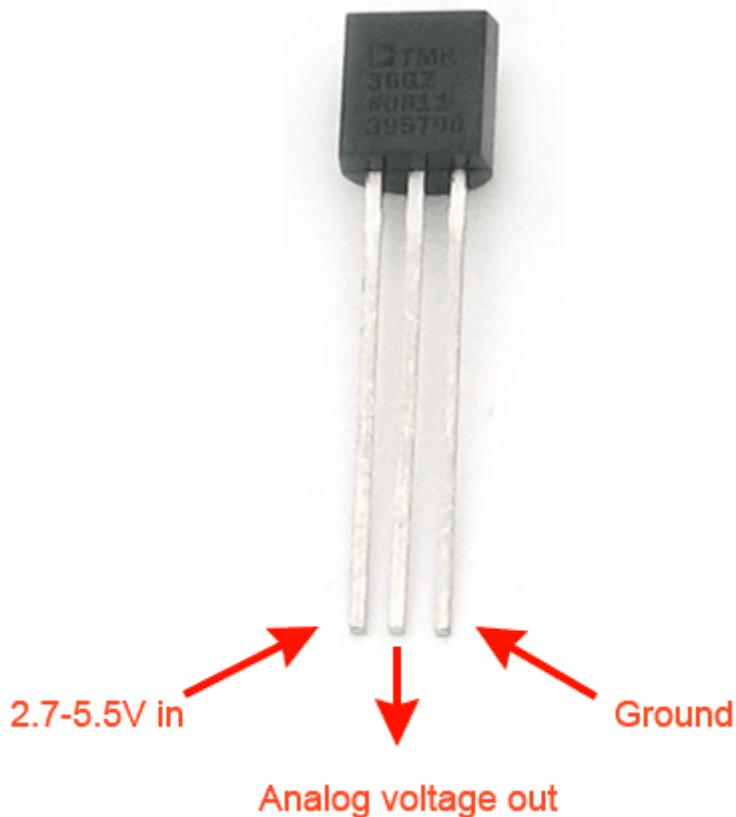
void loop() {
    float h = dht.readHumidity();
    float t = dht.readTemperature();

    Serial.print("Humidity: ");
    Serial.println(h);

    Serial.print("Temperature: ");
    Serial.println(t);

    delay(1000);
}
```

Dans le site web tinkercad on a testé le capteur de température TMP36  
La branche du milieu vient se connecter à un pin Analogique sur la carte arduino.



### **Test Programme :**

```
// Capteur Température sur TinkerCard
|
int Capteur = A1; // Corresponds à la broche A1 ou est connecté le capteur de température

void setup() {
    Serial.begin(9600); // Permet d'initialiser le moniteur série
}

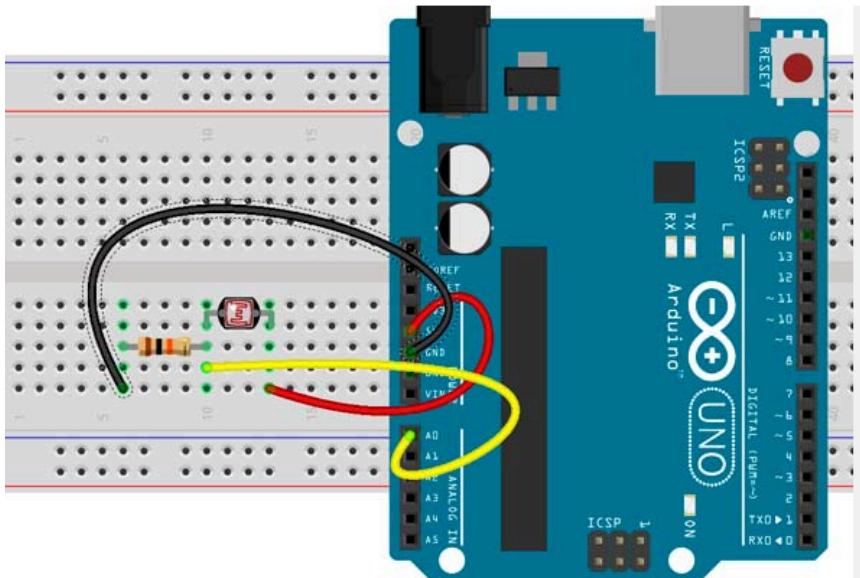
void loop () {
    int lecture = analogRead(Capteur); // On lit la valeur du capteur
    float voltage = 5.0 /1024 * lecture ; // On transforme cette valeur en tension
    float temperature = voltage * 100 - 50 ; // On transforme cette tension en température en degré
    Serial.print("La température est : "); // On affiche la température
    Serial.print (temperature);
    Serial.println(" degrées");
    delay(500); // On fait une pause entre chaque mesure
}
```

### **• Capteur LDR NSL 19M51 :**



### **Caractéristique capteur :**

- **Type** : LDR (Light Dependent Resistor), photorésistance sensible à la lumière
- **Résistance à la lumière** : Environ 10kΩ à 50kΩ
- **Tension d'alimentation** : 5V
- **Sortie** : Signal analogique
- Température d'utilisation -60 / +75°C.



### Test Programme :

```
// Capteur Luminosité sur TinkerCard
#define LDR A2 // composante photoresistor sur la pin A2

unsigned int value;

void setup() {
    // initialise la communication avec le PC
    Serial.begin(9600);

    // initialise les broches
    pinMode(LDR, INPUT);
}

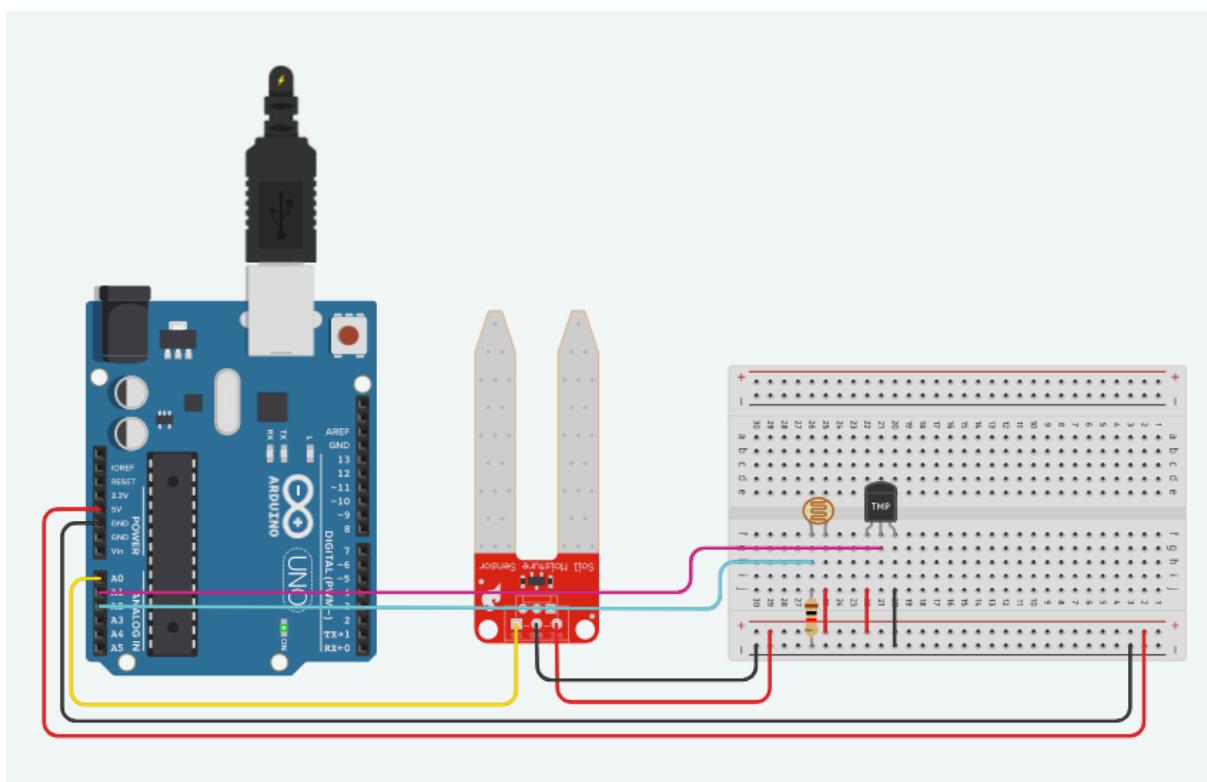
void loop() {
    value = analogRead(LDR); // mesure la tension sur la broche A2
    float pourcentage = (value / 679.0) * 100;

    if(value < 450){ // Inférieur à 400 = manque de lumière => Activer éclairage
        Serial.print("Peu de luminosité : ");
    }
    else if (value > 600){ // Supérieur à 500 = Trop de lumière => Désactiver éclairage
        Serial.print("Trop de luminosité : ");
    }
    else { // Bon intervalle pour la fraise = Bonne luminosité => Désactiver éclairage
        Serial.print("Bonne luminosité : ");
    }

    Serial.print(value);
    Serial.print(" - En pourcentage : "); Serial.print(pourcentage); Serial.println("%");

    delay(200);
}
```

- Câblage sur TinkerCad :



- **Test - Programme final sur TinkerCad :**

```

// -----
// Union des Capteurs sur TinkerCard
// -----

#define Soil_Moisture_Sensor A0 // composante photoresistor sur la pin A0
#define Temperature_Sensor A1 // Corresponds à la broche A1 ou est connecté le capteur de température A1
#define LDR A2 // composante photoresistor sur la pin A2

#define MaxSol 600
#define MinSol 400

#define MaxLum 450
#define MinLum 300
|
#define MaxTemp 22
#define MinTemp 10

void setup() {
    // initialise la communication avec le PC
    Serial.begin(9600);

    // initialise les broches
    pinMode(LDR, INPUT);
    pinMode(Soil_Moisture_Sensor, INPUT);
}

void loop() {
    int value_Luminosite = Eclairage(); // Lecture pour la luminosité
    float pourcentage_Luminosite = (value_Luminosite / 679.0) * 100;

    Serial.print("Eclairage : "); Serial.print(value_Luminosite);
    Serial.print(" - En pourcentage : "); Serial.print(pourcentage_Luminosite); Serial.println(" %");

    int value_Sol = Sol(); // Lecture pour le sol
    float pourcentage_Sol = (value_Sol / 876.0) * 100;

    Serial.print("Humidite sol : "); Serial.print(value_Sol);
    Serial.print(" - En pourcentage : "); Serial.print(pourcentage_Sol); Serial.println(" %");

    float voltage = 5.0 / 1024 * analogRead(Temperature_Sensor); // On lit la valeur du capteur
    float temperature = voltage * 100 - 50; // On transforme cette tension en température en degré
    Temp(temperature);

    Serial.print("Temperature ambiant : "); Serial.print(temperature); Serial.println(" degrees");

    Serial.println("-----"); // Séparation entre chaque lecture

    delay(1000);
}

// -----
// Fonctions
// -----

int Eclairage(){
    int value = analogRead(LDR); // mesure la tension sur la broche A2

    if(value < MaxLum){ // Inférieur à 400 = manque de lumière => Activer éclairage
        Serial.print("Peu de luminosité : ");
    }
    else if (value > MinLum){ // Supérieur à 500 = Trop de lumière => Désactiver éclairage
        Serial.print("Trop de luminosité : ");
    }
    else { // Bon intervalle pour la fraise = Bonne luminosité => Désactiver éclairage
        Serial.print("Bonne luminosité : ");
    }

    return(value);
}

```

```

int Sol(){
    int value = analogRead(Soil_Moisture_Sensor); // mesure la tension sur la broche A0

    if(value < MinSol){ // Inférieur à 400 = Peu humide => Activer arrosage
        Serial.print("Sol Peu humide : ");
    }
    else if (value > MaxSol){ // Supérieur à 500 = Trop humide => Désactiver arrosage
        Serial.print("Sol Trop humide : ");
    }
    else { // Bon intervalle pour la fraise = Bonne humidité du sol => Désactiver arrosage
        Serial.print("Sol bien humidifie : ");
    }

    return(value);
}

int Temp(int value){
    if(value < MinTemp){ // Inférieur à 0 = Tres froid
        Serial.print("Trop froid : ");
    }
    else if (value > MaxTemp){ // Supérieur à  = Tres chaud
        Serial.print("Trop chaud : ");
    }
    else { // Bon intervalle pour la fraise = frais
        Serial.print("Air parfait : ");
    }

