**Installation :**

Version choisie : Raspberry Pi Os (32-bit). Avant installation, cocher SSH dans les paramètres

Username : DI22

Password : DI22

1. 4 dissipateurs de chaleur collés.
2. Arduino IDE : <https://www.arduino.cc/en/donate/> -> IDE pour l’ESP. Spécialisé en C et C++.

Installer la bibliothèque : esp8266 by ESP8266 Community

*~~Type de carte : ESP8266 Module. Fichiers – Exemple – Basics – Blink.~~*

*~~Créer la constante : cons tint LED = 2, et remplacer tous les LED\_BUILTIN par LED.~~*

*~~Brancher le convertisseur sur l’ordinateur et brancher le câble d’alimentation sur le module d’alimentation.~~*

*~~Installer Driver CH341~~*

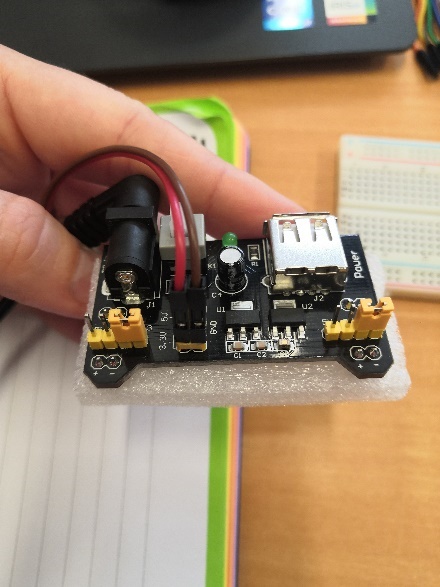
* Ne pas utiliser Arduino local mais Arduino web. Penser à modifier LED\_BUILTIN et la remplacer par 1.

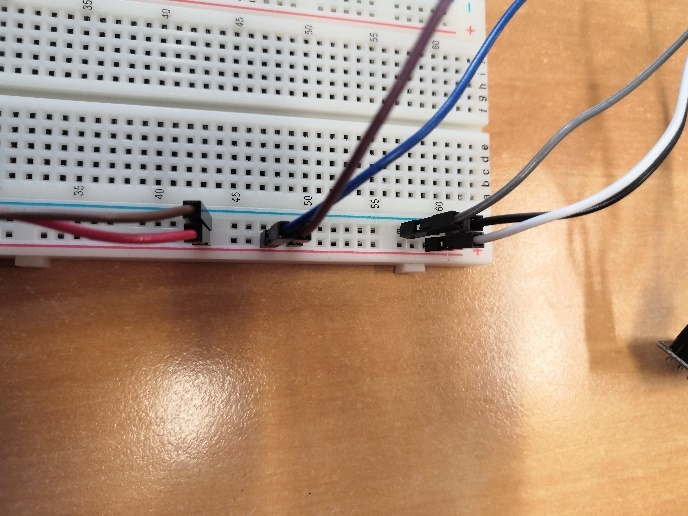
1. Montage de transmission - réception **ESP / Convertisseur USB vers TTL /Module d’alimentation –** POUR BRANCHEMENT SUR ORDINATEUR

