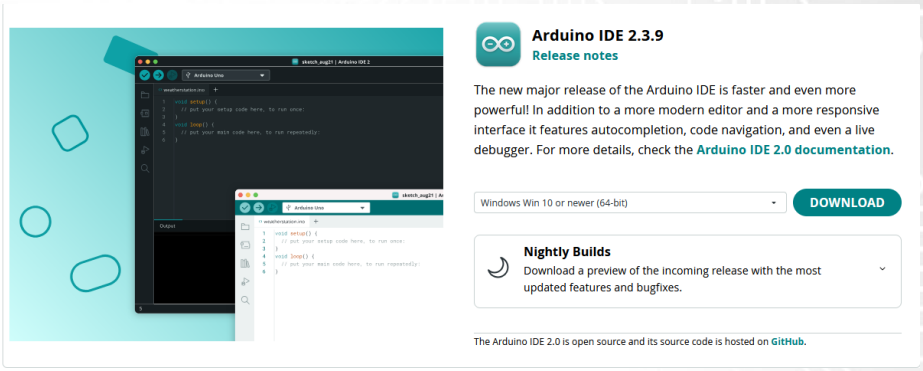
0. Installation de l'IDE Arduino (Windows)

Téléchargement

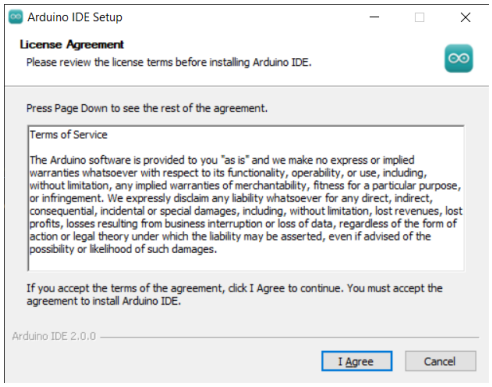
1.Visite le site suivant: https://www.arduino.cc/en/software/
2.Télécharge l'IDE pour votre version de système d'exploitation. (ici windows)



Installation

1.Double-clique sur le fichier arduino-ide_2.3.9.exe pour exécuter le fichier téléchargé.

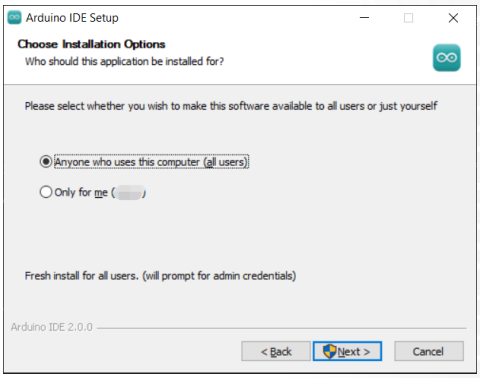
2.Lis l'Accord de Licence et accepte-le.



1 et 2

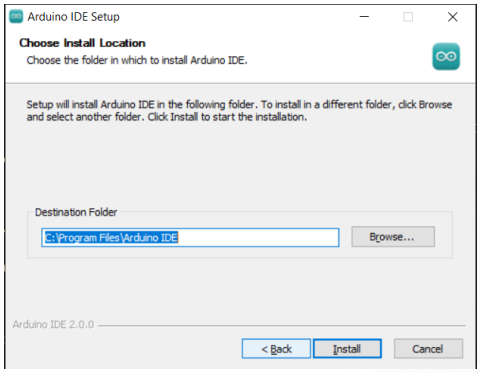
3.Choisis les options d'installation.
Ici Anyone who uses this computers (all users)

4.Choisis le lieu d'installation. Il est recommandé d'installer le logiciel sur un disque autre que le disque système.