    return(value);
}

```

## Données :

### Les fraisiers :

- Température optimale :
  - entre 18 et 24°C
- Humidité sol :
  - entre 60 et 70/80 %
- Humidité ambiante
  - entre 70 et 80 %
- Lumière
  - entre 25k et 50k lux
  - entre 6 et 8 heures du soleil par jour
- PH entre 6.5 et 7.5

➤ **SEN0193** : Mesurer l'humidité du sol pour assurer un arrosage optimal.

➤ **DHT11** : Mesurer la température ambiante et l'humidité, permettant d'ajuster l'environnement de croissance de la plante (ventilation, chauffage, ouverture/fermeture de la serre).

➤ **LDR NSL-19M51** : Déetecter l'intensité lumineuse afin de simuler l'éclairage nécessaire pour la photosynthèse et ajuster l'activation des LED de croissance.

- **Type de scénario possible : (avec les actionneurs)**

Dans le code où l'on a l'assemblage de chaque fonctions afin de pouvoir faire les tests global, nous avons mis en place des étiquettes (`#define`) afin de pouvoir changer les valeurs dans l'entièreté du code sans à se prendre la tête les uns après les autres, mais aussi sans en oublier.

Dans notre test, nous avons mis de simples valeurs afin de voir le bon fonctionnement. Maintenant, que nous avons les informations pour la bonne pousse d'un plan de fraise, nous pouvons modifier les étiquettes afin de pouvoir permettre la mesure dans les bons intervalles.

Etant donné que dans notre code nous faisons le calcul afin de pouvoir lire en pourcentage, et que pour les informations de la pousse de fraise ils sont en pourcentage, nous pouvons modifier les étiquettes en faisant le calcul inverse. Ici, on en aura besoin pour l'humidité du sol : Pour rappel, MinSol on avait dit 60% et le MaxSol = 80%

$$\text{MinSol} = (60/100)*876 = 0,60*876 = \mathbf{525,6}$$

$$\text{MaxSol} = (80/100)*876 = 0,80*876 = \mathbf{700,8}$$

De plus, il faudrait retirer le pourcentage Lux et plutôt mettre les unités et ajouter les kilos lux, donc faire  $(\text{Value} * 1000) = \text{valeurs en lux}$ .

Exemple : Lecture(value) == 23 ->  $(\text{Value} * 1000) = 23000$  Lux donc manque de lumière !

A noter, que dans notre code, nous avons mis les valeurs max des valeurs lues par nos capteurs sur TinkerCad, avant la conception il faudra les changer en étiquette et mettre les valeurs correspondantes afin de pouvoir lire le min/max de nos capteurs.

On changera donc comme étiquettes :

- MinSol : **525,6**
- MaxSol : **700,8**

- MinLum : **25**
- MaxLum : **50**
- MinTemp : **18**
- MaxTemp : **24**

Maintenant, voici différents types d'actions que nous pourrions faire selon les cas, il faudrait simplement activer une variable dans le code, qui mettrait une sortie à 1 par exemple pour allumer des lumières, puis à 0 pour éteindre, pareil pour une électrovanne etc. Dans le code ([Test - Programme final sur TinkerCad](#)) nous avons simplement mis des messages, avec des Serial.print(), il y aura juste à rajouter l'incrémentation de la variable souhaitée.

Afin d'y avoir un aperçu, voici un algorithme résumé des différents cas :

- Température (DHT11) :

Si Température optimale == [18, 24]

Afficher(Température Bonne)

Sinon si Température optimale > 24

Allumer la ventilation / Ouvrir la serre

Sinon si Température optimale < 18

Allumer le chauffage / Fermer la serre

- Humidité Sol (SEN0193) :

Si Humidité sol == [60, 80] // EN POURCENTAGE

Afficher(Humidité du sol Bonne)

Sinon si Humidité sol > 80

Arrêter l'arrosage / allumer pendant peu de temps le chauffage

Sinon si Humidité sol < 60

Allumer l'arrosage

- Luminosité (LDR NSL-19M51) :

Si Luminosité == [25, 50] // EN KILOS LUX

Afficher(Luminosité Bonne)

Sinon si Luminosité > 50

Eteindre la lumière ET Fermer la serre // Pour l'assombrir

Sinon si Luminosité < 25

Allumer la lumière ET Ouvrir la serre si le soleil est présent

Donc si nous pouvons ajouter le capteur solaire, nous pouvons voir s'il y a du soleil ou non (jour/nuageux/nuit), mais aussi un capteur de luminosité vers le plant de fraise afin de voir si la luminosité qu'elle possède suffit.

Si oui = On éteint tout

Si non ET soleil dehors = On ouvre la serre

Si non ET absence soleil dehors = On allume la lumière

## **Conclusion de la séance :**

Nous avons choisi plusieurs capteurs pour notre projet de serre automatisée pour fraises :

1. **Capteur d'humidité du sol SEN 0193** : Permet de mesurer l'humidité du sol grâce à un signal analogique, avec une plage de mesure de sec à très humide.
2. **Capteur de température et d'humidité DHT11** : Fournit des données sur la température (0-50°C) et l'humidité (20-80%) via un signal digital.
3. **Capteur de luminosité LDR NSL 19M51** : Mesure l'intensité lumineuse avec un signal analogique et fonctionne dans une plage de température de -60°C à +75°C.

Nous avons également testé les capteurs sur Tinkercad, en utilisant des valeurs de référence pour chaque capteur. En fonction des lectures, nous avons développé un algorithme pour ajuster les conditions de la serre :

- **Température** : Si elle est trop élevée ou trop basse, on active la ventilation ou le chauffage.
- **Humidité du sol** : En fonction de l'humidité, on active l'arrosage ou le chauffage.
- **Luminosité** : Si elle est insuffisante, on allume la lumière ou on ouvre la serre si le soleil est présent.

Les valeurs de capteurs ont été calibrées avec des étiquettes dans le code, ce qui permet d'ajuster facilement les seuils et de contrôler les actionneurs comme les lumières, l'arrosage ou la ventilation. Le projet se prépare ainsi à répondre automatiquement aux besoins des plantes de fraises.

## Séance 3 - 25.09.2024

- **Réflexion - Création Pages IHM (Interface Homme Machine)**

Notre plan est de faire une page d'accueil avec toutes les statistiques (Température, luminosité, humidité air et sol).

**TOUTES PAGES ET MESURES SONT DES EXEMPLES DE NOS DIFFÉRENTS CAS !**

- **Température :**

- Capteur (DHT11) : 23°C
- Qualité : Optimale
- Actionneur 1 : Serre Ouverte/Fermer
- Actionneur 2 : Système chauffant ON/OFF

- **Humidité :**

- Capteur (SEN0193) : 75%
- Qualité : Optimale
- Actionneur : Activer l'arrivée d'eau

- **Luminosité :**

- Capteur (LDR NSL-19M51) : 30 kilos lux
- Qualité : Optimale
- Actionneur : Allumée / Éteinte

- **Types de problèmes possibles :**

- Niveau d'eau trop bas (réservoir)
- Température ambiante trop haute / trop basse
- Humidité ambiante trop basse / trop haute
- Humidité du sol trop haute / trop basse
- Luminosité trop faible / trop élevée
- Actionneur bloqué / en panne
- etc.

- **Réalisation de l'IHM :**

**Qu'est ce qu'une IHM ?**

IHM signifie interface homme-machine et fait référence à un tableau de bord qui permet à un utilisateur de communiquer avec une machine, un programme informatique ou un système. Les IHM affichent des données en temps réel et permettent à l'utilisateur de contrôler les machines grâce à une interface utilisateur graphique. (source : copadata.com)

## Page ACCUEIL :

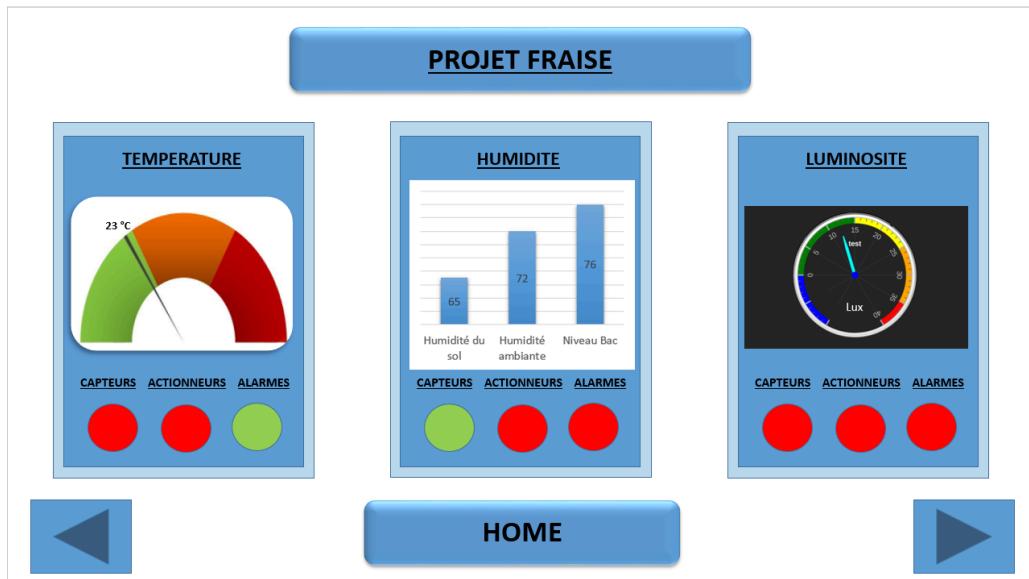
La page d'accueil a les trois informations des capteurs de notre fraisier, *la température, l'humidité et la luminosité*.

En dessous de chaque information il y a trois LEDs qui nous informent visuellement en vert ou rouge , si :

- la condition d'un capteur (n') est (pas) remplie
- un actionneur est en marche
- il y a un problème, si un actionneur est bloqué

Il y a des représentations graphiques de la valeur de chaque capteur sous forme de diagramme.

Les flèches en bas de l'écran permettent de naviguer entre chaque page.



Nous pourrions changer le tout, en y mettant une page plus agréable pour un utilisateur, comme la pousse de fraise, une sorte de croissance de la plante où l'on pourrait y voir un schéma s'accroître.

L'option aussi d'un menu déroulant, permettant de pouvoir naviguer entre les pages plus facilement.

L'objectif serait de rendre l'IHM plus attractif mais aussi simple d'utilisation pour les utilisateurs.

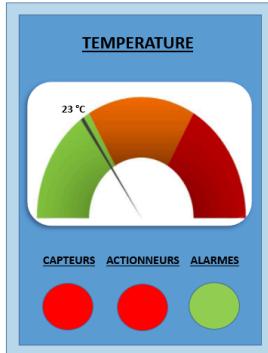
## Page TEMPÉRATURE / HUMIDITÉ / LUMINOSITÉ :

Chaque capteur possède sa propre page avec plus d'informations que sur la page d'accueil, notamment avec les zones de texte, où l'on peut y lire les éventuels problèmes, avec la LED correspondante qui est allumée.

On peut également y visionner la valeur relevée d'un capteur. Les valeurs en bleu sont variables en temps réel.

Dans la ligne des actionneurs, il y a un ou deux boutons qui nous permettent de gérer l'actionneur en cas de besoin selon la valeur envoyée par le capteur, sinon, le programme fera l'action seule. Cela nous permet de gérer à distance si nous le souhaitons.

**PROJET FRAISE**



**TEMPERATURE**

23 °C

CAPTEURS ACTIONNEURS ALARMES

**CAPTEURS :**

TEMPERATURE AMBIANTE : 23 °C

**ACTIONNEURS :**

OUVRIR/FERMER LA SERRE  
ALLUMER LE SYSTÈME CHAUFFANT

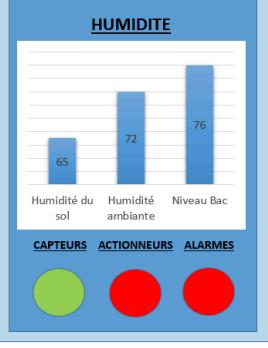
Ouvrir Fermer      Allumer Eteindre

**ALARMS :**

TEMPERATURE TROP FROIDE/CHAUDE  
LE SYSTÈME CHAUFFANT EN PANNE  
SERRE BLOQUEE SUR OUVERTURE/FERMETURE

◀TEMPERATURE▶

**PROJET FRAISE**



**HUMIDITE**

Humidité du sol	65
Humidité ambiante	72
Niveau Bac eau	76

CAPTEURS ACTIONNEURS ALARMES

**CAPTEURS :**

HUMIDITE DU SOL : 65 %  
HUMIDITE AMBIANTE : 72 %  
NIVEAU BAC EAU : 76 %

**ACTIONNEURS :**

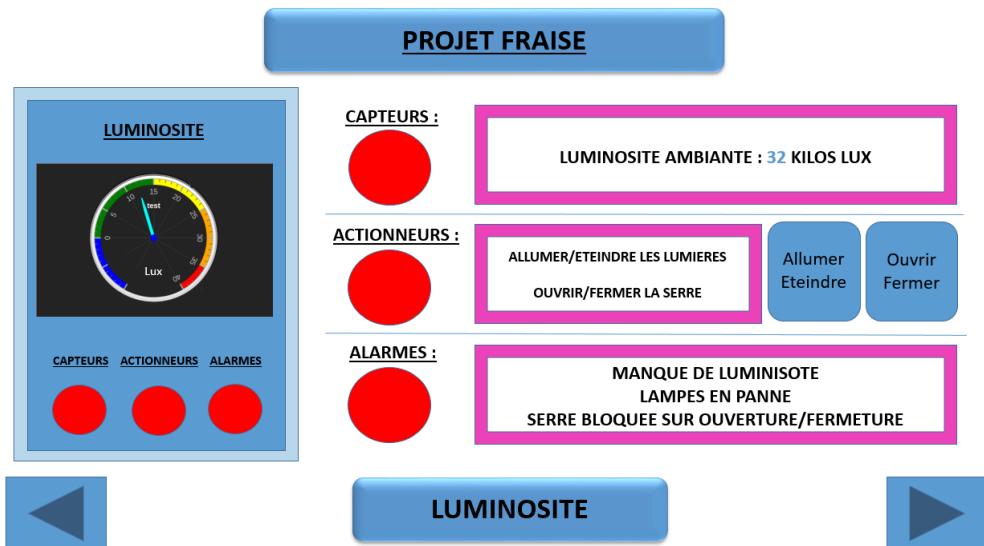
OUVRIR/FERMER L'ARRIVEE D'EAU

Ouvrir Fermer

**ALARMS :**

PAS ASSEZ/TROP HUMIDITE  
MANQUE EAU DANS LE BAC  
ARRIVEE D'EAU BLOQUEE

◀HUMIDITE▶



### Page HISTORIQUE :

Sur la dernière page, il y a un tableau qui nous permet de garder un historique des données relevées par nos capteurs.

Ces valeurs sont associées avec la date et l'heure d'activation de l'actionneur permettant d'adapter nos besoins à notre plantation, le tout complété avec une alarme en cas de problème où alors d'une notification.

**PROJET FRAISE**

DATE	HEURE	VALEUR CAPTEUR	ACTIONNEUR ACTIVE	ALARMS
25/09/24	10h29	23 %	POMPE A EAU	MANQUE D'EAU
02/10/24	22h46	2k LUX	X	LAMPES EN PANNE
12/10/24	07h05	62° C	SERRE	OUVERTURE SERRE
12/10/24	09h12	52° C	SERRE	FERMETURE SERRE

◀
HISTORIQUE
▶

## **Conclusion de la séance :**

Cette séance nous devions développer une interface homme-machine (IHM) permettant de surveiller et de contrôler les conditions de la serre. La page d'accueil présente les valeurs en temps réel des capteurs de température, d'humidité et de luminosité. En dessous de chaque information, des LEDs indiquent visuellement si les conditions sont optimales, si un actionneur est en marche ou si un problème est détecté, comme un actionneur bloqué. Des graphiques sont également présents pour suivre l'évolution des données. La navigation entre les pages est simplifiée grâce à des flèches et un menu déroulant.

Chaque capteur dispose d'une page dédiée, où des informations détaillées sont affichées. Cette page permet également de contrôler les actionneurs manuellement si nécessaire, bien que le système prenne généralement les décisions automatiquement. Enfin, une page historique permet de consulter les données enregistrées, avec les dates et heures des relevés, ainsi que l'activation des actionneurs. Cette page inclut également un système d'alerte pour notifier l'utilisateur en cas de problème. L'objectif est de rendre l'IHM à la fois simple à utiliser et visuellement attrayante, tout en offrant un contrôle efficace sur la serre.

# Séance 4 - 09.10.2024

## • Communication Grove WiFi 8266 et Arduino

### Objectif de l'utilisation en WiFi :

L'objectif de cette séance est de comprendre le fonctionnement du grove wifi 8266.

Nous utiliserons ce module pour récupérer et lire en direct les données de nos capteurs connectés à l'arduino, que l'on mettra dans un tableau disponible sur l'IHM.

### 1. Datasheet et documentation pour le GROVE WIFI 8266 :

<https://docs.google.com/document/d/1ub3WuRrp1F8QijoHCDjOez6SC1mtG5pQglhiPdeB12k/edit>

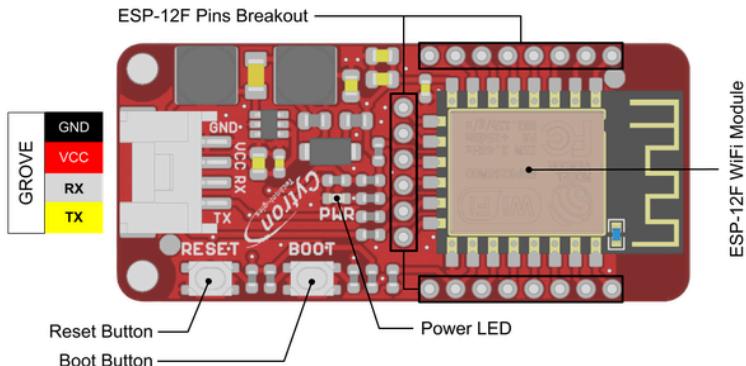


Figure 1: GRV-WIFI-8266 Board Functions

Function	Description
Grove Port	Connect to the host controller. <ul style="list-style-type: none"><li>• GND : Ground</li><li>• VCC : Positive Supply (3V - 6V)</li><li>• RX : Serial Rx pin for ESP8266. Connect to host's Tx.</li><li>• TX : Serial Tx pin for ESP8266. Connect to host's Rx.</li></ul>
Reset Button	Press to reset the ESP8266.
Boot Button	Press and hold this button while resetting the ESP8266 will enter the bootloader mode. Used to update the firmware or program the ESP8266.
Power LED	Turn on when powered up.
ESP-12F Pins Breakout	Breakout of the pins for ESP-12F module. Refer to the bottom of the PCB for pinout.
ESP-12F WiFi Module	Ai-Thinker ESP-12F WiFi module. Powered by the ESP8266.

Table 1: GRV-WIFI-8266 Board Functions

No	Parameters	Min	Max	Unit
1	Power Input Voltage (VCC)	3.0	6.0	V
2	Digital Input Voltage (Rx & GPIOs)	Low Level	-0.3	0.8 V
		High Level	2.5	3.6 V
3	Digital Output Voltage (Tx & GPIOs)	Low Level	0	0.4 V
		High Level	2.6	3.3 V
4	Operating Temperature	-20	85	°C

Table 2: GRV-WIFI-8266 Specifications

## 2. Connexion du module Grove WiFi 8266 à l'Arduino :

- Commande AT :

[https://wiki.seeedstudio.com/Grove-UART\\_Wifi\\_V2/#features](https://wiki.seeedstudio.com/Grove-UART_Wifi_V2/#features)

- Exemples de code que l'on a essayé :

<https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>

```
#include <ESP8266WiFi.h>

void setup()
{
    Serial.begin(115200);
    Serial.println();

    WiFi.begin("network-name", "pass-to-network");

    Serial.print("Connecting");
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println();

    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());
}

void loop() {}
```

### Problème :

Arduino ne reconnaît pas la fonction WiFi.begin() car WiFi n'est pas initialisé dans la bibliothèque.h

Exemple avec la bibliothèque fournie de base avec Arduino.

```

#include <SoftwareSerial.h>

SoftwareSerial mySerial(0,1); // RX, TX

void setup() {
    // Open serial communications and wait for port to open:
    Serial.begin(9600);
    while (!Serial) {
        ; // wait for serial port to connect. Needed for native USB port only
    }

    Serial.println("Goodnight moon!");

    // set the data rate for the SoftwareSerial port
    mySerial.begin(9600);
    mySerial.println("AT");
}

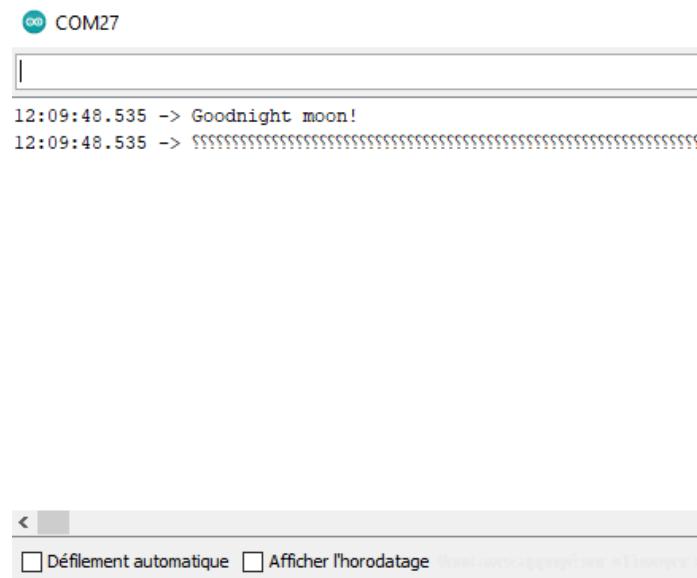
void loop() { // run over and over
    if (mySerial.available()) {
        Serial.write(mySerial.read());
    }
    if (Serial.available()) {
        mySerial.write(Serial.read());
    }
}

```

### Problème :

Dans le serial moniteur, nous avons l'affichage de "Goodnight moon!" étant qu'il s'agit d'un simple affichage.

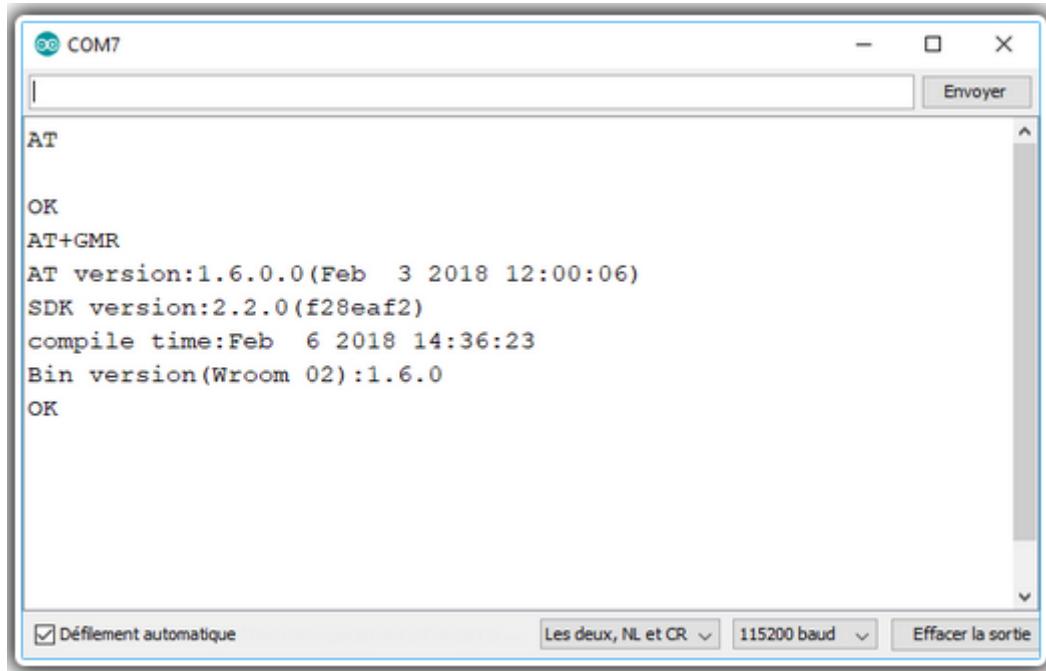
Néanmoins, nous avons par la suite une suite de '?' qui s'affiche. Voir ci-dessous.



Il faut que l'on trouve une solution pour que le moniteur série nous renvoie Ok lorsque l'on met la commande AT+RST. Cette fonction permet de reset la carte avant que l'on commence à travailler.

Voici le résultat attendu :

<https://f-leb.developpez.com/tutoriels/arduino/esp8266/debuter/>



Nous avons essayé divers programme, où l'on peut essayer de paramétrier la connexion en y mettant l'host et le mot de passe de notre partage de connexion mais notre carte affiche en continu que la connexion n'est pas réussie.

Peut-être un problème comme nous l'a dit notre professeure avec le fait qu'il faille flasher nos Grove WiFi.

Durant cette séance, nous avons rien réussi de spécifique appart essayé divers programme, exemple dans des bibliothèque ou sur des forum.

Néanmoins, nous avons pu en savoir plus sur le fonctionnement du composant via la datasheet, les commandes AT afin de permettre la liaison et la communication d'informations et comprendre le fonctionnement de notre démarche.

Pour rappel, nous aurons pour objectif, d'utiliser ce composant afin de pouvoir envoyé nos informations des mesures de nos divers capteurs pour ce projet, qui seront lu par la carte arduino sur notre mobile, dans une IHM avec un tableau et interface comme nous avons pu faire la séance précédente.

Nous continuerons donc d'essayer et de trouver une solution afin d'atteindre la connexion et la communication entre notre carte et notre point final, durant la prochaine séance.

### **Conclusion de la séance :**

Lors de cette séance, l'objectif était de comprendre le fonctionnement du module Grove WiFi 8266 pour récupérer les données des capteurs connectés à l'Arduino et les afficher dans un tableau sur l'IHM. Nous avons consulté la documentation et essayé plusieurs exemples de code pour configurer la connexion WiFi via la commande AT. Cependant, nous avons rencontré des problèmes techniques. L'Arduino ne reconnaissait pas la fonction `WiFi.begin()` et la carte ne parvenait pas à se connecter malgré nos tentatives de configuration avec le mot de passe et l'hôte. Il semble que le Grove WiFi nécessite un flashage préalable, comme suggéré par notre professeur.

Bien que nous n'ayons pas résolu le problème cette fois-ci, cette séance nous a permis de mieux comprendre le fonctionnement du module et des commandes AT. L'objectif reste de connecter l'Arduino au réseau pour envoyer les données des capteurs à l'IHM via WiFi.

Nous poursuivrons nos essais lors de la prochaine séance pour réussir cette communication.

## Séance 5 - 16.10.2024

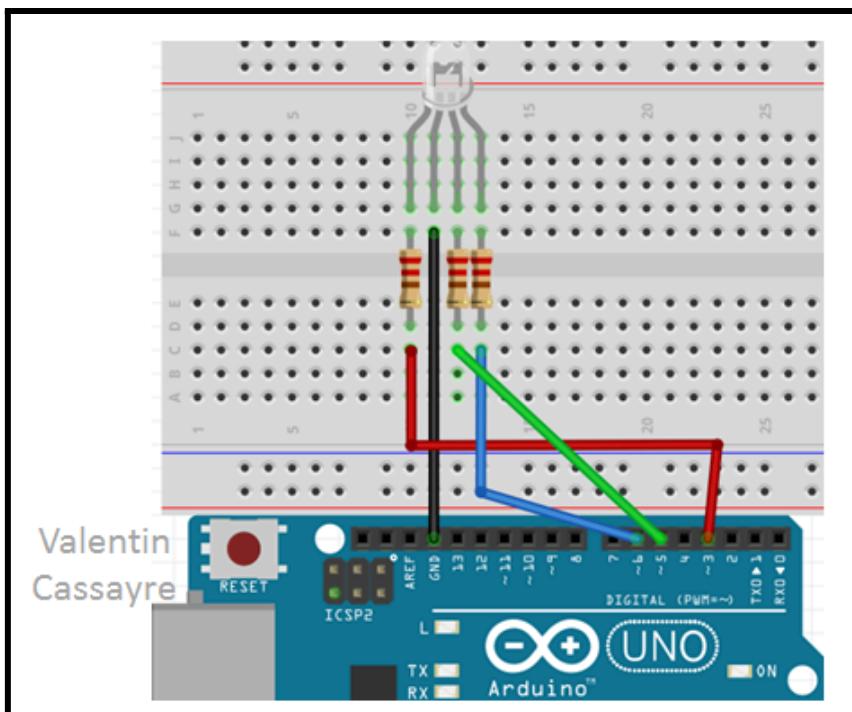
Durant cette séance, nous avons fait le test de nos capteurs en physique, en les connectant à une carte Arduino.

De plus, nous avons fait l'ajout d'une LED RGB afin de montrer directement quand nous avons le manque d'humidité/luminosité/chaleur, le surplus ou la quantité suffisante.

Nous pourrons ainsi, selon la couleur révélée, activer ou désenclencher un actionneur.

- **LED RGB :**

<https://projecthub.arduino.cc/semssemharaz/interfacing-rgb-led-with-arduino-b59902>



- Doc technique de la LED :

<https://docs.rs-online.com/ad97/0900766b8139d70b.pdf>

Nous prenons comme résistance RGB :

- RED = 200 Ohms
- GREEN = 220 Ohms
- BLUE = 220 Ohms

Voici le programme qui lui correspond :