Une image contenant texte, tableau blanc

Description générée automatiquement

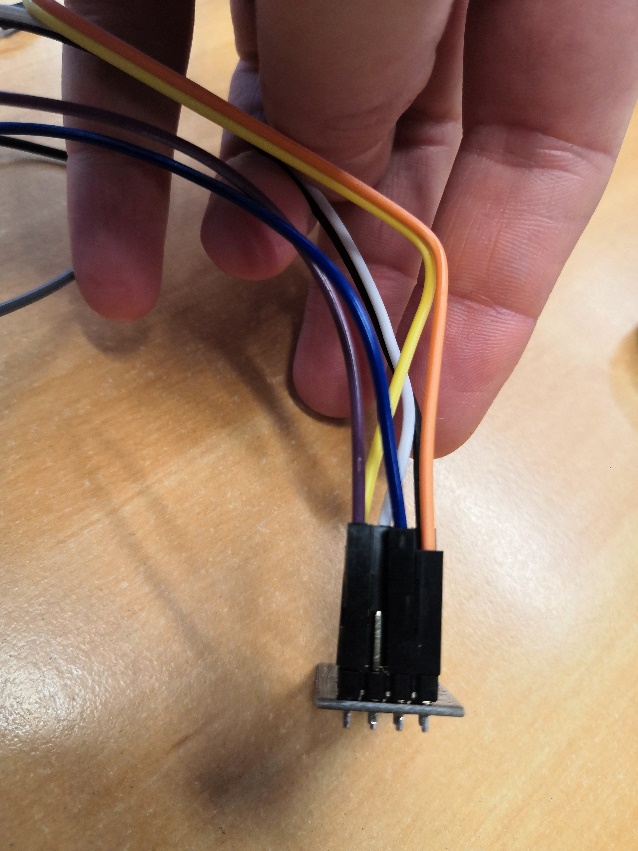
* USB :
  + GNB vers - Breadboard
  + TXD vers RXD de l’ESP (en haut à gauche -orientation ESP : zig zag à gauche)
  + RXD vers TXD de l’ESP (en bas à droite)
* ESP :
  + GRD vers – Breadboard
  + GPIO0 vers – Breadboard
  + CH\_PD vers + Breadboard
  + VCC vers + Breadboard
* Module d’alimentation
  + GRD vers – Breadboard
  + 3.3V vers + Breadboard