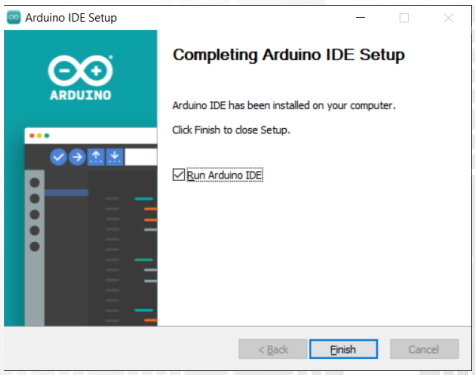


3

5.Puis clique sur terminez.



4



5

0. Installation de l'IDE Arduino (Mac ou Linux)

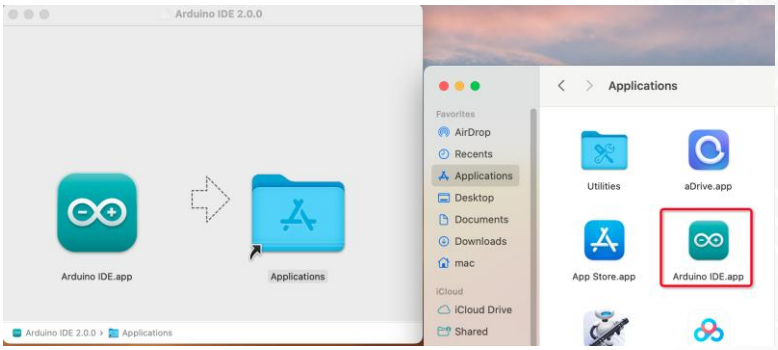
MAC
Téléchargement

1.Visite le site suivant: https://www.arduino.cc/en/software/
2.Télécharge l'IDE pour ta version de système d'exploitation. (ici mac)

Installation

1.Double-clique sur le fichier 2.arduino_ide_2.3.9.dmg téléchargé

3. Copie le fichier Arduino IDE.app dans le dossier Applications, tu verra l'IDE Arduino s'installer avec succès après quelques secondes.