```
// Fonction led rgb
int redPin= 5;
int greenPin = 7;
int bluePin = 6;

void setup() {
    //Defining the pins as OUTPUT
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}

void loop() {
    setColor(0,255, 0); // GREEN Color
    delay(1000);
}

void setColor(int redValue, int greenValue, int blueValue) {
    analogWrite(redPin, redValue);
    analogWrite(greenPin, greenValue);
    analogWrite(bluePin, blueValue);
}
```

Après avoir tester le programme de la LED RGB, nous avons fusionné l'intégralité de nos programmes dans le même code, afin de pouvoir tester les capteurs ensemble.

- **Programme TEST UNION :**

```
// -----
//      Union des Capteurs
// -----
#include <DFRobot_DHT11.h>
DFRobot_DHT11 DHT;

#define Soil_Moisture_Sensor A0 // Humidite sol
#define DHT11_PIN 2 // Corresponds à la broche A1 ou est connecté le capteur de
température A1
#define LDR A2 // composante photoresistor sur la pin A2

#define MaxSol 600
#define MinSol 400

#define MaxLum 450
#define MinLum 300

#define MaxTemp 22
#define MinTemp 10

int redPin= 5;
int greenPin = 7;
int bluePin = 6;

void setup() {
    // initialise la communication avec le PC
    Serial.begin(9600);

    // initialise les broches
    pinMode(LDR, INPUT);
    pinMode(Soil_Moisture_Sensor, INPUT);

    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}
```

```

void loop() {
    int value_Luminosite = Eclairage(); // Lecture pour la luminosité
    float pourcentage_Luminosite = (value_Luminosite / 679.0) * 100;

    Serial.print("Eclairage : "); Serial.print(value_Luminosite);
    Serial.print(" - En pourcentage : "); Serial.print(pourcentage_Luminosite); Serial.println("%");

    int value_Sol = Sol(); // Lecture pour le sol
    float pourcentage_Sol = (value_Sol / 876.0) * 100;

    Serial.print("Humidite sol : "); Serial.print(value_Sol);
    Serial.print(" - En pourcentage : "); Serial.print(pourcentage_Sol); Serial.println(" %");

    DHT.read(DHT11_PIN);

    //float voltage = 5.0 /1024 * analogRead(Temperature_Sensor); // On lit la valeur du
    capteur
    //float temperature = voltage * 100 - 50 ; // On transforme cette tension en température en
    degréé
    Temp(DHT.temperature);

    Serial.print("Temperature ambiante : "); Serial.print(DHT.temperature); Serial.println("degrees");

    Serial.println("-----"); // Séparation entre chaque lecture

    delay(1000);
}

// -----
//      Fonctions
// -----


// fonction lumiere

```

```

int Eclairage(){
    int value = analogRead(LDR);      // mesure la tension sur la broche A2

    if(value < MinLum){ // Inférieur à 400 = manque de lumière => Activer éclairage
        Serial.print("Peu de lumenosite --> ");
        setColor(255, 0, 0); // Red Color - ACTIVE ACTIONNEUR = ALLUMER LUMIERE
    }
    else if (value > MaxLum){ // Supérieur à 500 = Trop de lumière => Désactiver éclairage
        Serial.print("Trop de lumenosite --> ");
        setColor(0, 0, 255); // BLUE Color
    }
    else { // Bon intervalle pour la fraise = Bonne luminosité => Désactiver éclairage
        Serial.print("Bonne lumenosite --> ");
        setColor(0, 255, 0); // green Color - ETEINDRE LA LUMIERE
    }

    return(value);
}

// fonction Humidite Sol
int Sol(){
    int value = analogRead(Soil_Moisture_Sensor); // mesure la tension sur la broche A0

    if(value < MinSol){ // Inférieur à 400 = Peu humide => Activer arrosage
        Serial.print("Sol Peu humide --> ");
        //setColor(255, 0, 0); // Red Color
    }
    else if (value > MaxSol){ // Supérieur à 500 = Trop humide => Désactiver arrosage
        Serial.print("Sol Trop humide --> ");
        //setColor(0, 255, 0); // green Color
    }
    else { // Bon intervalle pour la fraise = Bonne humidité du sol => Désactiver arrosage
        Serial.print("Sol bien humidifie --> ");
        //setColor(0, 0, 255); // blue Color
    }

    return(value);
}

```

return(value);

```

}

// Fonction temperature air
int Temp(int value){
    if(value < MinTemp){ // Inférieur à 0 = Tres froid
        Serial.print("Trop froid --> ");
        //setColor(255, 0, 0); // Red Color
    }
    else if (value > MaxTemp){ // Supérieur à  = Tres chaud
        Serial.print("Trop chaud --> ");
        //setColor(0, 255, 0); // green Color
    }
    else { // Bon intervalle pour la fraise = frais
        Serial.print("Air parfait --> ");
        //setColor(0, 0, 255); // blue Color
    }

    return(value);
}

// Fonction RGB LED
void setColor(int redValue, int greenValue, int blueValue) {
    analogWrite(redPin, redValue);
    analogWrite(greenPin, greenValue);
    analogWrite(bluePin, blueValue);
}

```

Nous obtenons ainsi les résultat suivant :

```
-----
Bonne luminosite --> Eclairage : 416 - En pourcentage : 61.27 %
Sol bien humidifie --> Humidite sol : 535 - En pourcentage : 61.07 %
Trop chaud --> Temperature ambiante : 25 degrees
-----
Bonne luminosite --> Eclairage : 436 - En pourcentage : 64.21 %
Sol bien humidifie --> Humidite sol : 533 - En pourcentage : 60.84 %
Trop chaud --> Temperature ambiante : 25 degrees
-----
Trop de luminosite --> Eclairage : 456 - En pourcentage : 67.16 %
Sol bien humidifie --> Humidite sol : 534 - En pourcentage : 60.96 %
Trop chaud --> Temperature ambiante : 26 degrees
-----
```

Néanmoins, nous pouvons constater un problème en lien avec le capteur d'humidité du sol.

Lorsque l'on laisse le capteur dans l'air ambiant, nous avons une humidité d'environ 60-70%, mais lorsque l'on fait le test dans gourde remplie d'eau, nous arrivons à un résultat inférieur d'environ 30-40%. Ce qui devrait plutôt l'inverse, nous devons ainsi corriger cela.

### **Conclusion :**

Après avoir fait nos test, nous avons pu voir que notre capteur de température DHT11, ne fonctionnait pas avec notre code, nous avons donc changé le code étant donné que celui que nous avons de base était utilisé sur TinkerCard et que nous avions pas de DHT11 sur l'outil en question.

Au lieu de mesurer une tension et de la convertir, nous utilisons directement la bibliothèque DHT afin de relever la valeur.

- Voici le code en question :

```
#include <DFRobot_DHT11.h>
DFRobot_DHT11 DHT;
#define DHT11_PIN 10
```

```
void setup(){
  Serial.begin(115200);
}
```

```

void loop(){
    DHT.read(DHT11_PIN);
    Serial.print("temp:");
    Serial.print(DHT.temperature);
    Serial.print(" humi:");
    Serial.println(DHT.humidity);
    delay(1000);
}

```

Nous avons câblé le capteur de température, humidité et luminosité.

On a dû changer la résistance de la photorésistance car nos valeurs en lux étaient bien trop grandes alors qu'il n'y avait pas beaucoup de luminosité. On est passé de 15 kohm à 10 kohm car il n'y a pas assez de courant dans la photorésistance. On a également codé les 3 capteurs ensemble et on a ajouté une LED RGB qui nous affiche visuellement la luminosité détectée (Rouge → Pas assez de luminosité ,Vert → Bonne luminosité ,Bleu→ Trop de luminosité).

**Electrical / Optical Characteristics at TA=25°C**

Symbol	Parameter	Device	Typ.	Max.	Units	Test Conditions
$\lambda_{peak}$	Peak Wavelength	Hyper Red Blue Green	645 460 520		nm	$I_F=20mA$
$\lambda_D [1]$	Dominant Wavelength	Hyper Red Blue Green	630 465 525		nm	$I_F=20mA$
$\Delta\lambda/2$	Spectral Line Half-width	Hyper Red Blue Green	25 25 35		nm	$I_F=20mA$
C	Capacitance	Hyper Red Blue Green	45 100 100		pF	$V_F=0V;f=1MHz$
$V_F [2]$	Forward Voltage	Hyper Red Blue Green	1.9 3.3 3.2	2.5 4 4	V	$I_F=20mA$
$I_R$	Reverse Current	Hyper Red Blue Green		10 50 50	uA	$V_R=5V$

On a pris la ligne Forward Voltage et le courant correspondant :

- RED =  $1.9 / 20mA = 95$  Ohms
- GREEN =  $3.3 / 20mA = 165$  Ohms
- BLUE =  $3.2 / 20mA = 160$  Ohms

Afin de ne pas avoir de problèmes avec la RGB LED, nous avons pris des résistances plus importantes tout en gardant la résistance pour la broche du RED plus faible que les autres.

- RED = 200 Ohms
- GREEN & BLUE = 220 Ohms

Nous avons pu constater une solution pour notre problème avec le capteur d'humidité, dans la datasheet, nous avons l'information suivante :

[https://wiki.dfrobot.com/Capacitive\\_Soil\\_Moisture\\_Sensor\\_SKU\\_SEN0193](https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193)

- Dry: (520 430]
- Wet: (430 350]
- Water: (350 260]

Nous savons donc qu'en milieu sec nous avons une valeur comprise entre 430 et 520, et en milieu humide une valeur entre 260 et 350.

Afin de pallier ce problème, nous allons faire une différence.

- **100 - Value = Humidité Exacte**

Ce qui nous donne le résultat suivant :

En air ambiant :

```
Sol bien humidifie --> Humidite sol : 536 - En pourcentage : 38.81 %
Trop chaud --> Temperature ambiante : 26 degrees
-----
Peu de lumenosite --> Eclairage : 281 - En pourcentage : 41.38 %
Sol bien humidifie --> Humidite sol : 546 - En pourcentage : 37.67 %
Trop chaud --> Temperature ambiante : 26 degrees
-----
Bonne luminosite --> Eclairage : 304 - En pourcentage : 44.77 %
Sol bien humidifie --> Humidite sol : 536 - En pourcentage : 38.81 %
Trop chaud --> Temperature ambiante : 26 degrees
-----
Bonne luminosite --> Eclairage : 306 - En pourcentage : 45.07 %
Sol bien humidifie --> Humidite sol : 536 - En pourcentage : 38.81 %
Trop chaud --> Temperature ambiante : 26 degrees
```

Dans la gourde :

```
Sol Peu humide --> Humidite sol : 313 - En pourcentage : 64.27 %
Trop chaud --> Temperature ambiante : 26 degrees
-----
Peu de lumenosite --> Eclairage : 298 - En pourcentage : 43.09 %
Sol Peu humide --> Humidite sol : 309 - En pourcentage : 64.73 %
Trop chaud --> Temperature ambiante : 26 degrees
-----
Peu de lumenosite --> Eclairage : 269 - En pourcentage : 39.62 %
Sol Peu humide --> Humidite sol : 298 - En pourcentage : 65.98 %
Trop chaud --> Temperature ambiante : 26 degrees
-----
Peu de lumenosite --> Eclairage : 244 - En pourcentage : 35.94 %
Sol Peu humide --> Humidite sol : 294 - En pourcentage : 66.44 %
Trop chaud --> Temperature ambiante : 26 degrees
```

Ce qui nous semble plus logique que le cas inverse que nous avions auparavant.

Nous l'avons fait de cette façon dans le code, afin d'obtenir ce résultat :

```
int value_Sol = Sol(); // Lecture pour le sol
float pourcentage_Sol = 100 - ((value_Sol / 876.0) * 100);

Serial.print("Humidite sol : "); Serial.print(value_Sol);
Serial.print(" - En pourcentage : "); Serial.print(pourcentage_Sol); Serial.println("%");
```

- Programme final après correction de notre problème :

```
#include <DFRobot_DHT11.h>
DFRobot_DHT11 DHT;

#define Soil_Moisture_Sensor A0 // Humidite sol
#define DHT11_PIN 2 // Corresponds à la broche A1 ou est connecté le capteur de
température A1
#define LDR A2 // composante photoresistor sur la pin A2

#define MaxSol 600
#define MinSol 400

#define MaxLum 450
#define MinLum 300

#define MaxTemp 22
#define MinTemp 10

int redPin= 5;
int greenPin = 7;
int bluePin = 6;

void setup() {
    // initialise la communication avec le PC
    Serial.begin(9600);

    // initialise les broches
    pinMode(LDR, INPUT);
```

```

pinMode(Soil_Moisture_Sensor, INPUT);

pinMode(redPin, OUTPUT);
pinMode(greenPin, OUTPUT);
pinMode(bluePin, OUTPUT);
}

void loop() {
    int value_Luminosite = Eclairage(); // Lecture pour la luminosité
    float pourcentage_Luminosite = (value_Luminosite / 679.0) * 100;

    Serial.print("Eclairage : "); Serial.print(value_Luminosite);
    Serial.print(" - En pourcentage : "); Serial.print(pourcentage_Luminosite); Serial.println("%");

    int value_Sol = Sol(); // Lecture pour le sol
    float pourcentage_Sol = 100 - ((value_Sol / 876.0) * 100);

    Serial.print("Humidite sol : "); Serial.print(value_Sol);
    Serial.print(" - En pourcentage : "); Serial.print(pourcentage_Sol); Serial.println(" %");

    DHT.read(DHT11_PIN);

    //float voltage = 5.0 /1024 * analogRead(Temperature_Sensor); // On lit la valeur du
    capteur
    //float temperature = voltage * 100 - 50 ; // On transforme cette tension en température en
    degrée
    Temp(DHT.temperature);

    Serial.print("Temperature ambiante : "); Serial.print(DHT.temperature); Serial.println(" degrees");

    Serial.println("-----"); // Séparation entre chaque lecture

    delay(1000);
}

```

```

// -----
//      Fonctions
// -----


// fonction lumiere
int Eclairage(){
    int value = analogRead(LDR);      // mesure la tension sur la broche A2

    if(value < MinLum){ // Inférieur à 400 = manque de lumière => Activer éclairage
        Serial.print("Peu de luminosite --> ");
        setColor(255, 0, 0); // Red Color - ACTIVE ACTIONNEUR = ALLUMER LUMIERE
    }
    else if (value > MaxLum){ // Supérieur à 500 = Trop de lumière => Désactiver éclairage
        Serial.print("Trop de luminosite --> ");
        setColor(0, 0, 255); // BLUE Color
    }
    else { // Bon intervalle pour la fraise = Bonne luminosité => Désactiver éclairage
        Serial.print("Bonne luminosite --> ");
        setColor(0, 255, 0); // green Color - ETEINDRE LA LUMIERE
    }

    return(value);
}

// fonction Humidite Sol
int Sol(){
    int value = analogRead(Soil_Moisture_Sensor); // mesure la tension sur la broche A0

    if(value < MinSol){ // Inférieur à 400 = Peu humide => Activer arrosage
        Serial.print("Sol Peu humide --> ");
        //setColor(255, 0, 0); // Red Color
    }
    else if (value > MaxSol){ // Supérieur à 500 = Trop humide => Désactiver arrosage
        Serial.print("Sol Trop humide --> ");
        //setColor(0, 255, 0); // green Color
    }
    else { // Bon intervalle pour la fraise = Bonne humidité du sol => Désactiver arrosage

```