Une image contenant personne, ciseaux

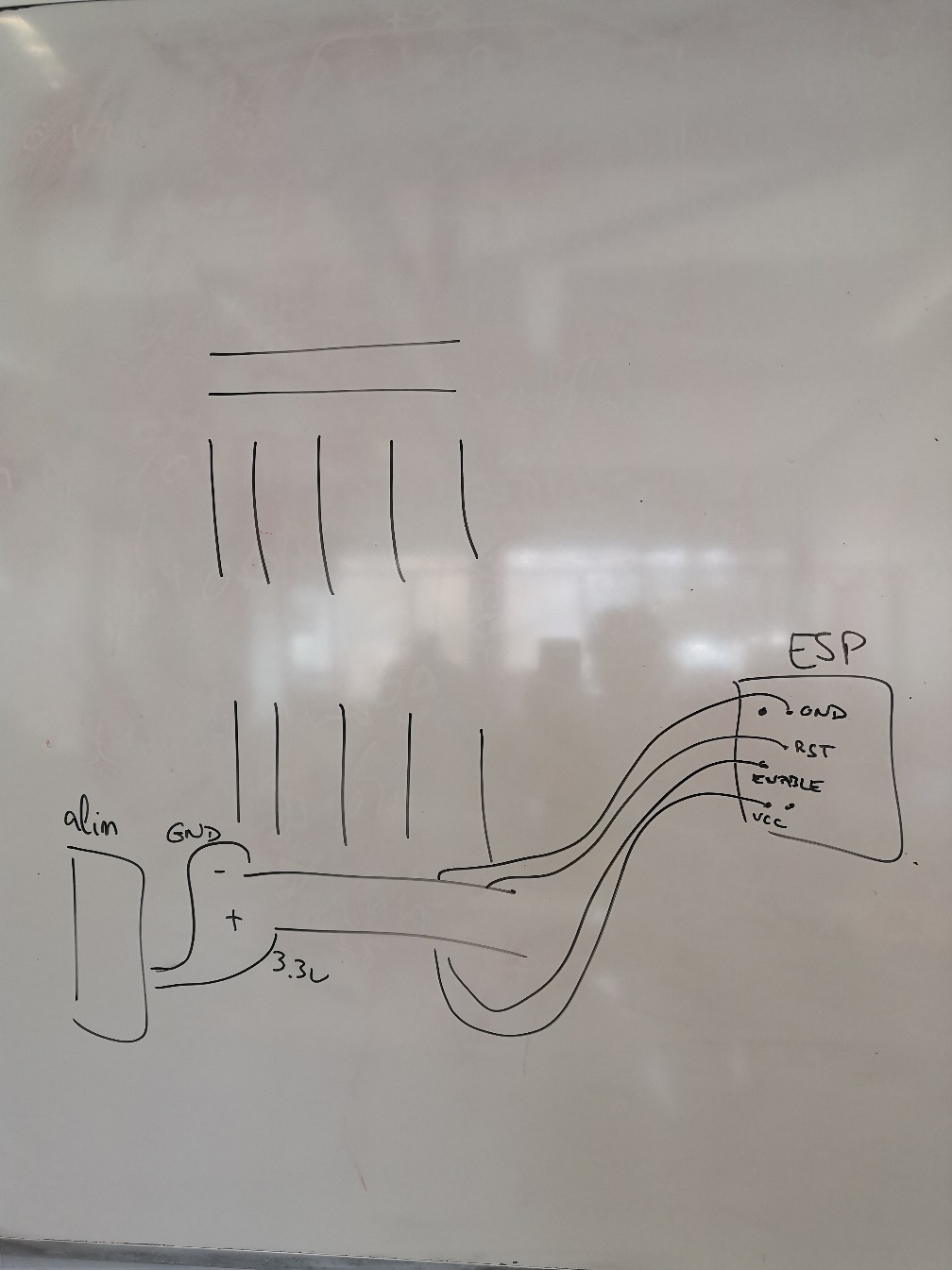
Description générée automatiquement



Une image contenant texte, intérieur, clavier

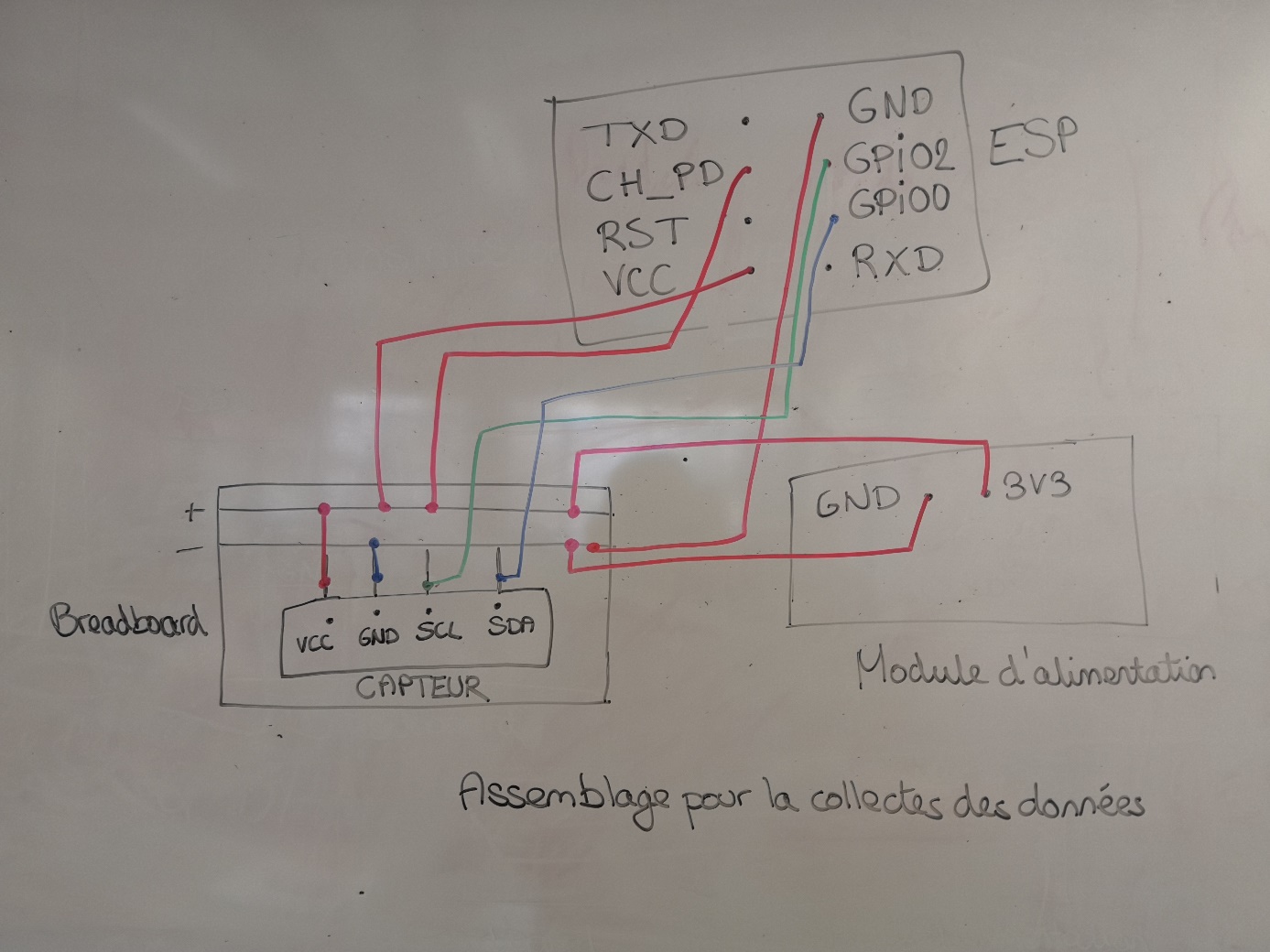