Linux
Téléchargement

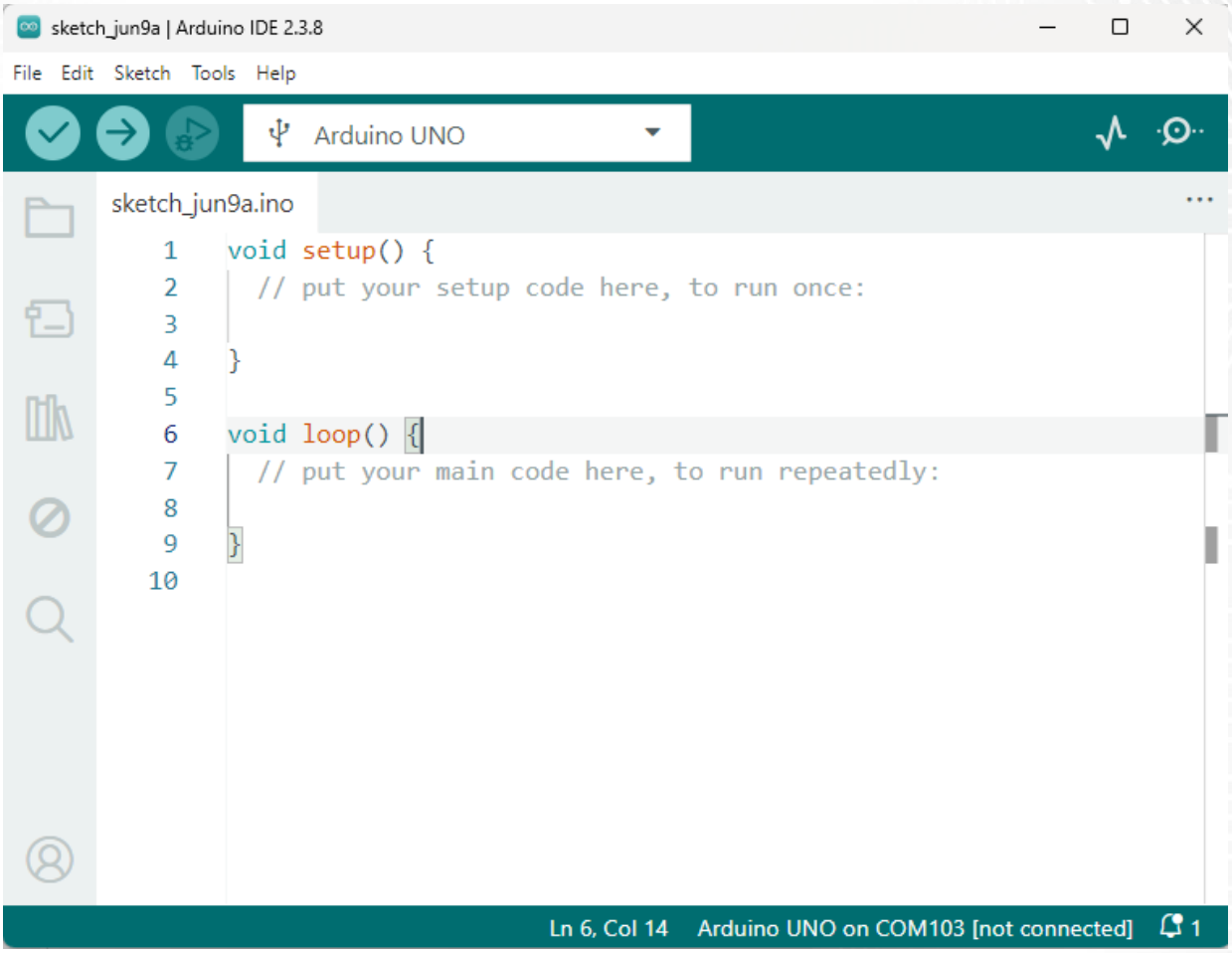
1.Visite le site suivant: https://www.arduino.cc/en/software/
2.Télécharge l'IDE pour ta version de système d'exploitation. (ici Linux)

Installation

1. Fais un clic droit sur le fichier
2. Clique sur **Propriétés**
3. Sélectionne **Permissions**
4. Coche la case **Autoriser à exécuter le fichier comme un programme**
5. Il faut aussi ouvrir un terminal et installer le programme FUSE en tapant les commandes suivantes:
   sudo add-apt-repository universe
   sudo apt install libfuse2
6. Tu dois également rajouter la ligne suivante dans le fichier /etc/udev/rules.d/99-arduino.rules pour autoriser ton IDE à communiquer avec l'USB
SUBSYSTEMS=="usb", ATTRS{idVendor}=="2341", GROUP="plugdev", MODE="0666"

0. Démarrage et anatomie de l'IDE Arduino

Pour démarrer, sur les trois systèmes d'exploitation, il suffit de double cliquer sur l'icone de l'application.
La fenêtre suivante va apparaitre



Je te propose de te montrer les 5 fonctions les plus importantes

Il s'agit de l'icone de précompilation simple qui permet de vérifier que le code que tu écris sera compréhensible par le microcontrôleur.

Il s'agit de l'icone de compilation/téléversement qui va transformer le code en langage machine puis l'envoyer dans le microcontrôleur.