```

Serial.print("Sol bien humidifié --> ");
//setColor(0, 0, 255); // blue Color
}

return(value);
}

// Fonction temperature air
int Temp(int value){
if(value < MinTemp){ // Inférieur à 0 = Tres froid
    Serial.print("Trop froid --> ");
    //setColor(255, 0, 0); // Red Color
}
else if (value > MaxTemp){ // Supérieur à  = Tres chaud
    Serial.print("Trop chaud --> ");
    //setColor(0, 255, 0); // green Color
}
else { // Bon intervalle pour la fraise = frais
    Serial.print("Air parfait --> ");
    //setColor(0, 0, 255); // blue Color
}

return(value);
}

// Fonction RGB LED
void setColor(int redValue, int greenValue, int blueValue) {
    analogWrite(redPin, redValue);
    analogWrite(greenPin, greenValue);
    analogWrite(bluePin, blueValue);
}

```

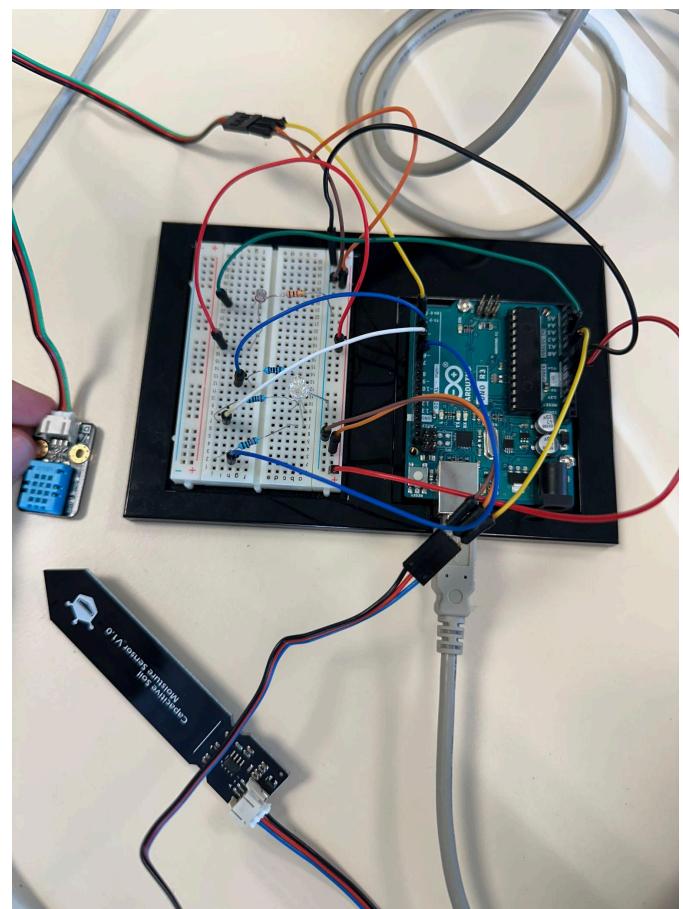
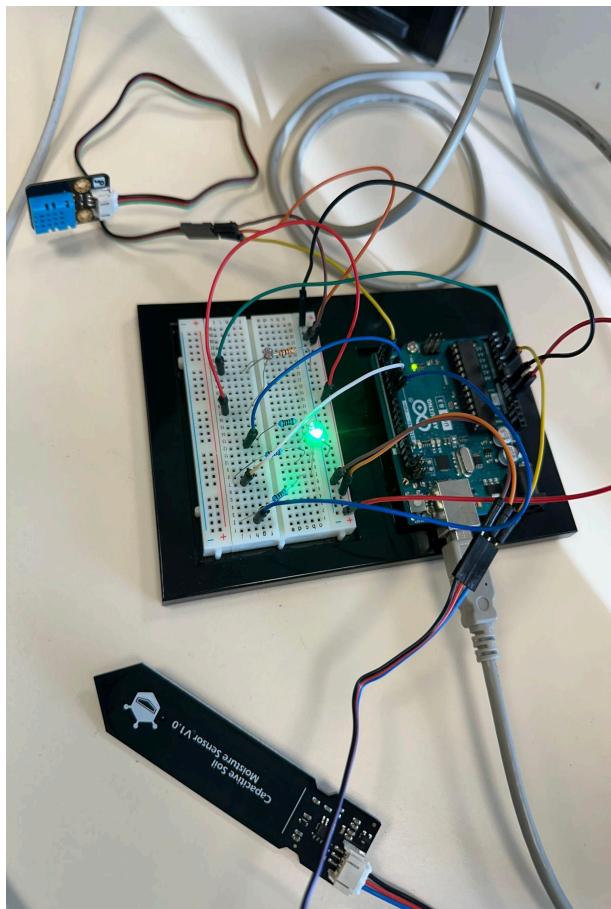
Pour la suite nous procéderons de la manière suivante avec les actionneurs :

Nous voulons que lorsque la LED RGB passe à une certaine couleur cela déclenche l'actionneur associé pour chaque capteur.

Exemple pour la luminosité :

(Rouge → Pas assez de luminosité ,Vert → Bonne luminosité ,Bleu→ Trop de luminosité)

LED Vert = Luminosité OK donc on remet à l'état initial.



## **Conclusion de la séance :**

Lors de cette séance, nous avons testé nos capteurs avec une carte Arduino, en ajoutant une LED RGB pour afficher les niveaux de luminosité, d'humidité et de température. La couleur de la LED indique si les paramètres sont trop faibles, trop élevés ou dans la plage idéale, permettant ainsi d'activer ou désactiver un actionneur en fonction des valeurs mesurées.

Nous avons rencontré un problème avec le capteur d'humidité, mais après avoir consulté la datasheet, nous avons ajusté le calcul pour obtenir des résultats plus cohérents. Le programme a été ajusté pour intégrer les trois capteurs, et la LED RGB change de couleur selon les seuils de luminosité, d'humidité et de température.

Finalement, nous avons corrigé le problème du DHT11 en utilisant la bibliothèque correcte. Le programme permet désormais de mesurer et d'afficher les valeurs, tout en contrôlant les actionneurs en fonction des résultats. Pour la suite, nous prévoyons d'utiliser la LED RGB pour déclencher des actions, comme allumer ou éteindre des appareils en fonction de la luminosité.

# Séance 6 - 23.10.2024

- Dispositif “GROVE 433 MHz Simple RF Link” :

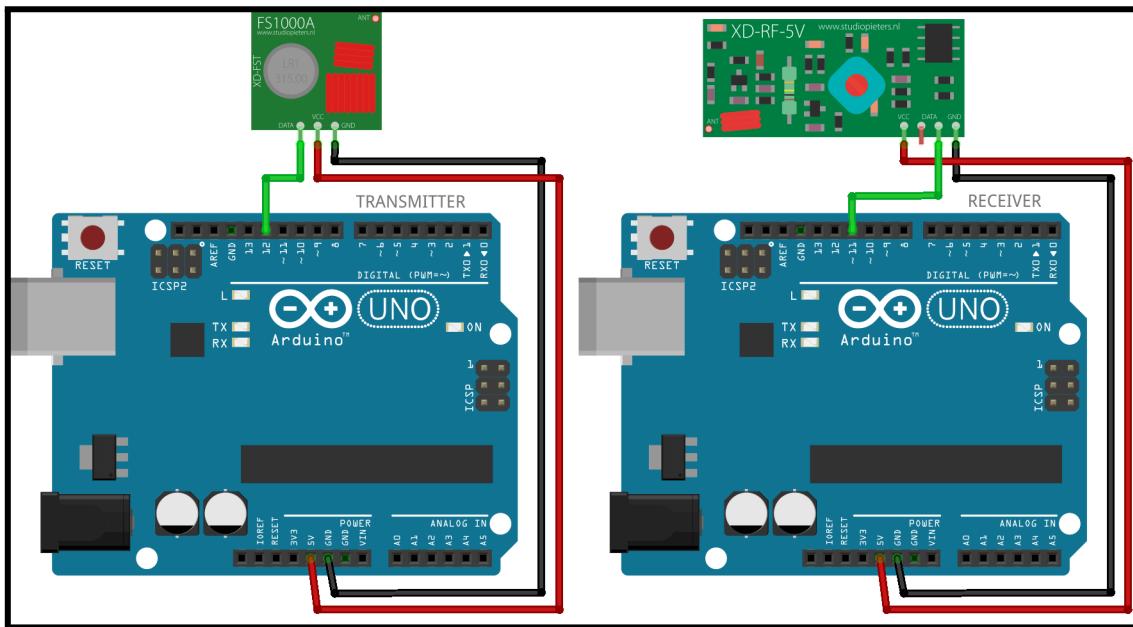
## 1) Recherche des informations :

- Fréquence d'utilisation
- Conformité avec les régulations
- Portée de transmission
- Exigences d'alimentation
- Comment Envoyer et Recevoir des messages avec ce kit
- etc.

La fréquence d'utilisation est de 433MHz.

La recommandation 1999/519/CE du Conseil du 12 juillet 1999 relative à la limitation de l'exposition du public aux champs électromagnétiques (de 0 à 300 GHz).

La portée de transmission est de 40 à 100 m.



- **Transmitter Module :**

Voltage :	Min -> 3V	Typical -> 5V	Max -> 12V
Current :	Min -> 3mA		Max -> 10mA
Transmit Power :	Min -> 3V	Typical -> 5V	Max -> 12V
Working Distance:	Min -> 40m		Max -> 100m

- **Receiver Module :**

Voltage :	Typical -> 5V
Current :	Typical -> 5mA
Receiver Sensitivity :	Typical -> -105 dBm
Operating Frequency :	Typical -> 433.92 MHz

## 2) **Etablir une connexion entre E&R (PTP) depuis l'Arduino :**

- Programme simple pour vérifier une connexion et échange des messages

Pour envoyer et recevoir des messages avec le kit Grove 433 MHz, suivre ces étapes :

- **Programme pour l'envoi :**

```
#include <VirtualWire.h>

int RF_TX_PIN = 2;

void setup(){
    vw_set_tx_pin(RF_TX_PIN); // Setup transmit pin
    vw_setup(2000); // Transmission speed in bits per second.
}

void loop() {
    const char *msg = "hello";
    vw_send((uint8_t *)msg, strlen(msg)); // Send 'hello' every 400ms.
    delay(400);
}
```

- Programme pour la réception :

```
#include <VirtualWire.h>

int RF_RX_PIN = 2;

void setup(){
    Serial.begin(9600);
    Serial.println("setup");

    vw_set_rx_pin(RF_RX_PIN); // Setup receive pin.
    vw_setup(2000); // Transmission speed in bits per second.
    vw_rx_start(); // Start the PLL receiver.
}

void loop(){
    uint8_t buf[VW_MAX_MESSAGE_LEN];
    uint8_t buflen = VW_MAX_MESSAGE_LEN;

    if(vw_get_message(buf, &buflen)) { // non-blocking I/O
        int i;

        Serial.print("Got: "); // Message with a good checksum received, dump HEX

        for(int i = 0; i < buflen; ++i) {
            Serial.print((char)buf[i]); // Convertir en caractère
            //Serial.print(buf[i], HEX);
        }
        Serial.println("");
    }
}
```

### **3) Re-connecter vos capteurs et définir le format des messages pour récupérer :**

- Température
- Humidité
- Lumière
- Soil moisure

#### **• Notre programme combiné (EMISSION + MESURER VIA CAPTEUR)**

```
#include <VirtualWire.h>
#include <DFRobot_DHT11.h>

DFRobot_DHT11 DHT;

#define Soil_Moisture_Sensor A0 // Humidité sol
#define DHT11_PIN 11 // Broche où est connecté le capteur de température
#define LDR A2 // Photoresistor sur la broche A2

#define MaxSol 600
#define MinSol 400

#define MaxLum 450
#define MinLum 300

#define MaxTemp 22
#define MinTemp 10

int RF_TX_PIN = 2;

int redPin = 5;
int greenPin = 7;
int bluePin = 8;

void setup() {
    Serial.begin(9600);

    pinMode(LDR, INPUT);
```

```

pinMode(Soil_Moisture_Sensor, INPUT);

pinMode(redPin, OUTPUT);
pinMode(greenPin, OUTPUT);
pinMode(bluePin, OUTPUT);

vw_set_tx_pin(RF_TX_PIN); // Setup transmit pin
vw_setup(1999); // Réduire la vitesse de transmission à 2000 bps
}

void loop() {
    int value_Luminosite = Eclairage(); // Lecture pour la luminosité
    int pourcentage_Luminosite = (value_Luminosite / 679.0) * 100;

    Serial.print("Eclairage : ");
    Serial.print(value_Luminosite);
    Serial.print(" - En pourcentage : ");
    Serial.print(pourcentage_Luminosite);
    Serial.println(" %");

    int value_Sol = Sol(); // Lecture pour le sol
    int pourcentage_Sol = 100 - ((value_Sol / 876.0) * 100);

    Serial.print("Humidité sol : ");
    Serial.print(value_Sol);
    Serial.print(" - En pourcentage : ");
    Serial.print(pourcentage_Sol);
    Serial.println(" %");

    DHT.read(DHT11_PIN);

    Serial.print("Température ambiante : ");
    Serial.print(DHT.temperature);
    Serial.println(" degrés");

    Serial.print("Humidité ambiante : ");
    Serial.print(DHT.humidity);
}

```

```

Serial.println(" %");

Serial.println("-----"); // Séparation entre chaque lecture

// Convertir les données en chaîne de caractères
char msg[100]; // Assurez-vous que le tableau est assez grand
sprintf(msg, "%d %d %d %d", pourcentage_Luminosite, DHT.temperature, DHT.humidity,
pourcentage_Sol);

Serial.print("Envoi: ");
Serial.println(msg); // Affiche le message envoyé
vw_send((uint8_t *)msg, strlen(msg)); // Envoyer le message

Serial.println("-----"); // Séparation entre chaque lecture

delay(200);
}

// -----
//      Fonctions
// -----


// fonction lumière
int Eclairage() {
    int value = analogRead(LDR); // mesure la tension sur la broche A2

    if (value < MinLum) { // Inférieur à 400 = manque de lumière => Activer éclairage
        Serial.print("Peu de luminosité --> ");
        setColor(255, 0, 0); // Red Color - ACTIVE ACTIONNEUR = ALLUMER LUMIERE
    }
    else if (value > MaxLum) { // Supérieur à 500 = Trop de lumière => Désactiver éclairage
        Serial.print("Trop de luminosité --> ");
        setColor(0, 0, 255); // BLUE Color
    }
    else { // Bon intervalle pour la fraise = Bonne luminosité => Désactiver éclairage
        Serial.print("Bonne luminosité --> ");
        setColor(0, 255, 0); // Green Color - ÉTEINDRE LA LUMIERE
    }
}

```

```

        }
        return value;
    }

// fonction Humidité Sol
int Sol() {
    int value = analogRead(Soil_Moisture_Sensor); // mesure la tension sur la broche A0

    if (value < MinSol) { // Inférieur à 400 = Peu humide => Activer arrosage
        Serial.print("Sol Peu humide --> ");
    }
    else if (value > MaxSol) { // Supérieur à 500 = Trop humide => Désactiver arrosage
        Serial.print("Sol Trop humide --> ");
    }
    else { // Bon intervalle pour la fraise = Bonne humidité du sol => Désactiver arrosage
        Serial.print("Sol bien humidifié --> ");
    }
    return value;
}

// Fonction RGB LED
void setColor(int redValue, int greenValue, int blueValue) {
    analogWrite(redPin, redValue);
    analogWrite(greenPin, greenValue);
    analogWrite(bluePin, blueValue);
}

```

- **Exemple du résultat obtenu :**

```

Peu de luminosité --> Eclairage : 290 - En pourcentage : 42 %
Sol bien humidifié --> Humidité sol : 536 - En pourcentage : 38 %
Température ambiante : 24 degrés
Humidité ambiante : 44 %

-----
Envoi: 42 24 44 38

```

- Notre programme combiné (RÉCEPTION + STOCKAGE)

```
#include <VirtualWire.h>

int RF_RX_PIN = 2;

// Variables pour stocker les données reçues
int Luminosite = 0;
int Temperature = 0;
int HumiditeAir = 0;
int HumiditeSol = 0;

void setup() {
    Serial.begin(9600);
    Serial.println("setup");

    vw_set_rx_pin(RF_RX_PIN); // Configuration de la broche de réception.
    vw_setup(1999); // Vitesse de transmission en bits par seconde.
    vw_rx_start(); // Démarrer le récepteur PLL.
}

void loop() {
    uint8_t buf[100]; // Taille du buffer de réception
    uint8_t buflen = sizeof(buf);

    // Vérification s'il y a un message
    if (vw_get_message(buf, &buflen)) { // non-blocking I/O
        buf[buflen] = '\0'; // Terminer la chaîne par un nul pour l'affichage

        // Extraire les valeurs de luminosité, température et humidité
        sscanf((char*)buf, "%d %d %d %d", &Luminosite, &Temperature, &HumiditeAir,
        &HumiditeSol);

        // Afficher les valeurs stockées
        Serial.print("Luminosité : "); Serial.print(Luminosite); Serial.println(" %");
        Serial.print("Température ambiante : "); Serial.print(Temperature); Serial.println(" °C");
    }
}
```

```

Serial.print("Humidite ambiante : "); Serial.print(HumiditeAir); Serial.println(" %");

Serial.print("Humidité du sol : "); Serial.print(HumiditeSol); Serial.println(" %");
}

else {
    Serial.println("Aucun message reçu.");
}

Serial.println("-----"); // Séparation entre chaque lecture

delay(1000); // Petite pause pour éviter la surcharge de la sortie série
}

```

- **Exemple du résultat obtenu :**