Description générée automatiquement

Schéma de montage ESP / Breadboard / Alimentation par Rémi :

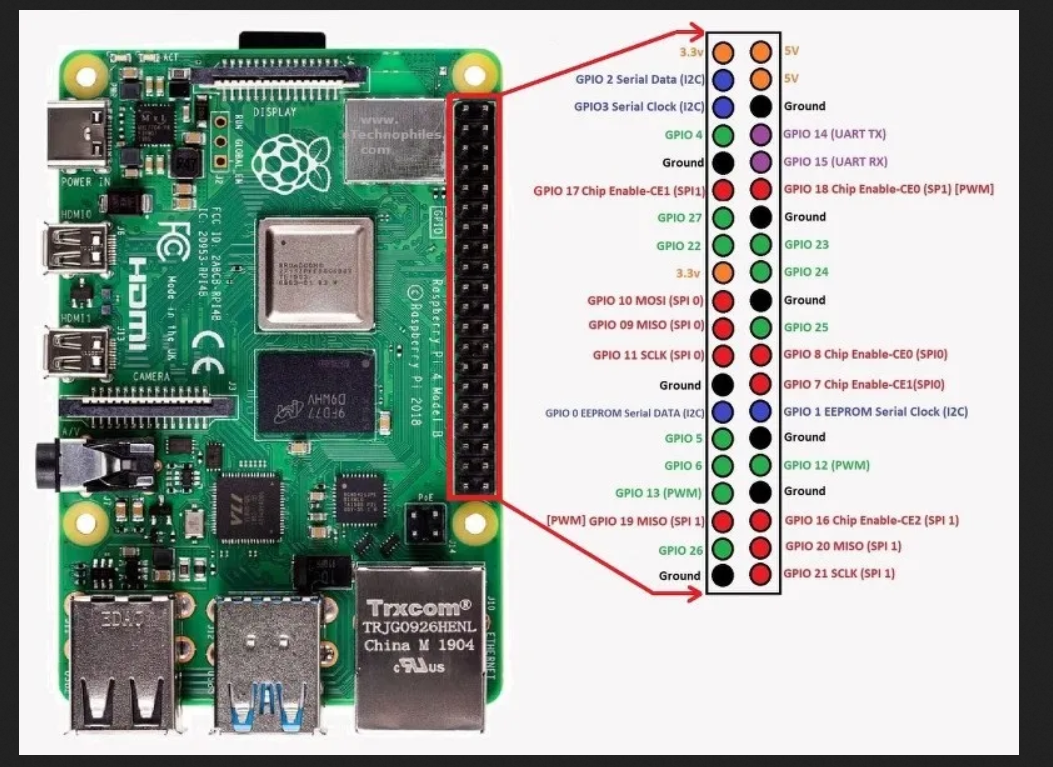


Test via Arduino web, envoi d’un script sur l’ESP -> les diodes s’allument.