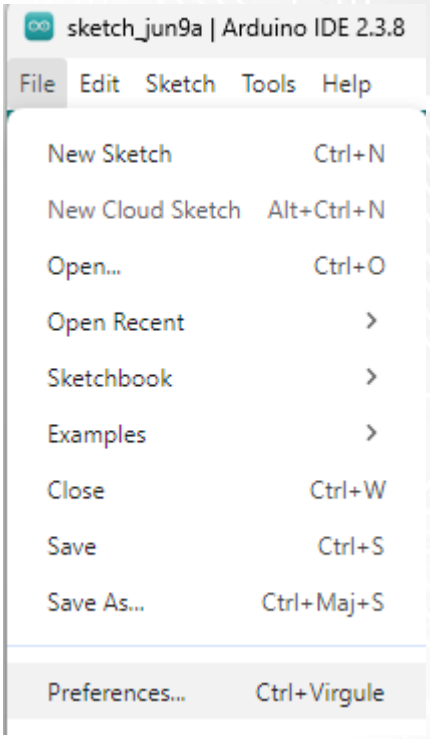
Il s'agit de l'icone d'ajout de nouvelles boards, elle permet de rajouter de nouveaux microcontrôleurs ou de mettre à jour les existants

Il s'agit de l'icone d'organisation des librairies. Les librairies sont des bouts de code que d'autres personnes ont créer et qui permettent de faire des fonctions complexes plus simplement

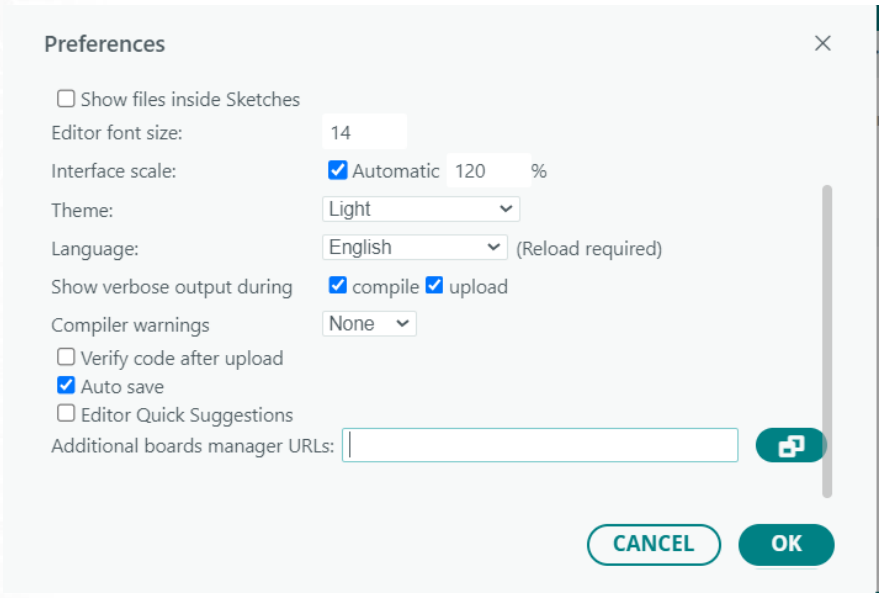
Il s'agit de l'icone de choix de la board, c'est ici que tu choisis quel type de microcontrôleur que tu vas programmer

0. Comment préparer son IDE
Ajout d'un nouveau type de board
Exemple: les ESP32

Après avoir démarré ton IDE,
clique sur Fichier (ou File en fonction de la langue de ton IDE) puis choisis Préférences...



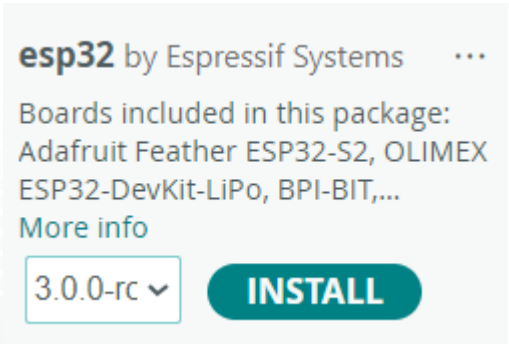
Une nouvelle fenêtre va apparaitre scrolle jusqu'en bas tu auras une case nommée Additional board manager URLs qui sera vide.



