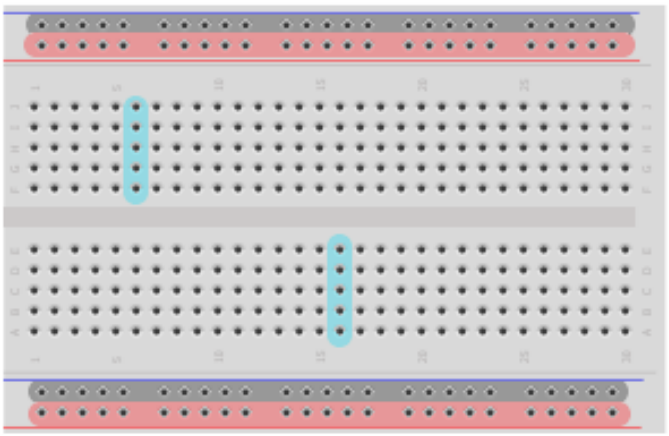
TP : ESP32

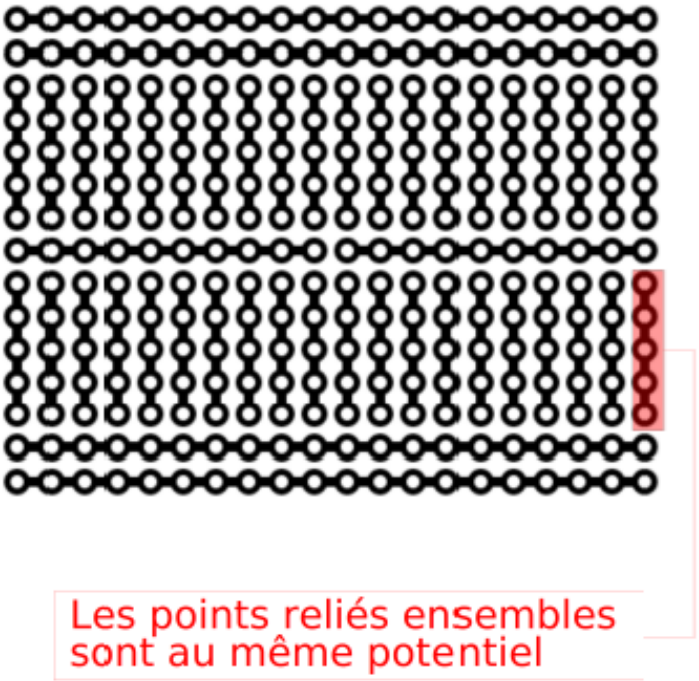
Objectifs du Projet :

Ce TP porte sur la mise en pratique d'envoi de données par l'intermédiaire de la carte ESP32.

Prise en main du matériel :

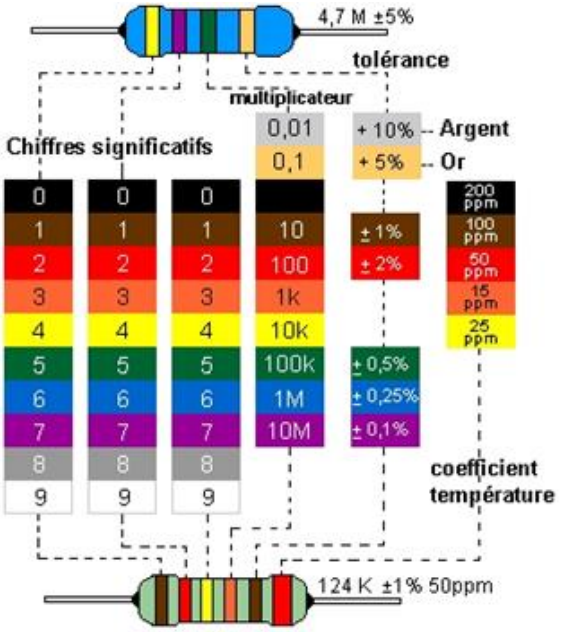
1. Breadboard
   Cet équipement jouera le rôle de plaquette électronique par laquelle les circuits électriques se réalisent. Ses différents trous permettent aux composants de se brancher ou se relier entre eux.