1. Assemblage LED / Raspberry :



* LCD
* GND vers pin 5 (GND) du Raspberry
* VCC vers pin 1 (3V3) du Raspberry
* SDA vers pin 2 (SDA) du Raspberry
* SCL vers pin 3 (SCL) du Raspberry



Une image contenant texte, équipement électronique, capture d’écran

Description générée automatiquement

* ~~Capteur :~~ 
  + - ~~VCC vers + Breadboard~~
    - ~~GND vers GND Raspberry (Pin 5 rang gauche)~~
    - ~~SCL vers SCL Raspberry (Pin 3)~~
    - ~~SDA vers SDA (Pin 2)~~
* ~~Raspberry : Pin 1 vers + Breadboard (même rang + que le VCC du capteur)~~

1. Assemblage ESP / Capteur / Breadboard / Module d’alimentation

Une image contenant texte, tableau blanc

Description générée automatiquement

* Module d’alimentation
* GND vers – Breadboard
* 3.3 V vers + Breadboard
* ESP
* GND vers – BB
* GPIO2 vers ligne Capteur de la Breadboard SCL
* GPIO0 vers ligne Capteur de la Breadboard SCA
* CHPD vers + Breadboard
* VCC vers + Breadboard
* Breadboard
* VCC de la ligne capteur vers + Breadboard
* GND de la ligne capteur vers – Breadboard