```

Luminosité : 45 %
Température ambiante : 24 °C
Humidite ambiante : 44 %
Humidité du sol : 38 %
-----
```

- **Résumé des ajouts pour l'envoi :**

Dans le code d'émission, nous envoyons les valeurs que nous avons relevées grâce à nos capteurs afin de pouvoir les transmettre à l'autre carte Arduino.

Pour cela, nous avons appliqué la solution d'un tableau permettant de stocker tout en gardant la séparation entre chaque valeur.

- **Résumé des ajouts pour la réception :**

Dans le code de réception, nous récupérons les valeurs en prenant en compte les espaces entre chacun, permettant ainsi de savoir la limite de la taille de caractère de notre mesure.  
(espace = '\0')

Par la suite, nous stockons ses valeurs dans des variables qui leurs sont propres afin de pouvoir ainsi les réutiliser dans un tableau d'affichage, une IHM ou autre.

## **4) Conclusion :**

### **- Avantages**

Une communication et un affichage rapide entre des valeurs de capteurs et des microcontrôleurs. On peut organiser les valeurs que l'on récupère pour pouvoir les analyser par la suite et les stocker.

### **- Désavantage**

La portée de l'information est assez courte et est bloquée par des infrastructures. Les personnes sur la même fréquence que nous peuvent lire/ envoyer des données ce qui peut poser problème.

### **- Amélioration**

On pourrait faire une "chanel" sécurisé pour ne pas interférer et se protéger des autres ce qui nous donnerait une valeur plus sur. Augmenter la distance de communication sans pour autant impacter la qualité du signal.

### **- Possibilités**

On peut choisir une fréquence unique sur laquelle on communique afin de ne pas recevoir les messages des autres ou d' envoyer notre message à tout le monde. On peut également mettre un identifiant au début de notre message pour "prévenir " que notre message arrive ce qui nous permettra d'éliminer des signaux parasites.

### **● Lien du grove long range 868 MHz**

[https://wiki.seeedstudio.com/Grove\\_LoRa\\_Radio/](https://wiki.seeedstudio.com/Grove_LoRa_Radio/)

## **Conclusion de la séance :**

Lors de cette séance, nous avons utilisé le kit "GROVE 433 MHz Simple RF Link" pour envoyer et recevoir des données sans fil. Ce kit fonctionne à une fréquence de 433 MHz et permet de transmettre des informations sur une distance allant de 40 à 100 mètres. Nous avons vérifié la compatibilité du kit avec les régulations concernant l'exposition aux champs électromagnétiques.

Nous avons créé deux programmes avec Arduino : un pour envoyer un message ("hello") et l'autre pour le recevoir. Ensuite, nous avons ajouté des capteurs pour mesurer des paramètres comme la luminosité, la température et l'humidité, puis envoyer ces données via le module RF.

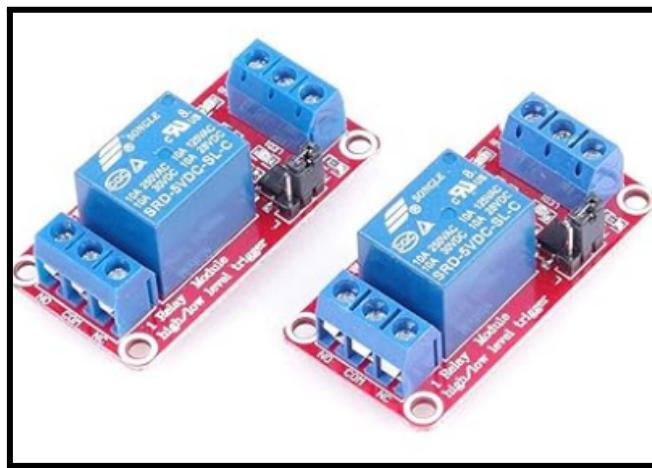
Le programme d'envoi lit les capteurs et envoie les valeurs sous forme de message, tandis que le programme de réception récupère ces données et les affiche. En conclusion, ce système permet une communication simple et rapide entre des capteurs et des microcontrôleurs. Cependant, la portée de transmission est limitée, et des interférences peuvent se produire si d'autres appareils utilisent la même fréquence. Pour améliorer la fiabilité, on pourrait utiliser des canaux sécurisés ou augmenter la portée avec des dispositifs comme le **Grove LoRa Radio 868 MHz**.

# Séance 7 - 12.11.2024

## Relay, Seleroval valve, battery 6V, diode

- 1) Proposer une connexion
- 2) Intégrer au système
- 3) Communication réseau

- Relay :



### Interface de module:

1. DC + : connecté au pôle positif de l'alimentation (la tension est requise par le relais)
2. DC - : connecté au pôle négatif de l'alimentation électrique
3. IN : Le relais de contrôle de haut ou bas niveau peut être engagé

### Borne de sortie relais:

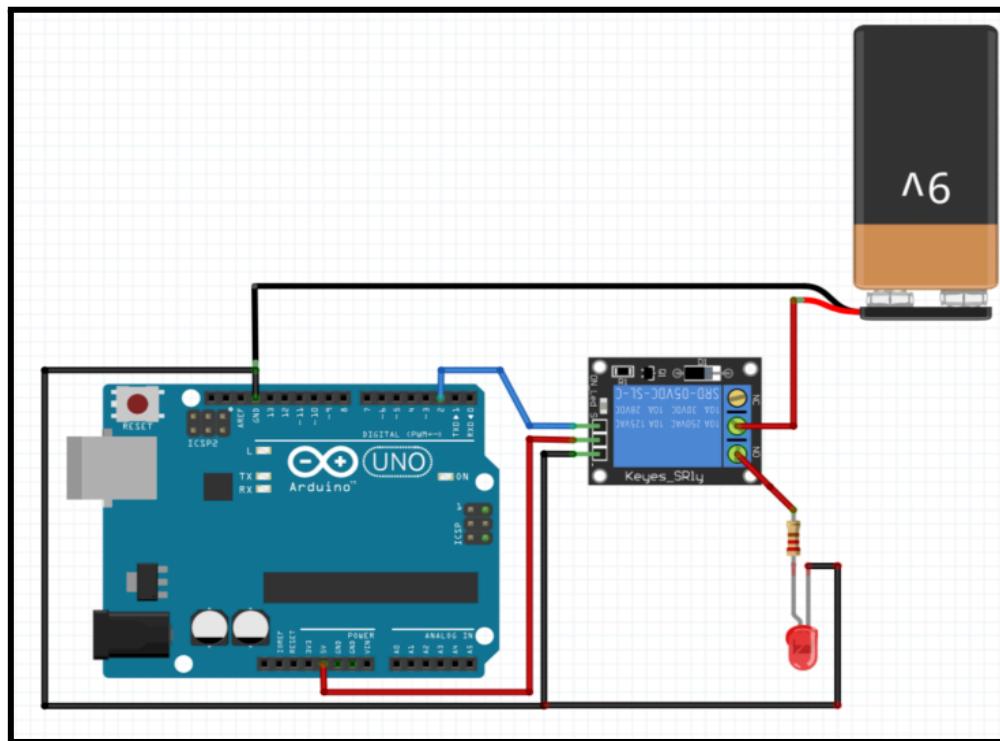
1. NO : interface normalement ouverte du relais; le relais est suspendu en l'air avant la fermeture et est court-circuité vers COM après la fermeture
2. COM : interface commune relais
3. NC : interface normalement fermée du relais, court-circuitée avec COM avant la fermeture du relais, suspendue après la fermeture

### Terminal de sélection de déclenchement de niveau haut et bas:

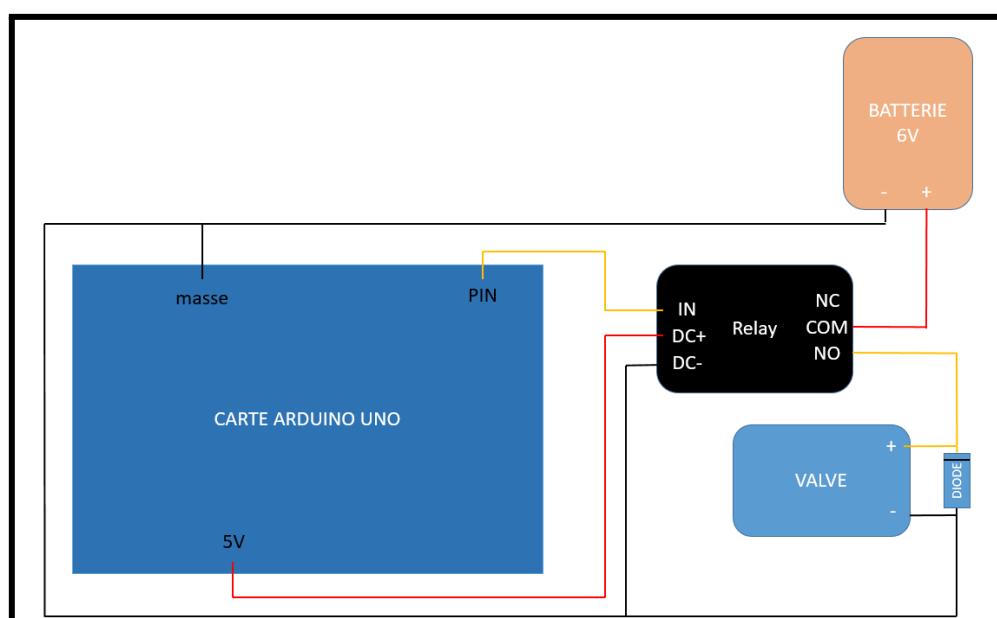
1. Déclencheur de niveau bas lorsque le cavalier est court-circuité à LOW;
2. Lorsque le cavalier est court-circuité à haut niveau, il est déclenché à haut niveau.

L'objectif est de commander la valve à partir du relay, afin de ne pas arroser en continue notre plante.

Avant de commencer, nous allons faire le test du relay afin de voir si celui-ci fonctionne bien. Pour cela, nous utilisons en sortie de celui-ci une LED qui s'allume ou non.



Par la suite, et après avoir vu le bon fonctionnement du relais, nous faisons l'ajout de la valve en question.



- **Programme pour le relais :**

```
const int digPin = 2; // choix du pin pour notre relais

void setup() {
    Serial.begin(9600);
    Serial.println(F("Initialize System"));
    pinMode(digPin, OUTPUT); //Init pwm output
}

void loop() {
    digitalWrite(digPin, HIGH); // ouvre
    delay(1000);

    digitalWrite(digPin, LOW); // ferme
    delay(1000);
}
```

En parallèle de notre valve, nous avons rajouté la LED de notre premier test afin de voir autrement quand notre relais laisse passer ou non notre eau.

## TEST EN RÉSEAU

L'ESP8266 est une toute petite carte qui contient un processeur tournant à 80MHz, une puce WI-FI avec son antenne, un port série et des GPIO (entrées/sorties). Le microcontrôleur est cadencé à 80Mhz par un processeur 32bits RISC avec 96K de RAM et une mémoire flash de 512Ko à 4Mo selon les modèles. Pour les GPIO il diffère d'un modèle à l'autre ,de 1 à plus de 16 I/O. Il dispose d'une connectivité Wifi 802.11 b/g/n supportant le WEP, WPA/2/WPS et réseau ouvert et 16 GPIO dont le support du SPI, I<sup>2</sup>C, UART et un port ADC (pour les I/O analogiques).

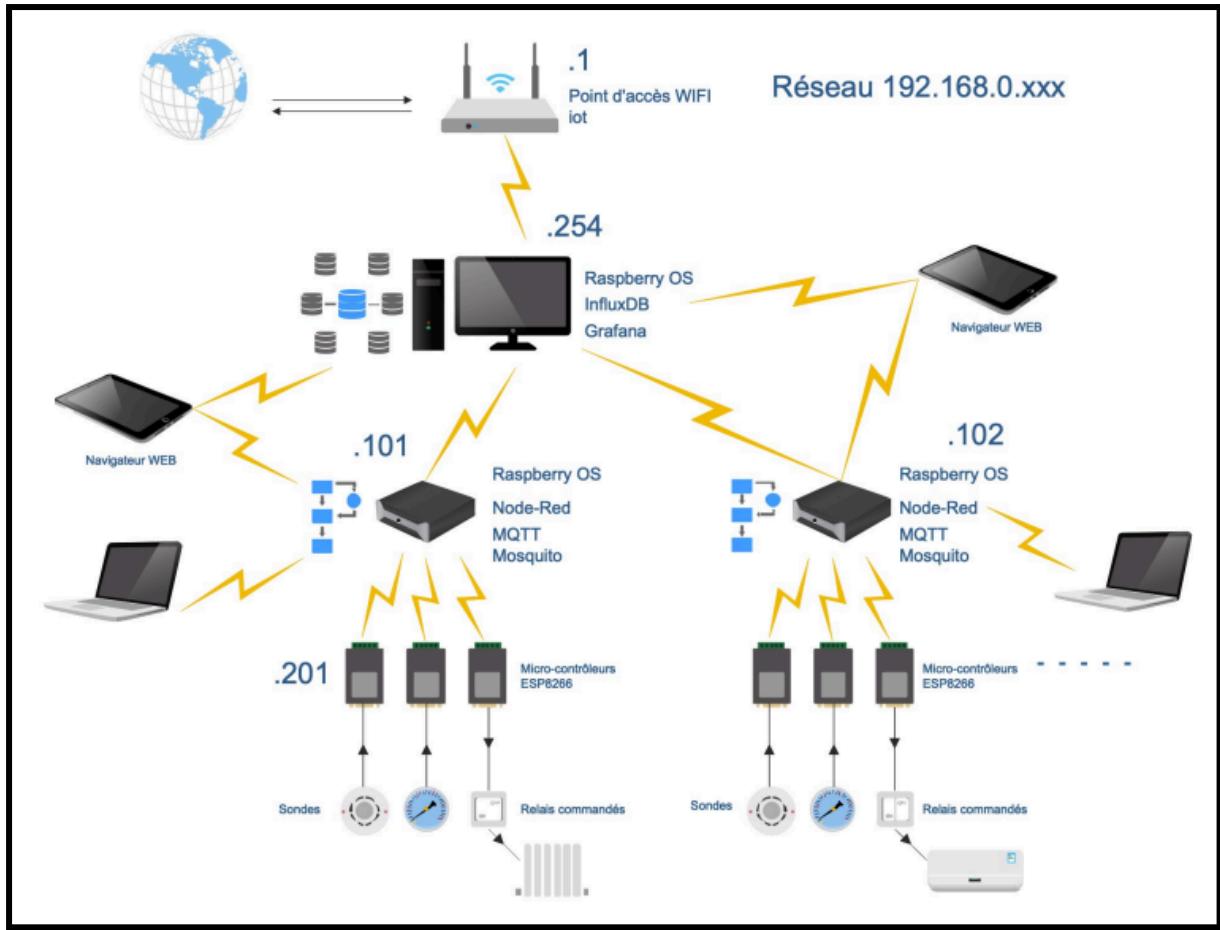
- Utiliser l'**ESP8266MOD** sur une carte arduino pour collecter les données via les capteurs.
- Envoyer les données à un serveur (**InfluxDB**) sur la Raspberry Pi.
- Utiliser Grafana sur la Raspberry Pi pour visualiser les données envoyées par l'ESP.

Dans le dossier Formation IoT (dossier Tuto)

Les parties 1 et 2 sont déjà faites, nous nous faisons la "**Création d'un BROKER**" et "**Création d'une SONDE ESP**".

avec dans "**Création d'un BROKER**" : **Installation MOSQUITO et Installation NODE-RED**

- en utilisant le logiciel **TERMIUS**.
  - Host = user : **pi** et mdp : **raspberrypi**
- 
- Flasher l'ESP8266 -> Utiliser **Tasmota**  
[tasmota.github.io/install/](https://tasmota.github.io/install/)
    - Choisir le port du COM
    - Erase device -> cocher -> next -> install
    - Configure Wi-Fi : IOT\_GEII - butgeii2024
    - visit device
    - Adresse IP active dans l'URL : **192.168.0.110**



### Connexion de notre Raspberry :

(Nous avons la **RASPBERRY NUMERO 6** et la **CARTE SD NUMERO 09**)

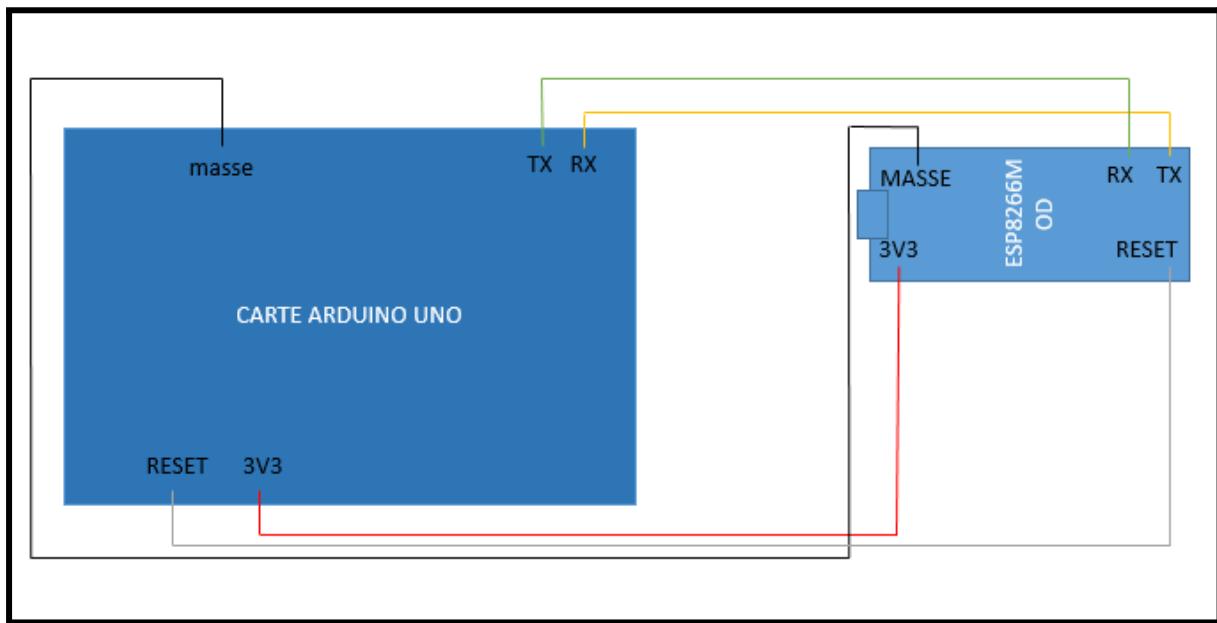
- Accéder au routeur **IOT\_GEII**
- Mdp : **butgeii2024**
- Faire “ipconfig” sur le CMD pour lire l’adresse de notre Raspberry.

Ici nous avons : **192.168.0.101**

Pour pouvoir voir tout le monde qui est connecté à notre réseau :

Logiciel “Advanced IP Scanner” pour pouvoir voir adresse MAC de chacun etc.“  
ou arp -a

### Schéma de câblage pour ESP8266MOD :



### Configuration : ESP3266

URL de gestionnaire de cartes supplémentaires :

[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

Puis dans Outils > gestionnaire de cartes et installer le package esp8266  
et dans type de carte choisir le NodeMCU correspondant.

**LA PARTIE QUI SUIT A ÉTÉ TROUVÉ SUR DES FORUMS, MAIS  
NOUS N'AVONS PAS ENCORE ESSAYÉ LES TESTS.**

### **Code pour ESP8266 avec capteur DHT11 : (exemple sur internet - non testé)**

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <ESP8266HTTPClient.h>

const char* ssid = "Ton_SSID"; // Remplace par le nom de ton réseau Wi-Fi
const char* password = "Ton_Mot_de_Passe"; // Remplace par ton mot de passe Wi-Fi

const char* serverName = "http://<IP_RaspberryPi>/write"; // URL de ton serveur InfluxDB

#define DHTPIN 2      // Le pin auquel le DHT est connecté (D2 sur NodeMCU)
#define DHTTYPE DHT11 // Ou DHT11 selon ton capteur

DHT dht(DHTPIN, DHTTYPE);

void setup() {
    Serial.begin(115200);
    delay(10);

    // Connexion au Wi-Fi
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.print("Connecté à Wi-Fi !");
    Serial.print("Adresse IP : ");
    Serial.println(WiFi.localIP());

    dht.begin(); // Initialiser le capteur DHT
}

void loop() {
    // Lire les données du capteur
    float temperature = dht.readTemperature(); // Lire la température en °C
    float humidity = dht.readHumidity(); // Lire l'humidité en %
}
```

```

// Vérifier si les lectures sont valides
if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Échec de la lecture du capteur DHT");
    return;
}

// Afficher les données dans le moniteur série
Serial.print("Température: ");
Serial.print(temperature);
Serial.print(" °C Humidité: ");
Serial.print(humidity);
Serial.println(" %");

// Créer une requête HTTP POST pour envoyer les données à InfluxDB
HTTPClient http;
http.begin(serverName);

// Ajouter les en-têtes HTTP
http.addHeader("Content-Type", "application/x-www-form-urlencoded");

// Créer les données au format InfluxDB (exemple de ligne de données)
String postData = "temperature=" + String(temperature) + "&humidity=" + String(humidity);

// Envoyer la requête POST
int httpResponseCode = http.POST(postData);

if (httpResponseCode > 0) {
    Serial.print("Réponse HTTP: ");
    Serial.println(httpResponseCode);
} else {
    Serial.print("Erreur HTTP: ");
    Serial.println(httpResponseCode);
}

http.end(); // Terminer la requête HTTP

```

```
// Attendre avant d'envoyer de nouvelles données  
delay(60000); // Attendre 60 secondes avant de lire à nouveau  
}
```

- Configurer InfluxDB sur la Raspberry Pi :

Créer une base données dans InfluxDB :

- Lancer interface InfluxDB en ligne de commande :

**influx**

- Créer une base de données pour les données :

**CREATE DATABASE iot\_data**

- Quitter InfluxDB :

**exit**

- Configurer Grafana sur la Raspberry Pi :

**sudo systemctl start grafana-server**

**sudo systemctl enable grafana-server**

- Accéder à l'interface web de Grafana :

**http://<IP\_de\_ta\_Raspberry\_Pi>:3000**

user et mdp = **admin**

- Ajouter InfluxDb comme source de données dans Grafana :

- Configuration > Sources de données > Ajouter une source de données

- Sélectionner InfluxDB

- Configurer l'URL InfluxDB :

URL : **http://localhost:8086**

Base de données : **iot\_data** (le nom de la base de données que tu as créée).