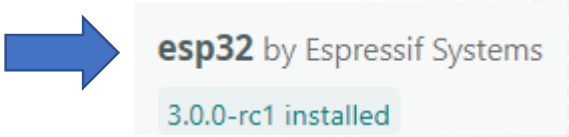
Colle la ligne suivante dans cette case puis clique sur OK
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

Maintenant ferme complètement ton IDE puis redémarre le.

Clique maintenant sur l'icone d'ajout de nouvelles boards et cherche esp32 by Espressif Systems puis clique sur INSTALL



Attends un peu et normalement tu devrais avoir un petit badge qui t'indique que tout est ok

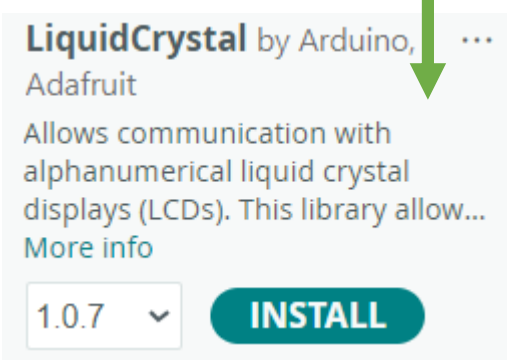


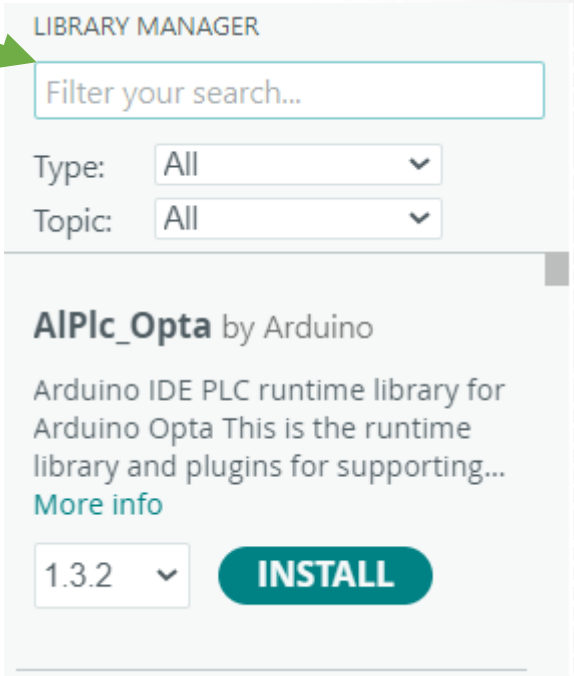
Bravo tu as installé avec succès ton nouveau type de board pour pouvoir démarrer tes projets dessus

0. Comment préparer son IDE
Ajout d'une nouvelle librairie

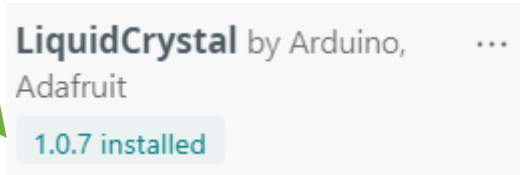
Dans certains projets, il y a la possibilité de réutiliser du code ou fonctionnalités fait par d'autres codeurs et qui ont décidé de partager ceux-ci avec la communauté afin de faciliter la vie des autres et de faire grandir les connaissances. Ils fabriquent alors un recueil de leur code que l'on appelle une librairie.
Pour pouvoir récupérer une librairie, la voie la plus simple est de cliquer sur l'icone d'organisation des librairies

Tu vas arriver sur le Library Manager et maintenant je t'invite à chercher par exemple la librairie LiquidCrystal by Arduino, Adafruit et à cliquer sur INSTALL