Une image contenant adaptateur

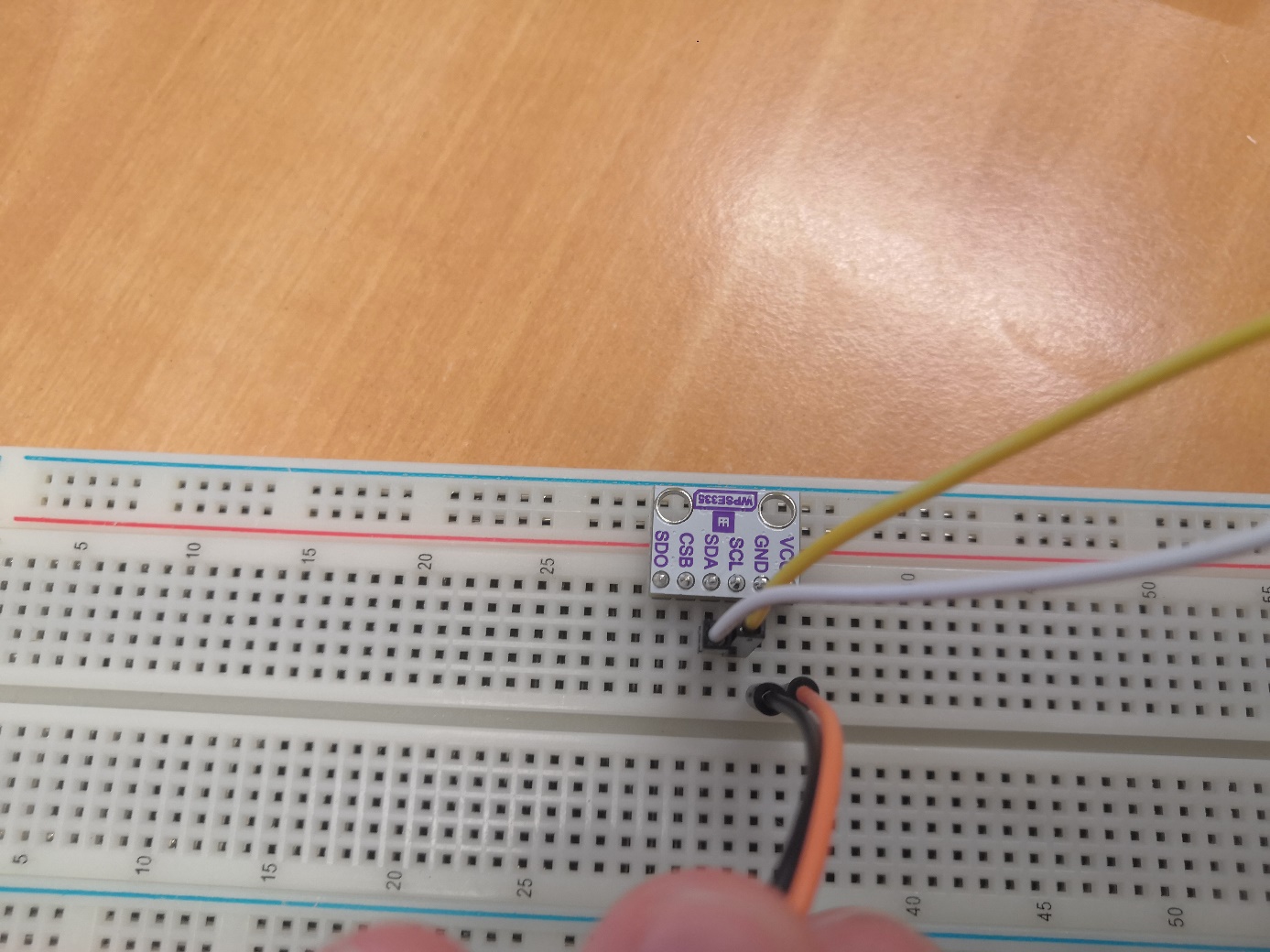
Description générée automatiquement

Une image contenant personne

Description générée automatiquement

Une image contenant équipement électronique

Description générée automatiquement



**Informations sur les composants :**

Sur le Rasberry : 2 broches réservées à l’I2C

Module d’alimentation (avec diode verte) : Il pourrait recevoir une batterie mais ça ne sera pas le cas dans notre projet, il peut aussi recevoir une USB. On le branche sur le breadboard. Il fournit une alimentation à tous les composants.

Broche 5V : Si le Raspberry fonctionne sous une tension de 5V, il est équipé d’un régulateur 5V. La sortie de ce régulateur est connectée sur cette broche qui peut donc être employée pour alimenter des circuits externes en 5V. Si on désire construire sa propre alimentation 5V, on peut également fournir cette tension au Raspberry via cette broche.

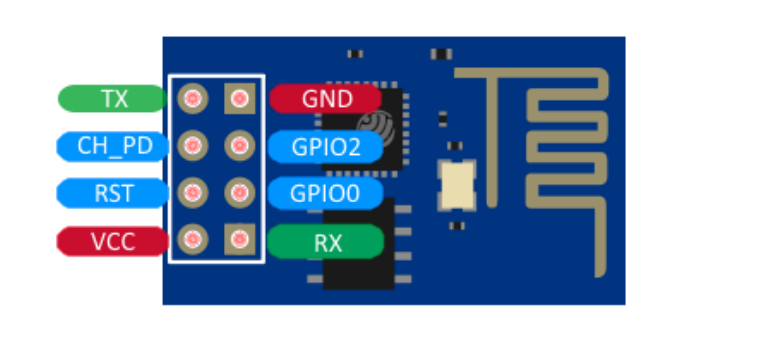
Broche 3V3 : Si on équipe le Raspberry d’un régulateur 3,3V, la sortie sera connectée à cette broche. On peut s’en servir pour alimenter un circuit externe en 3,3V.

Broche TXD : Signaux de transmission

Broche RXD : Signaux de réception

Broche GND (Ground) : La masse ou 0V, fil de terre

ESP : (la petite puce avec des broches d’un côté) : c’est une puce capable de discuter en WIFI et reçoit I2C. On peut y mettre un programme. On doit utiliser un câblage particulier pour transférer le programme et un autre câblage pour utiliser le programme. RST : aucun courant, permet de reset, on met la puce à 0v. Enable : branché sur le +, doit recevoir en permanence 3V pour être activé donc aussi branchée sur le 3v.