- Sauvegarder & Tester

- Créer un tableau de bord dans grafana :

- Nouveau tableau de bord

- Sélectionner la source de données InfluxDB

- Requête pour afficher les données prises par l'ESP

**SELECT "temperature" FROM "autogen"."temperature" WHERE \$timeFilter**

- Personnaliser le graphique et sauvegarder le tableau de bord

## **Conclusion de la séance :**

Nous avons utilisé un relais pour contrôler une valve afin d'éviter un arrosage continu. D'abord, nous avons testé le relais avec une LED pour vérifier son bon fonctionnement. Ensuite, nous avons utilisé un microcontrôleur ESP8266, qui se connecte au Wi-Fi pour envoyer les données de capteurs (comme la température et l'humidité) à un serveur installé sur une Raspberry Pi.

Ces données sont stockées dans une base de données InfluxDB et visualisées avec Grafana, ce qui permet de suivre facilement les conditions de la plante. Cette configuration permet de gérer à la fois l'arrosage et la surveillance environnementale à distance.

## Séance 8 - 13.11.2024

Durant cette séance, nous allons faire toutes les configurations + installation afin d'avoir notre Raspberry pi ainsi que notre sonde prêtent, nous permettant de relever les valeurs des capteurs et les envoyer par wifi jusqu'à notre serveur.

En utilisant les Adresses MAC :

Mettre des adresses fixes pour chaque groupe

- **De 60 à 69 = Sonde (ESP8266)**
- **De 70 à 79 = Raspberry**

exemple : Groupe Mochet - Auger -> Groupe 1

- Sonde = 192.168.0.61
- Raspberry = 192.168.0.71

PC IUT -> PGEII-PORT-20 numéro = 20

adresse IP = 192.168.0.106

Pour nous : Groupe 6

Pour avoir une adresse fixe :

- aller sur le routeur **192.168.0.1** (TP-link)
- aller dans DHCP -> address Reservation :
  - mettre l'adresse mac correspondant à notre carte raspberry
  - et changer l'adresse IP en mettant celle de l'intervalle [70-79] que nous avons vu au début.

puis faire la même pour la sonde.

**Mais, ayant un problème avec notre carte, nous avons récupéré celui de la professeur, où NOD-RED, MQTT Explorer sont déjà configurés.**

Broker = cache de la carte SD

avec la carte broker 00 -> Adresse de notre carte = **192.168.0.10**

(il ya tout qui est déjà téléchargé, car celle que nous avions la 09, ne se connecte pas)

- MQTT : **1883**
- Node-Red : **1880**
- InfluxDb : **8086**
- Grafana : **3000**

**exemple : 192.168.0.10:1880 pour nous sur NODE-RED**

**Nous avons changer l'adresse de notre sonde ESP8266 : 192.168.0.70 en fixe**

**Sachant que son adresse de base était : 192.168.0.108**

**sur CHROME car firefox, on ne peut pas connecter notre ESP au PORT :**

<https://tasmota.github.io/install/>

automation using timers or rules, expandability and entirely local control over MQTT, HTTP, Serial or KNX. Written for PlatformIO.

release v14.3.0 | downloads 6.8M | license GPL-3.0 | Discord 1.5k online | Gitpod Ready-to-Code

In light of current events we like to support the people behind *PlatformIO Project*, especially Ivan Kravets, and wish them the strength to help stop the war. See [platformio-is-ukrainian-project-please-help-us-stop-the-war](#) for what you can do.

### Easy install

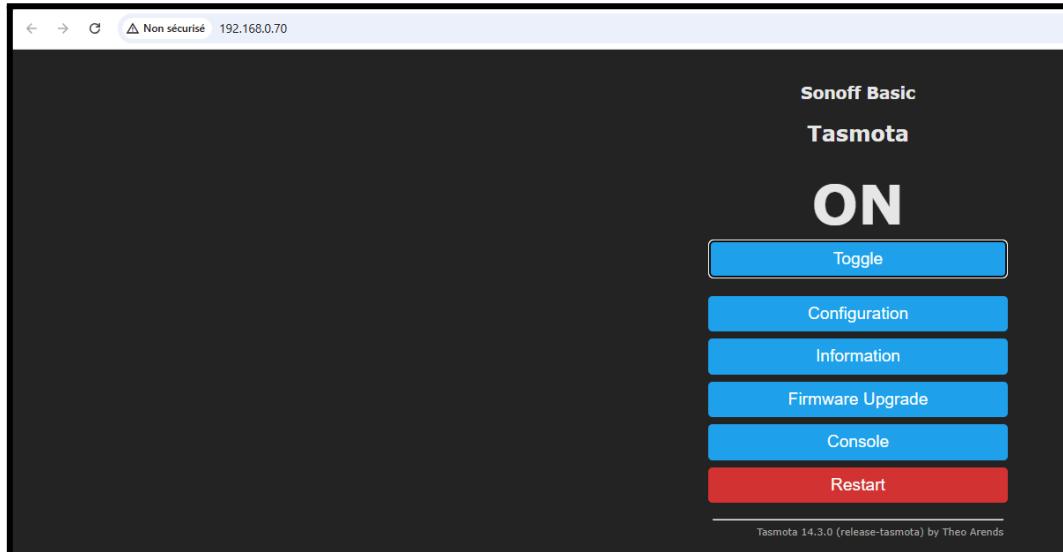
Easy initial installation of Tasmota can be performed using the [Tasmota WebInstaller](#).

If you like **Tasmota**, give it a star, or fork it and contribute!

[Star](#) 22k | [Fork](#) 4.8k | [donate PayPal](#)

See [RELEASENOTES.md](#) for release information.

**Et lorsque nous voyons bien après l'installation, que notre ESP à l'adresse fix .70**



on modifie le nom de notre sonde en question, pour la retrouver par la suite :

**Tasmota**

IOT\_GEII

Scan for all WiFi Networks

**Wifi parameters**

**WiFi Network ()**  
IOT\_GEII

**WiFi Password ■**  
....

**WiFi Network 2 ()**  
Type your Alternative WiFi Network

**WiFi Password ■**  
....

**Hostname (%s-%04d)**  
sonde G6

**Save**

**Configuration**

Tasmota 14.3.0 (release-tasmota) by Theo Arends

**sonde 1**

**Other parameters**

**Template**  
{"NAME":"Generic","GPIO": [1,1,1,1,1,1,1]}

**Activate**

**Web Admin Password ■**  
....

**HTTP API enable**

**MQTT enable**

**Device Name (sonde 1)**  
sonde 1

**Friendly Name 1 (Tasmota)**  
sonde 1

**Emulation**

**None**

**Belkin WeMo single device**

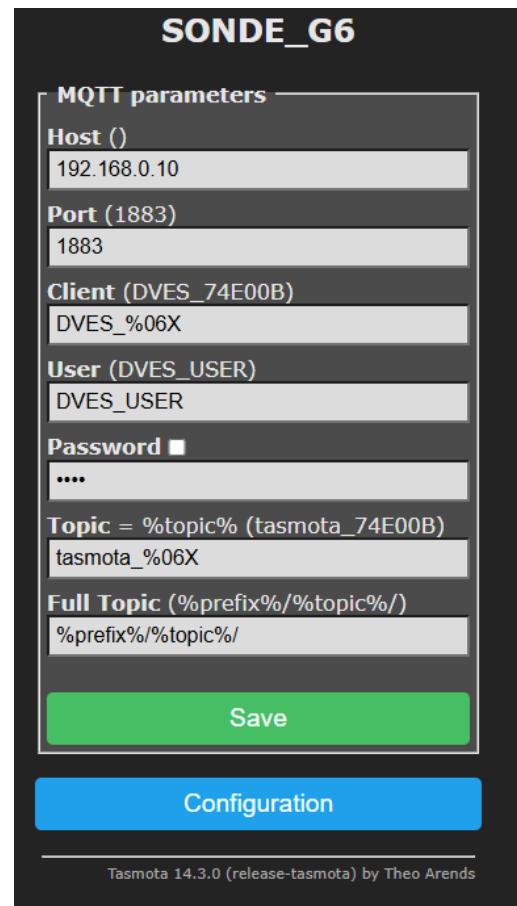
**Hue Bridge multi device**

**Save**

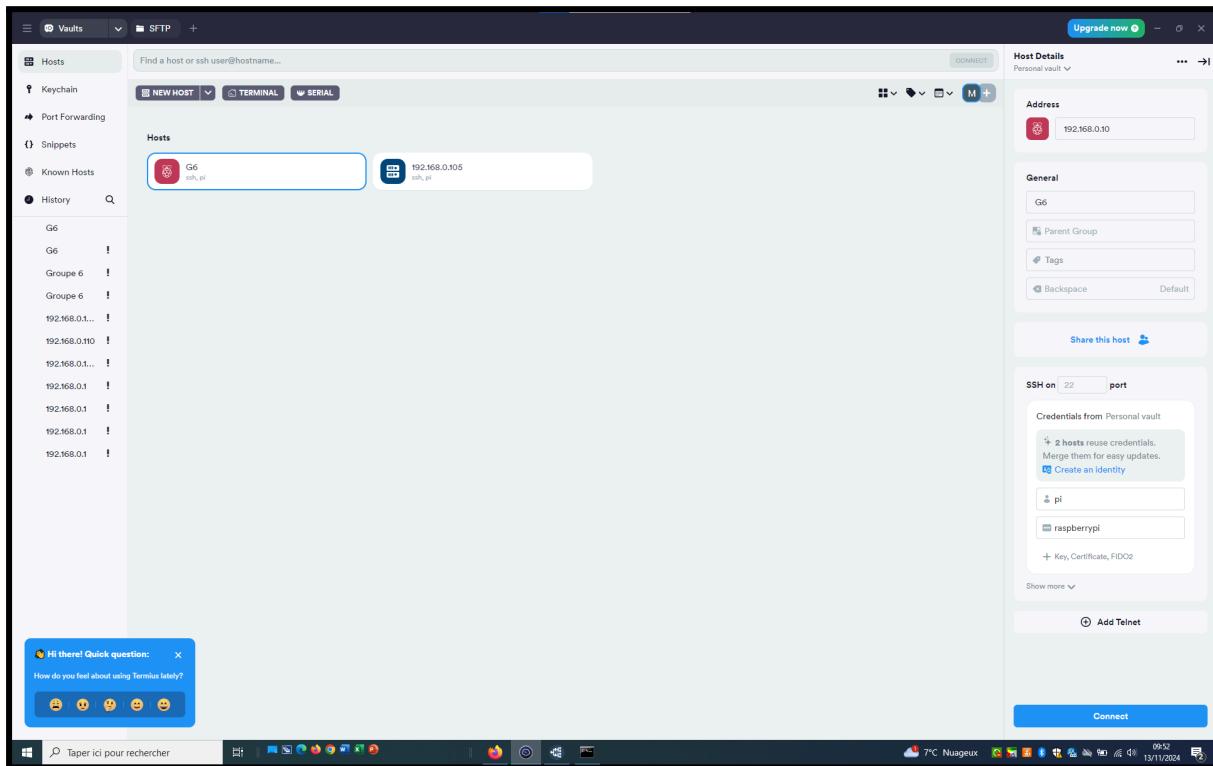
**Configuration**

Tasmota 14.3.0 (release-tasmota) by Theo Arends

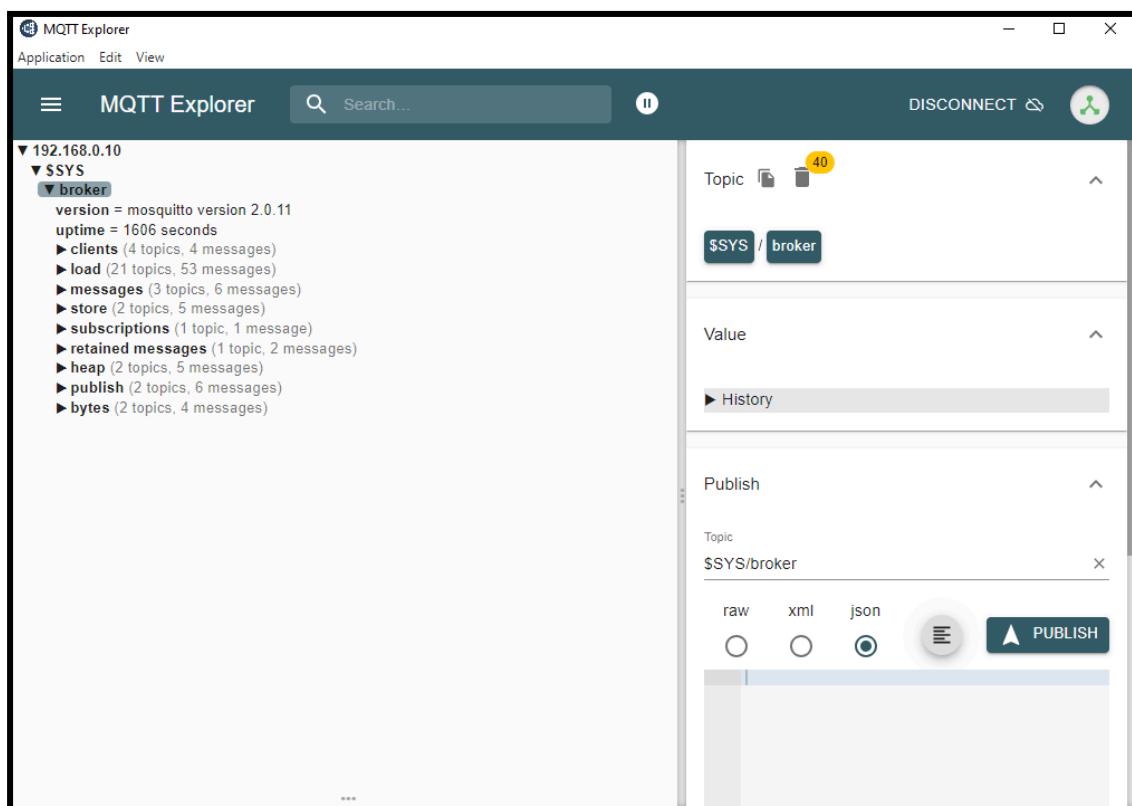
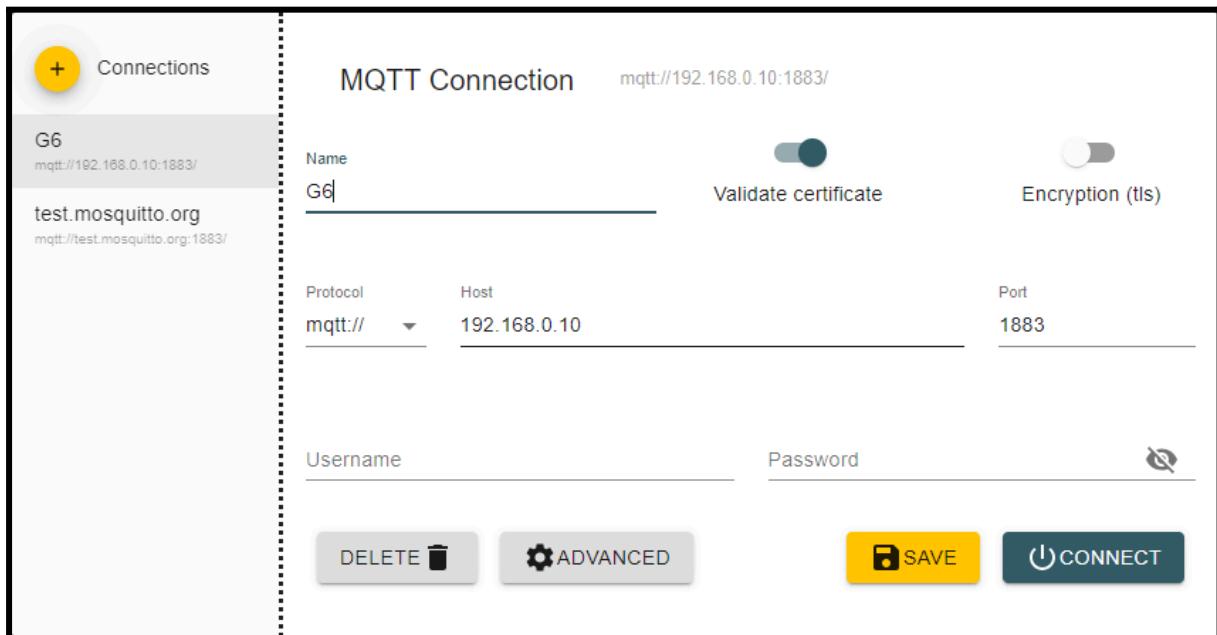
Nous connectons ici, notre ESP à notre Raspberry, permettant ainsi de pouvoir lire les valeurs mesurées, sur MQTT EXPLORER puis permettant l'échange jusqu'à InfluxDB, Grafana.



- sur TERMIUS :



- sur MQTT Explorer :



Nous voyons bien que tout est configuré, et connecté !

Maintenant, nous faisons des recherches pour voir comment faire la liaison, InfluxDB, Grafana etc.

Pour avoir 3.3V pour notre ESP8266, il faut des résistances de :

- 1 Kohms et 2 Kohms

- **Programme ESP8266 :**

```
#include <ESP8266WiFi.h>

const char* ssid = "IOT_GEII"; // Remplace par le nom de ton réseau Wi-Fi
const char* password = "butgeii2024"; // Remplace par ton mot de passe Wi-Fi

void setup() {
    Serial.begin(9600);

    WiFi.begin(ssid, password);

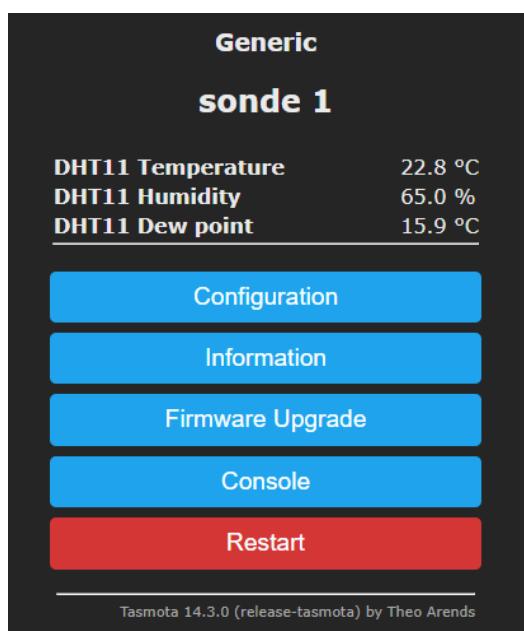
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

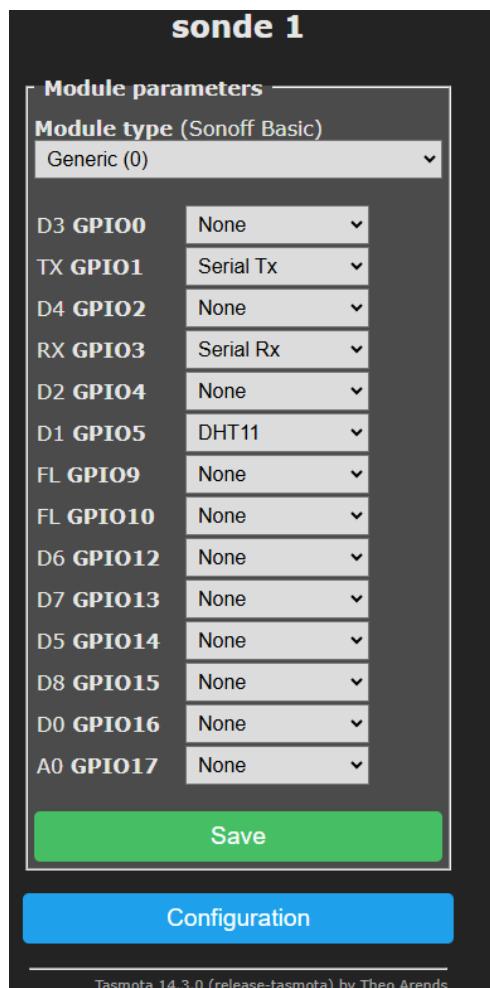
    Serial.println("");
    Serial.print("Connecté à Wi-Fi !");
}

void loop() {
    // Attendre avant d'envoyer de nouvelles données
    delay(2000); // Attendre 60 secondes avant de lire à nouveau
}
```

Après avoir rencontré des problèmes via la bibliothèque de notre ESP, nous avons fait le test de notre DHT en le connectant directement sur notre sonde, nous pouvons visualiser les données comme ci-dessous en sélectionnant le PIN où est branché notre capteur, le tout sur TASMOTA. De plus, nous pouvions bien voir en direct les données sous MQTT EXPLORER, nous ferons par la suite, le test sous Grafana avant d'avancer plus loin.