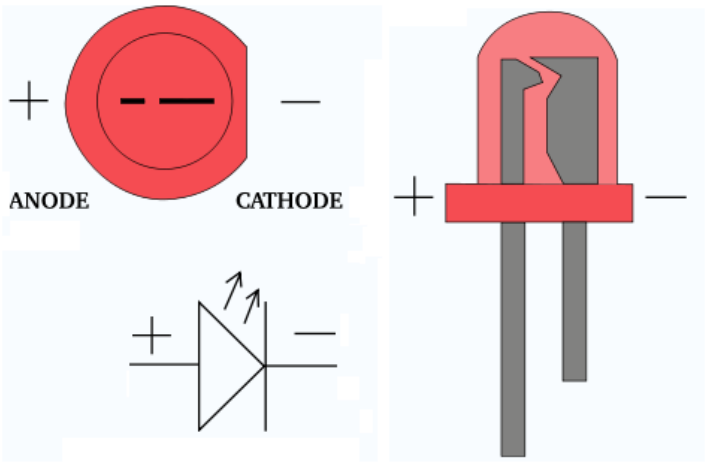
2. Composants passifs
   a. Résistors

Les résistances sont généralement utilisées pour réduire la tension sur un équipement particulier. Sa valeur est obtenue à partir de la lecture sur le code de couleur définie ci-dessous.



b. Capa
c. Self
d. Leds

Les LEDs (Light Emetting Diode) sont une catégorie de diodes émettant de la lumière par l'intermédiaire du courant reçu.



Elles ont selon la couleur des tensions de seuil différentes comme le reporte le tableau ci-dessous.

Couleurs	Tension seuil
Rouge	1,6V a 2V
Jaune	~ 1,8V
Verte	~ 1,8V
Bleue	2,7 a 3,2

3. ESP32

a. Description

L'ESP32 est un microcontrôleur et un module Wi-Fi/Bluetooth/BLE développé par la société chinoise Expressif Systems. Il cible une large variété d'applications du moins gourmande en énergie au très grand nombre d'instructions comme l'encodage de la voix, le streaming ou même décodage MP3. Il regroupe principalement les caractéristiques suivantes :

- Microcontrôleur : Xtensa Dual Core 32-bits LX6 processor
- Fréquence d'horloge : 80MHz a 140MHz réglable
- Connectivité sans fil : Wi-Fi 802.11 b/g/n
- Mémoire Flash : jusqu'à 32Mo
- Interfaces : UART, I2C, SPI, GPIO, ADC, DAC

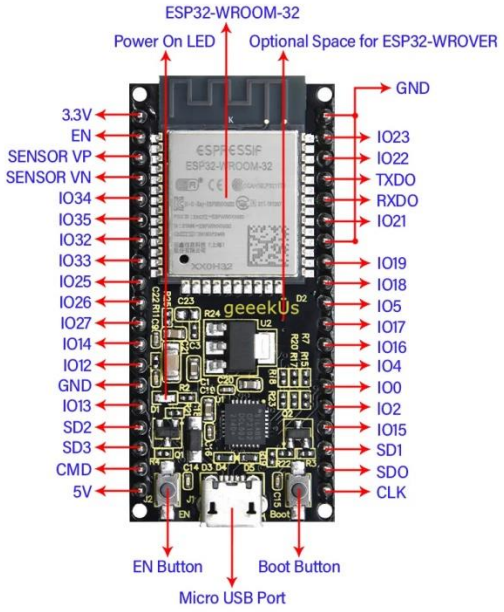


Fig 1 : ESP32-WROOM-32

- EN Button : c'est le bouton reset du module ESP .
- Boot Button : Ce bouton charge le code depuis Arduino dans le module ESP.
- GPIO : C'est sur ces pins que presque tout le développement se passe. Contrairement à ESP8266, tous les GPIO de l'ESP32 peuvent utiliser en écriture/lecture analogique, digitale, PWM, I2C, SPI, ADC, DAC et bien plus.