GPIO2 = SCL

GPIO0 = SDA

Le capteur d’humidité (petit capteur avec deux trous) (voir HTU21) : il sera branché à l’ESP. L’ESP remonte les données, il les envoie au Raspberry. <https://whadda.com/product/bme280-temperature-humidity-and-pressure-sensor-wpse335/> | Manuel d’utilisation plus détaillé : <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf> | Manuel pour utilisation avec API : <https://github.com/BoschSensortec/BME280_driver>

Définitions pins du capteur :

VCC: connected with 3.3V

SCL: used to generate the clock signal

SDA: used in sending and receiving data

Le convertisseur USB vers TTL : C’est un module qui sert à transférer le programme sur l’ESP. On branche une partie sur le PC et l’autre sur l’ESP. Ce module ne sera pas dans la version finale du Raspberry, c’est simplement un outil de développeur. Voir le schéma cablage\_esp\_flash.

LCD (SSD1306) : **à préciser avec Olivier** pour savoir s’il faut brancher sur l’ESP ou sur le Raspberry.

Breadboard : Simulateur de branchements : <https://www.tinkercad.com/>

Pour le wifi esp 8266.

**Informations utiles :**

Il existe plusieurs façons de configurer et utiliser un module ESP8266, mais voici les étapes générales pour l'utiliser avec un microcontrôleur Arduino : Branchez le module ESP8266 à un port série de l'Arduino en utilisant des câbles de liaison (TX de l'Arduino à RX de l'ESP8266, RX de l'Arduino à TX de l'ESP8266, et alimentation GND à GND). Téléchargez et installez les bibliothèques ESP8266 pour Arduino à partir de la bibliothèque de gestionnaire de bibliothèques d'Arduino. Utilisez la bibliothèque ESP8266 pour configurer le module en mode AT (AT Command) Utilisez les commandes AT pour configurer les paramètres réseau tels que le SSID et le mot de passe du réseau Wi-Fi. Utilisez les fonctions de la bibliothèque ESP8266 pour envoyer et recevoir des données via le réseau Wi-Fi.

La bibliothèque ESP8266 pour Arduino est appelée "ESP8266WiFi". Elle fournit une interface pour utiliser les fonctionnalités de connexion Wi-Fi du module ESP8266 avec un microcontrôleur Arduino. Cette bibliothèque inclut des fonctions pour se connecter à un réseau Wi-Fi, créer un point d'accès Wi-Fi, envoyer et recevoir des données via des connexions TCP et UDP, et bien plus encore. Il vous permet de contrôler votre ESP8266 en utilisant des commandes AT pour vous connecter à un réseau wifi, gérer le serveur web embarqué, et bien plus encore.

**Liens utiles :**

<https://raspberry-pi.fr/raspberry-pi-sans-ecran-sans-clavier/>

[https://fr.pinout.xyz/pinout/i2c#](https://fr.pinout.xyz/pinout/i2c)

<https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/>

joy.it.net -> Documentations

Docs info sur convertisseur USB en TTL : <https://cdn.planete-domotique.com/notices/CUBI_TTL.pdf>

Idées trame projet : <https://lastminuteengineers.com/bme280-esp8266-weather-station/>

**Conseils Rémi :**

* A la place de OpenWeatherMap : weatherapi.com
* Un exemple de serveur web pour esp8266 est dispo dans les exemples de l'ide; cherchez "helloserver"
* CRON : planificateur de tâches de Linux qui va collecter les informations toutes les tant minutes.

**Etapes du projet :**

1. Installation avec tests (voir partie Installation) (en partie validé par Rémi)
2. Installation de l’OS sur la carte micro SD
3. Organisation générale du projet et partage des tâches + émergence questions
4. Schéma de câblage -> Envoi à Olivier avec nos questions
5. Ecriture d’un script pour la connexion Wifi sur l’ESP et le Rasberry
6. 1 script pour récupérer les données du capteur et les envoyer en base, 1 script pour récupérer les données en base et les envoyer à l’IHM.
   * + Configurer le raspberry et l’OS
     + Utilisation de CRON pour planifier les appels à l’API
     + Réflexions sur l’interface pour faire émerger les grandes lignes du travail
     + Grandes lignes de la BDD (MySQL)
     + Relevé toutes les heures tous les jours // Tous les 15 jours, garder seulement 4 à 5 relevés (ex : 7h / 12h / 17h / 00h ?)