**TEST DIRECT DHT11 SUR ESP8266 SUR TASMOTA :**





Configurations de nos PINS.

Par la suite, nous avons commencé à connecter notre ESP, et DHT à notre Arduino. Nous allons essayer de faire durant la prochaine séance, les étapes suivantes :

- Relève des données sur Arduino via DHT
- Envoie des données stockées, vers ESP via Liaison Série (Tx, Rx)
- Affichage des données sur Tasmota
- Affichage des données sur MQTT EXPLORER
- Intégration dans la base de données InfluxDB
- Affichage sous graphe sur Grafana

Une fois ces étapes réalisées, nous intégrons l'intégralité de nos capteurs que nous utilisons pour notre plantation de fraise, puis faisons les mêmes méthodes afin de tout afficher sous Grafana.

**test à faire :**

```
#include <DFRobot_DHT11.h>

DFRobot_DHT11 DHT;

#define DHT11_PIN 11 // Broche où est connecté le capteur de température

void setup() {
    Serial.begin(115200); // Utilisez un baudrate plus élevé pour une communication série
    plus rapide
    delay(1000); // Un petit délai pour la stabilisation de la connexion série
}

void loop() {
    // Lire la température et l'humidité
    int temp = DHT.readTemperature(); // Lire la température en °C
    int humidity = DHT.readHumidity(); // Lire l'humidité en %

    if (isnan(temp) || isnan(humidity)) {
        Serial.println("Erreur de lecture du capteur DHT11");
    } else {
        // Envoi des données au port série
        Serial.print("Température ambiante : ");
        Serial.print(temp);
        Serial.println(" °C");

        Serial.print("Humidité ambiante : ");
        Serial.print(humidity);
        Serial.println(" %");
    }

    delay(2000); // Delai de 2 secondes avant la prochaine lecture
}
```

## **Conclusion de la séance :**

Lors de cette séance, nous avons configuré notre Raspberry Pi et notre sonde ESP8266 pour récolter des données de capteurs et les envoyer par Wi-Fi à un serveur. Chaque appareil a reçu une adresse IP fixe : les sondes ont des adresses de 60 à 69 et les Raspberry Pi de 70 à 79. Nous avons configuré la sonde ESP8266 avec l'adresse 192.168.0.70.

Nous avons utilisé une carte SD prête à l'emploi avec Node-RED et MQTT déjà configurés. Après avoir connecté l'ESP à notre réseau Wi-Fi, nous avons vérifié que les données étaient envoyées et visibles sur MQTT Explorer, puis intégrées dans InfluxDB et affichées sur Grafana.

Ensuite, nous avons ajusté la tension pour l'ESP8266 avec des résistances et testé le capteur DHT11, qui a correctement transmis les données de température et d'humidité. Les prochaines étapes consisteront à envoyer ces données de l'Arduino à l'ESP et à les afficher sur Grafana.

## Séance 9 - 19.11.2024

Modification Adresse Fixe : Raspberry Broker00 **192.168.0.60**

Donc modifier dans TASMOTA la liaison avec la Raspberry

192.168.0.10 -> **192.168.0.60**

### Bien branché les TX et RX :

- TX Arduino -> RX ESP8266
- RX Arduino -> TX ESP8266

### Activation de la règle pour lire les données en LIAISON SÉRIE dans TASMOTA :

Rule1 ON SerialReceived#Data DO publish stat/esp8266/sensors %value% ENDON

Rule1 1

Pour reconnaître la carte ESP8266 :

[https://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](https://arduino.esp8266.com/stable/package_esp8266com_index.json)

- Envoi des données :

```
#include "DHT.h"  
#include "SoftwareSerial.h" //instead of parenthesis () put angle bracket as YouTube  
description does not allow angle bracket  
#define DHTPIN 2  
  
// Uncomment whatever type you're using!  
##define DHTTYPE DHT11 // DHT 11  
#define DHTTYPE DHT11 // DHT 22 (AM2302), AM2321  
##define DHTTYPE DHT21 // DHT 21 (AM2301)
```

```
SoftwareSerial espSerial(5, 6);  
DHT dht(DHTPIN, DHTTYPE);  
String str;
```

```
void setup(){  
Serial.begin(115200);  
espSerial.begin(115200);  
dht.begin();  
delay(2000);  
}  
void loop()  
{  
float h = dht.readHumidity();  
// Read temperature as Celsius (the default)  
float t = dht.readTemperature();  
Serial.print("H: ");  
Serial.print(h);  
Serial.print("% ");
```

```

Serial.print(" T: ");
Serial.print(t);
Serial.println("C");
str =String("coming from arduino: ")+String("H= ") +String(h)+String("T= ") +String(t);
espSerial.println(str);
delay(1000);
}

```

**Message (Enter to send message to 'Arduino Uno' on 'COM16')**

```

H: 51.00% T: 23.50C
H: 51.00% T: 23.50C
H: 51.00% T: 23.60C
H: 51.00% T: 23.60C
H: 50.00% T: 23.60C

```

- Réception des données :

```

void setup() {
// Open serial communications and wait for port to open:
Serial.begin(115200);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
}

void loop() { // run over and over
if (Serial.available()) {
Serial.write(Serial.read());
}
}

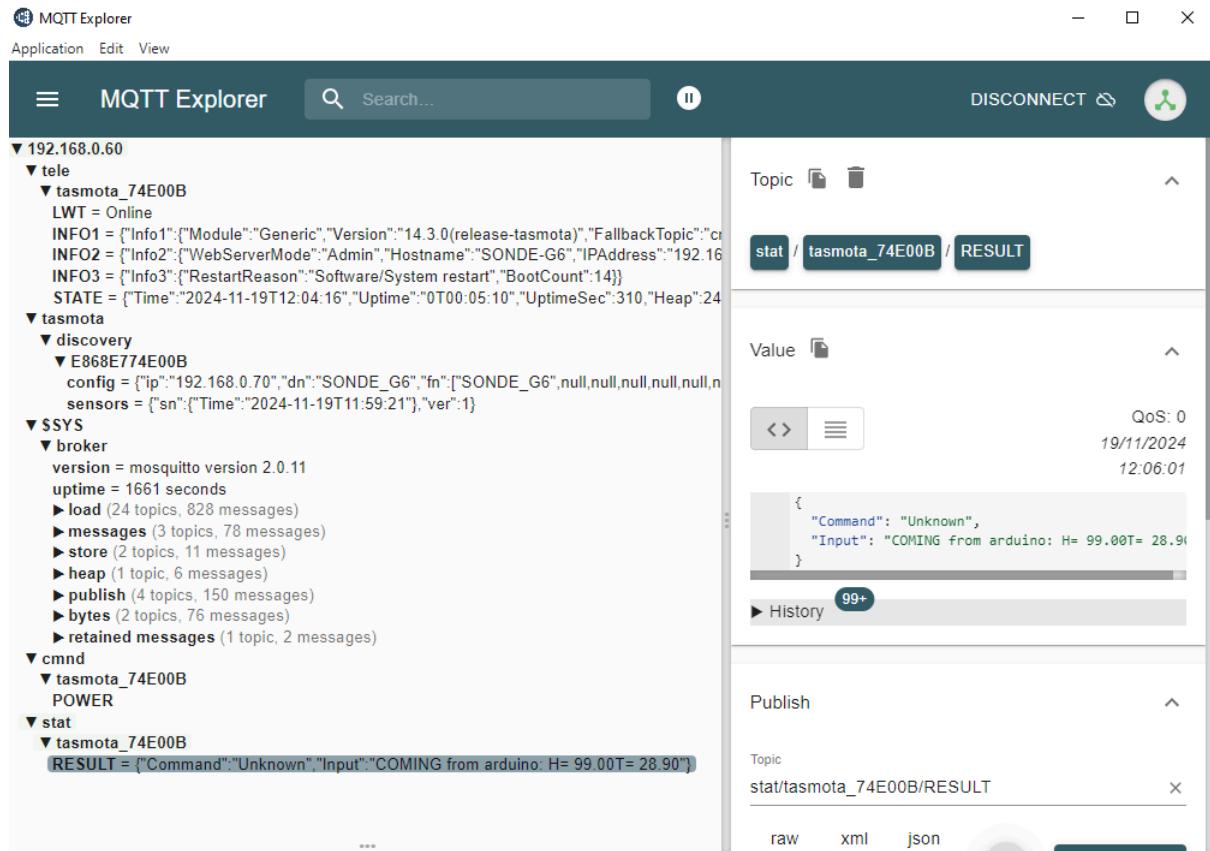
```

**Message (Enter to send message to 'Generic ESP8266 Module' on 'COM17')**

```

12:03:02.510 CMD: coming from arduino: H= 50.00T= 23.50
12:03:02.521 MQT: stat/tasmota_74E00B/RESULT = {"Command": "Unknown", "Input": "COMING from arduino: H= 50.00T= 23.50"}
12:03:03.515 CMD: coming from arduino: H= 50.00T= 23.50
12:03:03.523 MQT: stat/tasmota_74E00B/RESULT = {"Command": "Unknown", "Input": "COMING from arduino: H= 50.00T= 23.50"}
12:03:04.546 CMD: coming from arduino: H= 50.00T= 23.50
12:03:04.554 MQT: stat/tasmota_74E00B/RESULT = {"Command": "Unknown", "Input": "COMING from arduino: H= 50.00T= 23.50"}
12:03:05.550 CMD: coming from arduino: H= 50.00T= 23.50
12:03:05.561 MQT: stat/tasmota_74E00B/RESULT = {"Command": "Unknown", "Input": "COMING from arduino: H= 50.00T= 23.50"}
12:03:06.582 CMD: coming from arduino: H= 50.00T= 23.50
12:03:06.591 MQT: stat/tasmota_74E00B/RESULT = {"Command": "Unknown", "Input": "COMING from arduino: H= 50.00T= 23.50"}
12:03:07.587 CMD: coming from arduino: H= 50.00T= 23.50
12:03:07.595 MQT: stat/tasmota_74E00B/RESULT = {"Command": "Unknown", "Input": "COMING from arduino: H= 50.00T= 23.50"}

```



Nous voyons bien les valeurs de notre DHT11 qui s'affiche sur MQTT

Humidité de : 99%

Température de : 28.90°

On branche les autres capteurs (Luminosité et Humidité du sol).

- Code avec les capteurs pour stockés :

```
#include "DHT.h"
#include "SoftwareSerial.h" //au lieu des parenthèses () utilisez des chevrons comme la description YouTube l'exige
#define LDR A2 // Photoresistor sur la broche A2
#define Soil_Moisture_Sensor A0 // Humidité sol

#define DHTPIN 2
#define DHTTYPE DHT11 // DHT 22 (AM2302), AM2321

SoftwareSerial espSerial(5, 6); // RX et TX
DHT dht(DHTPIN, DHTTYPE);
String str;

void setup() {
    Serial.begin(115200);
    espSerial.begin(115200);
```

```

dht.begin();
pinMode(LDR, INPUT);
pinMode(Soil_Moisture_Sensor, INPUT);
}

void loop() {
    // DHT11
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    Serial.print("H: "); Serial.print(h); Serial.print("% ");
    Serial.print("T: "); Serial.print(t); Serial.println("C");

    // PHOTORESISTANCE
    int value_Luminosite = analogRead(LDR); // Lecture pour la
luminosité
    int pourcentage_Luminosite = (value_Luminosite / 679.0) * 100;
    Serial.print("Eclairage : "); Serial.print(value_Luminosite);
    Serial.print(" - En pourcentage : ");
    Serial.print(pourcentage_Luminosite); Serial.println(" %");

    // HUMIDITE SOL
    int value_Sol = analogRead(Soil_Moisture_Sensor); // Lecture pour
le sol
    int pourcentage_Sol = 100 - ((value_Sol / 876.0) * 100);
    Serial.print("Humidité sol : "); Serial.print(value_Sol);
    Serial.print(" - En pourcentage : "); Serial.print(pourcentage_Sol);
    Serial.println(" %");

    str = String("Hum:") + String(h) +
        String(",Temp:") + String(t) +
        String(",Lum:") + String(pourcentage_Luminosite) +
        String(",Sol:") + String(pourcentage_Sol);

    espSerial.println(str); // Envoyer via ESP8266 ou autre périphérique
série

    delay(5000); // Attendre 1 seconde avant de répéter
}

```

La commande “Rule” est nécessaire pour transmettre les informations à la carte arduino.

## Rule Syntax

Rule definition syntax

```
ON <trigger> DO <command> [ENDON | BREAK]
```

- **ON** - marks the beginning of a rule
- **<trigger>** - what condition needs to occur for the rule to trigger
- **DO** - statement marking end of trigger and beginning of command part
- **<command>** - command that is executed if the **<trigger>** condition is met
- **ENDON** - marks the end of a rule. It can be followed by another rule.
- **BREAK** - marks the end of a rule. **BREAK** will stop the execution of the remaining rules that follow this rule within the rule set. If a rule that ends with **BREAK** is triggered, the following rules in that rule set will not be executed. This allows the rules to somewhat simulate an "IF/ELSE" statement.

Rule sets are defined by using the **Rule<x>** command. After defining a rule set, you have to enable it (turn it on) using **Rule<x> 1**. Similarly you can disable the rule set using **Rule<x> 0**.

## Rule Trigger ~

Rule trigger names are derived from the JSON message displayed in the console. Each JSON level (all values enclosed in `{ ... }`) is separated in the trigger with a `#`. Top level JSON fields are referenced without any `#`, except when the JSON has only one field, where you need `#Data`.

A rule trigger can consist of:

- `[TriggerName]#[ValueName]`
- `[TriggerName]#[ValueName][comparison][value]`
- `[SensorName]#[ValueName]`
- `[SensorName]#[ValueName][comparison][value]`
- `Tele-[SensorName]#[ValueName]`
- `[TriggerName1]#[TriggerName2]#[ValueName]`
- `[TriggerName1]#?#[ValueName]`
- `[ValueName]`
- `[ValueName#Data]`

Use `?` as a wildcard for a single trigger level. Rule will trigger on `[TriggerName]#?#[Value]` where `?` is any value.

### Example

Rule with a trigger of `ZBReceived#?#Power=0` will trigger on `{"ZBReceived":{"0x4773":{"Power":0}}}` and on `{"ZBReceived":{"aqara_switch":{"Power":0}}}` both.

## Rule Command ~

A rule command can be any command listed in the [Commands list](#). The command's `<parameter>` can be replaced with `%value%` which will use the value of the trigger, strings are folded to uppercase.

```
ON Switch1#State DO Power %value% ENDON
```

To accomplish a rule with one trigger but several commands, you need to use `Backlog`:

```
ON <trigger> DO Backlog <command1>; <command2>; <command3> ENDON
```

### Appending new rule onto an existing rule set

Use the `+` character to append a new rule to the rule set. For example:

Existing Rule1: `ON Rules#Timer=1 DO Mem2 %time% ENDON`

Rule to append: `ON Button1#state DO POWER TOGGLE ENDON`

Command: `Rule1 + ON button1#state DO POWER TOGGLE ENDON`

Resulting in

```
Rule1 ON Rules#Timer=1 DO Mem2 %time% ENDON ON Button1#state DO POWER TOGGLE ENDON
```

You can repeat the same trigger in rules.

```
Rule
  ON Power2#state=1 DO Power1 1 ENDON
  ON Power2#state=1 DO RuleTimer1 100 ENDON
```

Power1#State when a power output is changed

use `Power1#state=0` and `Power1#state=1` for comparison, not `=off` or `=on`

Power2 for Relay2, etc.