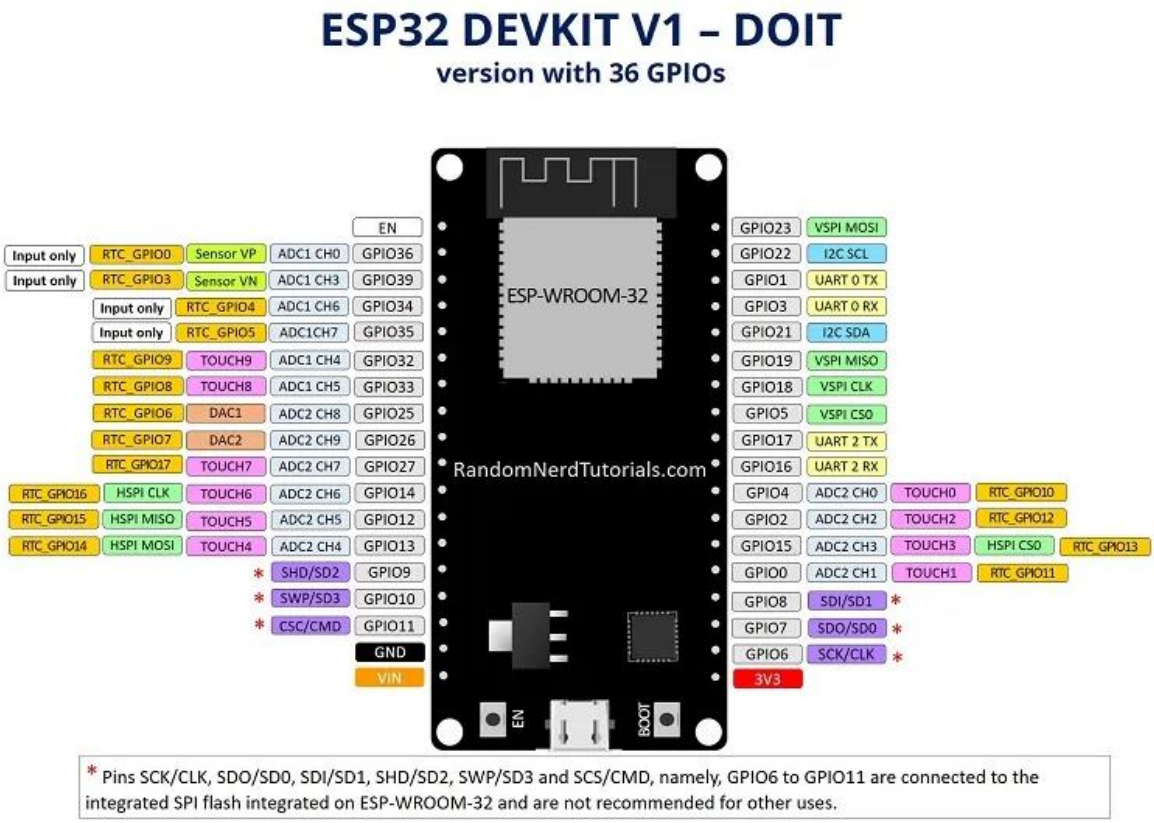


Fig 2 : ESP32-WROOM-32 Pinout

b. Utilisation

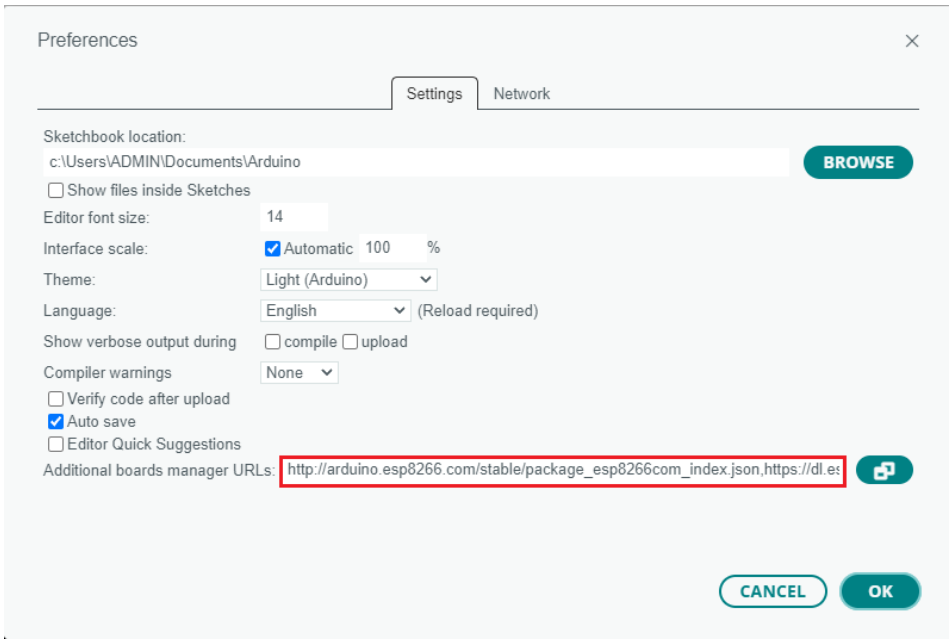
- La programmation de l'ESP32 peut se faire avec l'IDE Arduino en installant les packages « ESP32 Boards » sur la carte. Ça se fait de la même façon aussi avec l'IDE VS Code (Visual Studio Code)
- La programmation de l'ESP32 peut aussi se faire avec l'IDE MycroPython en installant le firmware MicroPython sur la carte.
- La programmation de l'ESP32 peut également se faire avec l'IDE Lua en installant la firmware NodeMCU sur la carte.

Partie Pratique

a. IDE Arduino

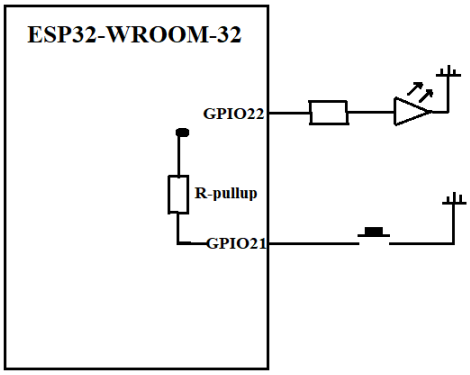
C'est en effet l'interface qui nous permettra de mettre des programmes à la carte. Le langage de programmation est typique au microcontrôleur et est semblable au C/C++ voir htpp://www.arduino.cc/référence.