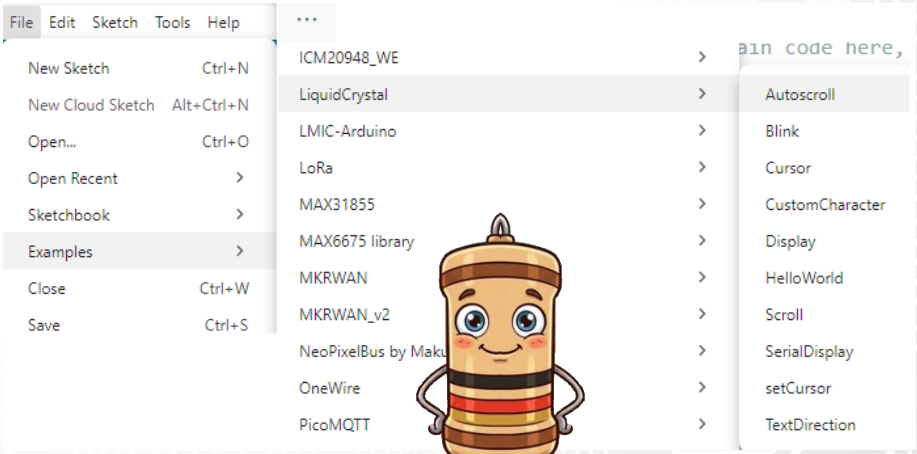




Tu devrais avoir le petit badge installed qui apparait après



Bravo, tu as réussi. Une petite astuce avant de finir, si tu clique sur File → Examples →LiquidCrystal
Tu arrivera sur des exemples de code déjà fourni par le codeur qui a publié la librairie pour te montrer comment celle-ci fonctionne. C'est très utile pour comprendre la logique qui se cache derrière



0. La structure principale d'un programme

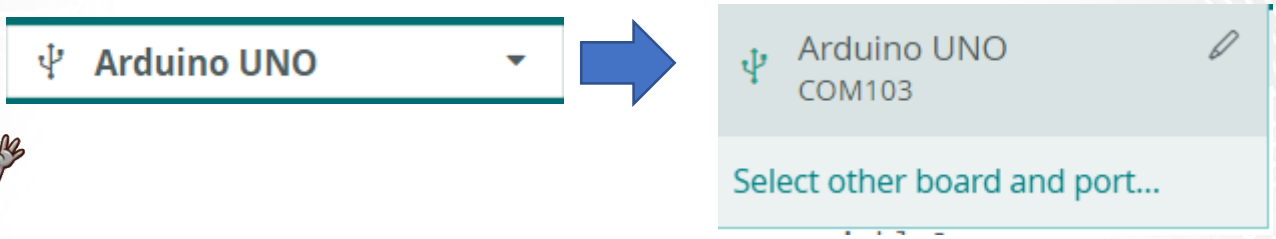
Dans la plupart des cas, un code arduino est composé de quatre grandes sections qui sont dans l'ordre suivant:

1. L'appel des librairies
2. La création des variables
3. Le void setup() où le code est exécuté une seule fois au démarrage (c'est souvent ici que l'on initialise les fonctions)
4. Le void loop () où le code sera exécuté en boucle à l'infini tant que nous alimentons le microcontrôleur

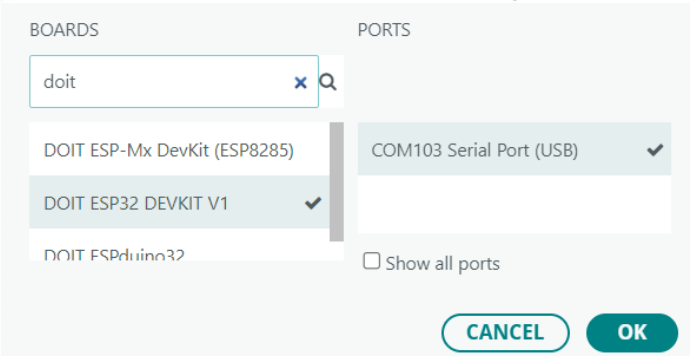
Ne t'inquiète pas, tu aura le temps de comprendre leur fonctionnement lors des exemples.