## **Informations de login :**

### **Network :**

SSID : IOT\_GEII

pswd : butgeii2024

### **Serveur :**

IP 192.168.0.254

#### **-Grafana : 3000**

usr : admin

pswd : butgeii2024

#### **-Influx DB : 8086**

usr : admin

pswd : butgeii2024

### **Broker : 192.168.0.6X**

usr : pi

pswd : raspberrypi

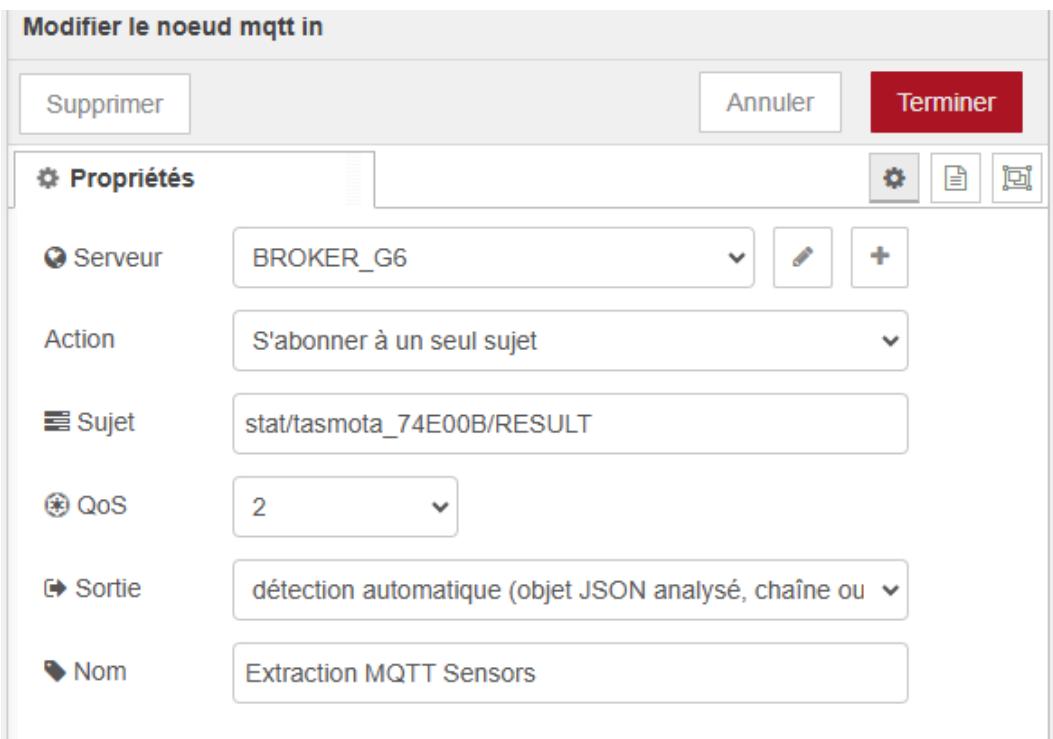
-MQTT (mosquitto)

-Node-red : 1880

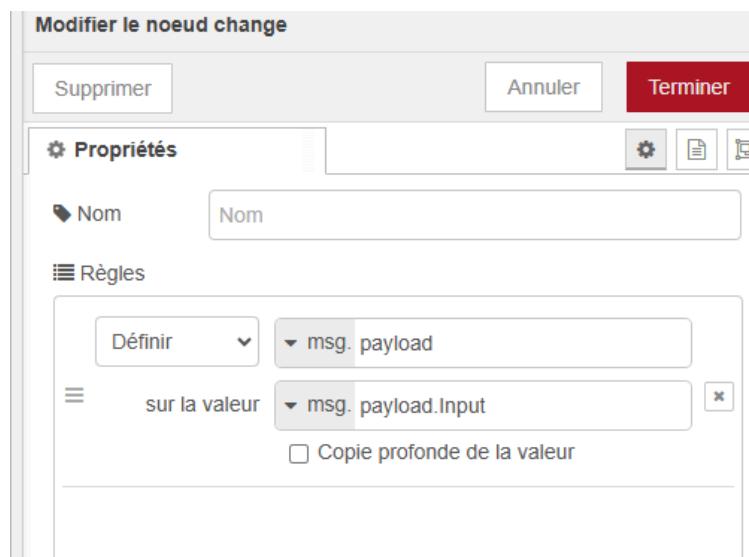
### **Node-RED :**

192.168.0.60

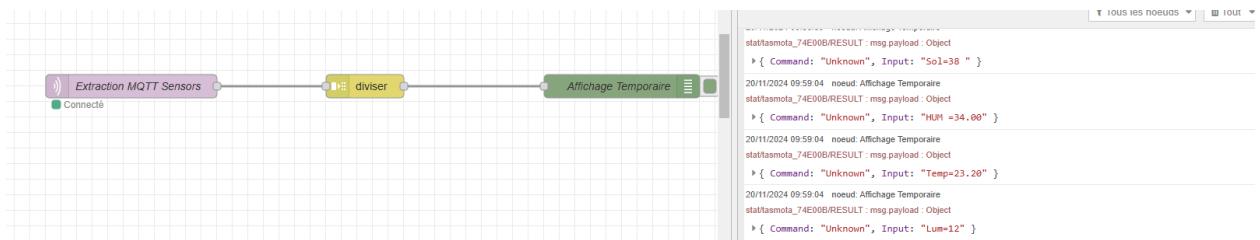
On rentre le nom de notre serveur et le sujet pour pouvoir relier tasmota avec MQTT pour pouvoir l'afficher sur NODE-RED.



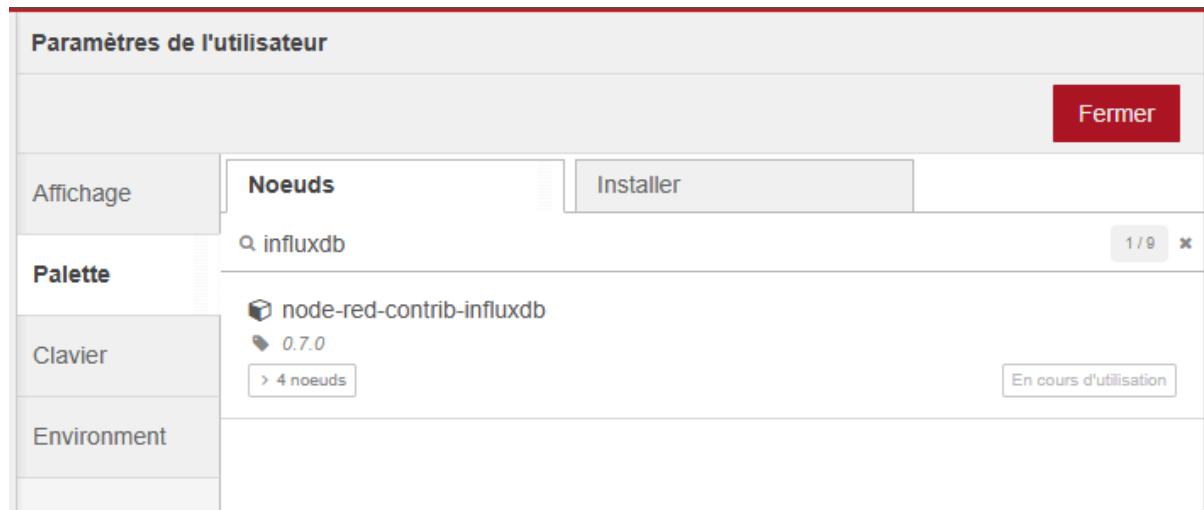
Dans le noeud de change on met payload et dans la case "sur la valeur" on met payload.input et on change la catégorie en "msg.".

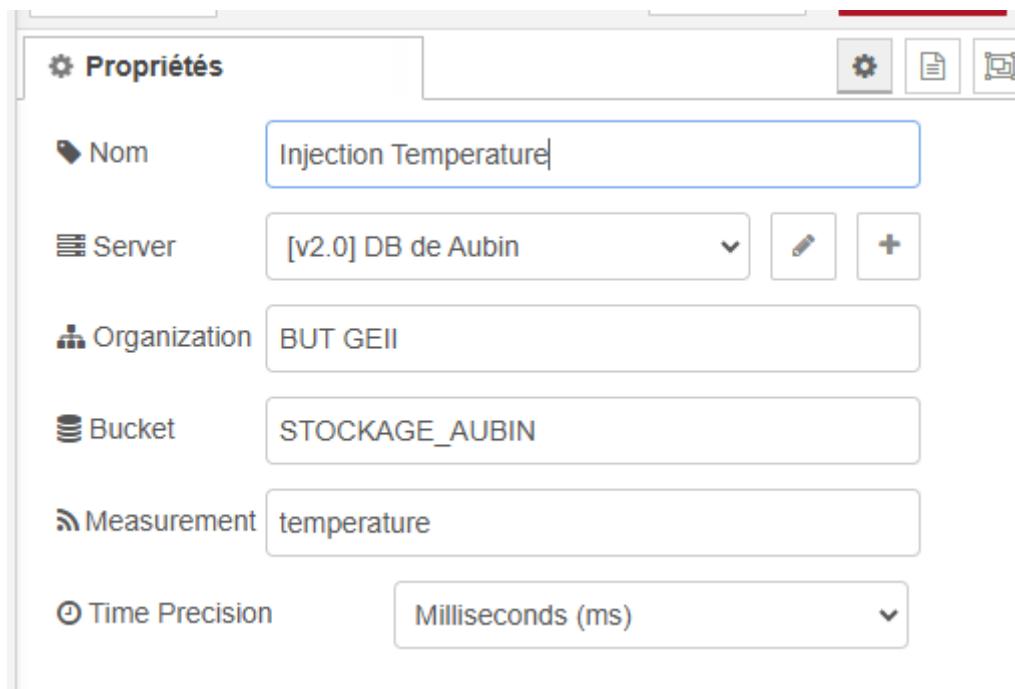


Notre programme fonctionne ainsi , en rose on récupère la data de MQTT en jaune on divise les différentes data dans leurs nom de variables associer(TEMP, HUM, SOL, LUM) puis en vert on affiche sur node-RED.



On a besoin d'installer la palette influxDB pour avoir les fonctions associées:





On récupère une chaîne de caractère de valeurs séparées de “ - ”.

La chaîne est divisée en un tableau où chaque élément contient une clé-valeur séparée par “ : ”.

On ajoute chaque clé-valeur dans un objet data, les espaces sont supprimés avec (“trim()”) et la valeur est convertie en entier avec “parseFloat(value.trim())”.

On donne l’objet “data” au payload du message avant de le retourner.

#### Le code :

```
let payload = msg.payload; // Récupère le payload du message

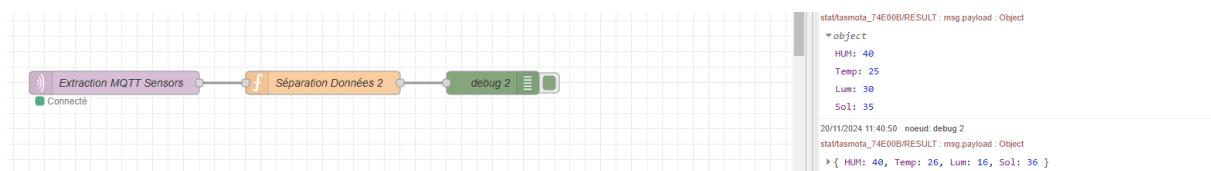
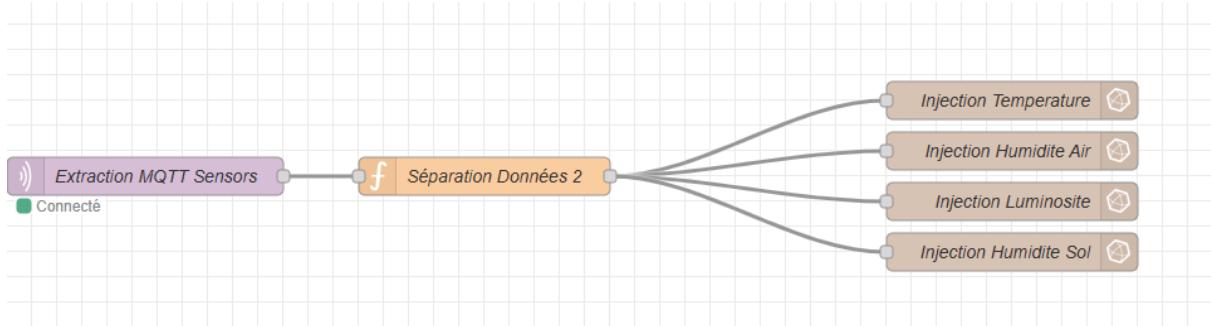
let metrics = payload.Input.split(',');
let data = {};

// Parcours chaque métrique
metrics.forEach(metric => {
    // Sépare chaque élément par le séparateur ':'
    let [key, value] = metric.split(':');
    // Ajoute la clé et la valeur dans l'objet data après avoir
    // supprimé les espaces
    data[key.trim()] = parseFloat(value.trim());
});

// Affecte l'objet 'data' à msg.payload
msg.payload = data;

return msg;
```

On a construit des "boîtes" pour chaque mesure de nos capteurs soit ranger dans cette boite pour que l'on affiche facilement avec node red et cela nous permet d'afficher les valeurs une par une



## InfluxDB :

Dans influxDB on génère un token que l'on lie avec graphana.

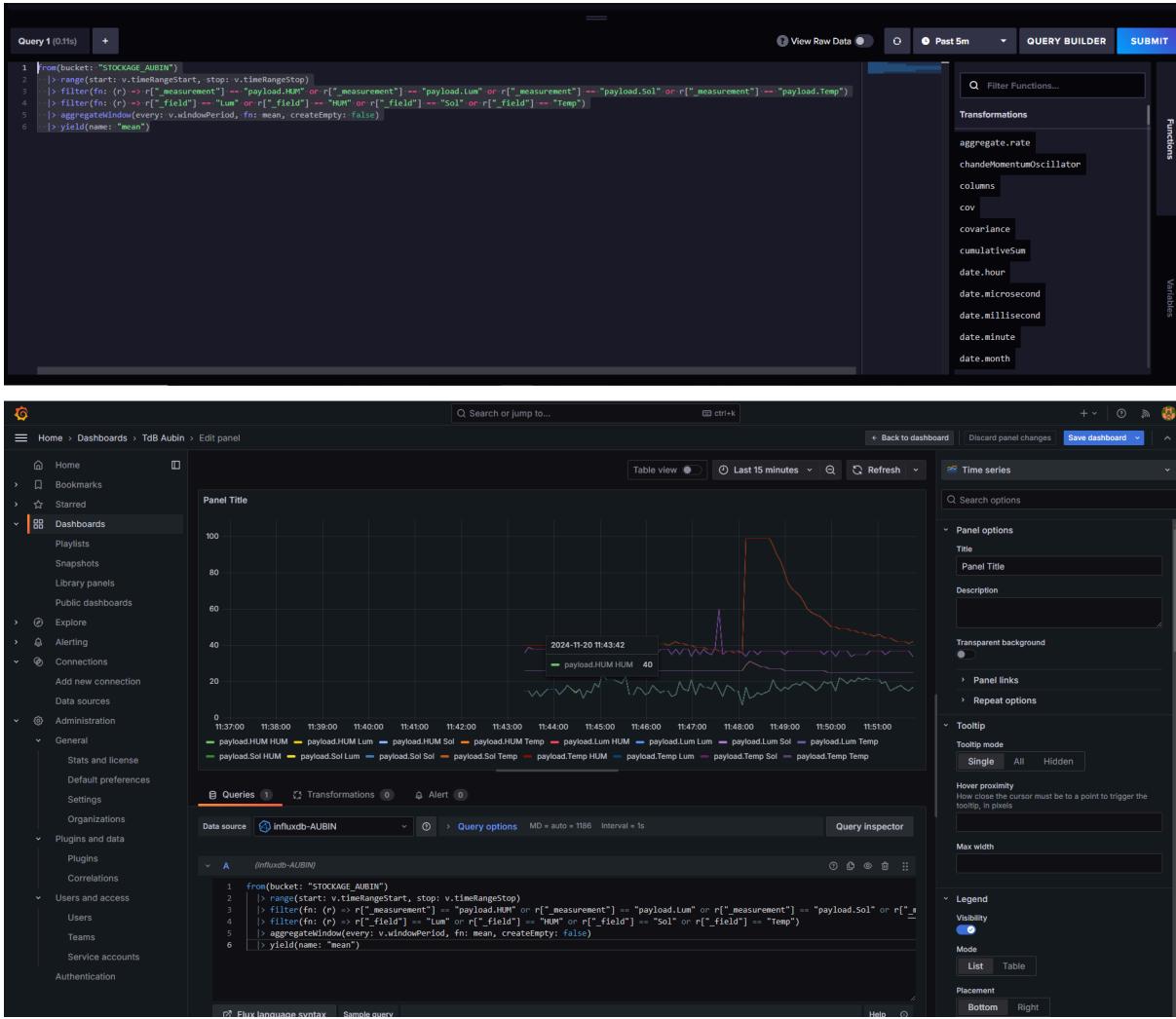
The screenshot shows the 'Load Data' section of the InfluxDB interface, specifically the 'API TOKENS' tab. It lists four tokens:

- admin-grafana**: Created at 2024-11-12 01:07:12, Owner: admin, Last Modified: 8 days ago.
- admin's Token**: Created at 2024-11-12 00:52:30, Owner: admin, Last Modified: 8 days ago.
- Aubin TOURAIS**: Created at 2024-11-20 10:09:56, Owner: admin, Last Modified: 0 seconds ago.
- Lukas LEONIDAS**: Created at 2024-11-19 09:33:54, Owner: admin, Last Modified: 1 day ago.

On the right side, there are buttons for '+ GENERATE API TOKEN' (highlighted in blue), 'All Access API Token', and 'Custom API Token'.



- Placement du script InfluxDB dans Grafana :



# Grafana

The screenshot shows the Grafana interface. On the left, there's a sidebar with navigation links like Home, Bookmarks, Starred, Dashboards, Explore, Alerting, Connections (selected), Data sources, Administration, General, Stats and license, Default preferences, Settings, Organizations, Plugins and data, and Users and access. The main area has a "Welcome to Grafana" header and a "Basic" section with a "DATA SOURCE AND DASHBOARDS" tutorial about Grafana fundamentals. It also features three "COMPLETE" cards: "Add your first data source", "Create your first dashboard", and "Learn how in the docs". Below this is a "Dashboards" section with "Starred dashboards" and "Recently viewed dashboards". To the right, there's a "Latest from the blog" section with two posts: one about YACE becoming a Prometheus-community project and another about Grafana's BIG TENT release.

Ici on copie le lien que l'on a généré avec influxDB dans “Token” afin de le configurer.

This screenshot shows the "InfluxDB Details" configuration page. It contains five input fields: "Organization" (BUT GEII), "Token" (configured), "Default Bucket" (iot), "Min time interval" (10s), and "Max series" (1000). A "Reset" button is located next to the Token field. At the bottom, a green success message states "✓ datasource is working. 6 buckets found". Below it, a blue link says "Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view."



### Exemple de script : *Humidité*

```

from(bucket: "STOCKAGE_AUBIN")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "payload.HUM")
|> filter(fn: (r) => r["_field"] == "HUM")
|> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
|> yield(name: "mean")

```

## Conclusion de la séance :

Lors de cette séance, nous avons configuré notre Raspberry Pi et notre broker MQTT avec l'adresse IP fixe 192.168.0.60. Nous avons également établi une connexion entre notre Arduino et l'ESP8266 en reliant les pins TX et RX. Une règle a été activée dans Tasmota pour envoyer les données de l'ESP8266 à MQTT lorsque des données série sont reçues. Ensuite, nous avons utilisé Arduino pour lire les valeurs des capteurs : température et humidité (DHT11), luminosité (photoresistor) et humidité du sol (capteur d'humidité du sol). Ces données ont été envoyées via la liaison série à l'ESP8266, qui les a transmises à Node-RED via MQTT. Dans Node-RED, les données ont été traitées et affichées en temps réel.

Les informations étaient ensuite stockées dans InfluxDB, permettant de les visualiser sous forme de graphiques sur Grafana. Nous avons vérifié que chaque étape fonctionnait correctement, depuis la collecte des données jusqu'à leur affichage et stockage dans les différentes plateformes.

## Conclusion

Ce projet a été une excellente occasion de mettre en pratique nos compétences en codage tout en nous familiarisant avec le câblage des capteurs et de la carte Arduino. Bien que Tasmota et MQTT aient été les parties les plus complexes et aient posé quelques difficultés, nous avons réussi à surmonter ces obstacles avec l'aide précieuse de notre professeure.

Grâce à ce système, nous pouvons suivre en temps réel la température, l'humidité du sol et de l'air, ainsi que la luminosité autour de notre fraisier, et afficher ces données sous forme de graphiques. Nous avons également la possibilité de visualiser un historique des valeurs de chaque capteur, ainsi que la date et l'heure de leur activation, ou encore d'identifier des erreurs, comme un sol sec avec une pompe activée mais un réservoir d'eau vide.

Malgré cela, bien que nous ayons rempli le cahier des charges du projet, il y a plusieurs améliorations que nous aurions pu apporter. Par exemple, nous aurions pu intégrer un véritable plan de fraise pour rendre le système encore plus réaliste. Nous aurions aussi pu ajouter des boutons sur l'interface utilisateur pour contrôler les actionneurs, comme les pompes, ou encore inclure des LEDs et même imaginer un environnement de serre pour enrichir davantage le projet.

En somme, même si des améliorations étaient possibles, ce projet a été très intéressant et a enrichi nos connaissances. Il nous a permis d'appliquer ce que nous avons appris et de mieux comprendre l'intégration de différents systèmes dans un projet IoT.