- Il est important tout d'abord de procéder à l'ajout de la carte ESP32 sur l'IDE Arduino. Le process est ainsi :
  Files → Preferences → Additionnal boards manager URLs
  Et ensuite d'ajouter autant de cartes possibles par leur URL en les séparant sur la barre Additionnal boards manager URLs par une virgule.
  Exemple de URLs pour ajout carte :
  https://dl.espressif.com/dl/package_esp32_index.json,
  http://arduino.esp8266.com/stable/package_esp8266com_index.json



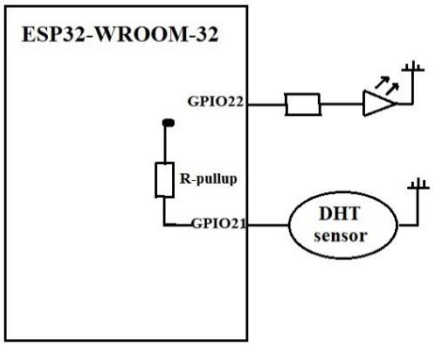
- Une fois la carte ajoutée, on doit procéder à l'installation des packages ESP32 sur l'IDE Arduino. Le process est comme suit :
  Tools → Boards Manager
  Et rechercher et installer les packages ESP32 requises (by Expressif).
- Dernière étape, c'est de bien choisir la carte et le port d'accès.

i. Schéma équivalent 1



- Réaliser le montage ci-dessous.
- Effectuer la programmation nécessaire pour allumer la led que lorsque le bouton poussoir est pressé.
- Remplacer le bouton poussoir par un capteur ultrason et programmer la led à s'allumer que lorsqu'un obstacle à une distance inferieure à 10cm est rencontré. (**NB** : installer la librairie relative a ultrason sensor dans **Tools → Manage Librairies**).

i. Schéma équivalent 2



- Réaliser le montage.
- Programmer la carte de telle sorte que lorsqu'elle est connectée à votre réseau, la led s'allume. Afficher dans le moniteur série, les caractéristiques de la connexion établie comme l'adresse IP, la passerelle, la puissance etc.
- Programmer la carte de telle sorte qu'elle envoie toutes les 10 secondes la température environnante à la plateforme Adafruit.
- Visualiser la courbe tracée sur le Dashboard.

Bibliothèques, Librairies et Fonctions de Base

- Serial

  Serial est une librairie essentielle du langage Arduino qui permet de visualiser sur le PC des messages reçus depuis la carte Arduino ou de commander la carte Arduino.
  Elle est utilisée pour les communications par le port série entre la carte Arduino et un ordinateur ou d'autres composants. Toutes les cartes Arduino ont au moins un port Série (également désigné sous le nom de UART ou USART). Ce port série communique sur les broches **RX** et **TX** avec l'ordinateur via le port USB.

  - **Serial.begin()**
    Cette fonction fixe le débit de communication en nombre de caractères par seconde.
  - **Serial.available()**
    Cette fonction donne le nombre d'octets (caractères) disponibles pour lecture dans la file d'attente (buffer) du port série.
  - **Serial.print()**
    Cette fonction affiche les données sur le port série sous forme lisible pour les humains (texte ASCII).
  - **Serial.read()**
    Cette fonction lit les données entrantes sur le port série.
  - **Serial.write()**
    Cette fonction écrit des données binaires sur le port série.
  - **Serial.flush()**
    Cette fonction s'assure que les données écrites sur le port série aient été physiquement envoyés avant de rendre la main (la fonction est donc bloquante).

- pinMode

  Cette fonction configure la broche spécifiée pour qu'elle se comporte soit en entrée, soit en sortie.

  - **pinMode(broche, mode)**
    - broche : le numéro de la broche
    - mode : soit INPUT (entrée) ou OUTPUT (sortie)

- digitalWrite, digitalRead

  Cette fonction met ou lit un niveau logique HIGH (HAUT équivalent de 5V ou 3V3) ou LOW (BAS équivalent de 0V) sur une broche numérique.

  - digitalWrite(broche, valeur)
  - digitalRead(broche)

- analogWrite, analogRead

  Cette fonction met ou lit la valeur de la tension (impulsion PWM largeur/periode considérée) sur une broche analogique.
  - analogWrite(broche, valeur)
  - analogRead(broche)

- delay()

Cette fonction Réalise une pause dans l'exécution du programme pour la durée (en millisecondes) indiquée en paramètre.