0. Comment programmer ton microcontrôleur

Une fois que ton code écrit, tu vas pouvoir l'envoyer sur ton microcontrôleur. Pour cela, rien de plus simple, la première étape est de brancher celui-ci sur ton ordinateur avec un câble USB puis si c'est la première fois que tu le fais de cliquer sur l'icone de sélection puis sur Select other board and port



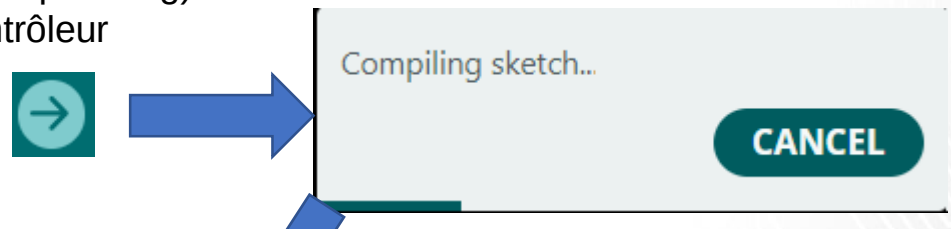
Il faut maintenant choisir ta board (pour nous on va chercher DOIT ESP32 DEVKIT V1) ainsi que la port de communication ou port COM (ici, nous n'avons qu'un périphérique de branché donc c'est facile, en cas de doute débranche l'ESP et regarde quel port COM a disparu) puis clique sur OK

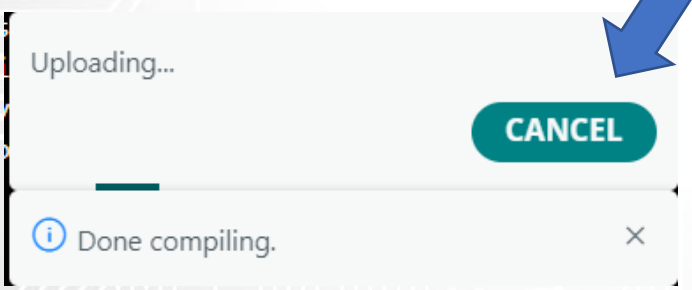


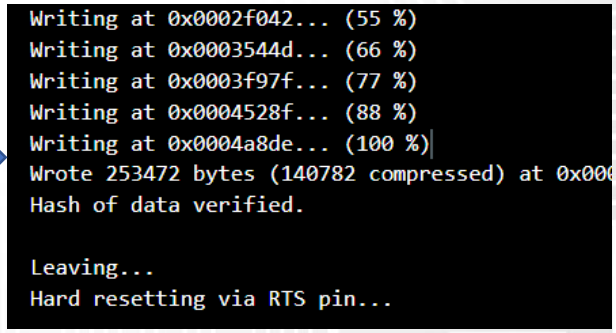
Normalement celui-ci doit apparaitre en gars maintenant dans la fenêtre principale

Si tout est ok; Il ne te reste maintenant plus qu'à a appuyer sur le bouton de compilation/téléversement qui va transformer le code en langage machine puis l'envoyer dans le microcontrôleur en passant par 3 étapes:

- Compilation du code
- Téléversement (ou Uploading)
- Reset du microcontrôleur





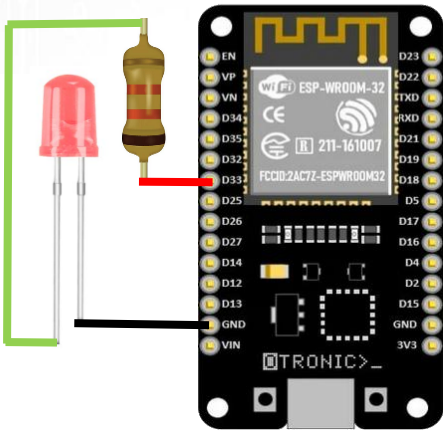


Bravo tu as réussi a programmer ton microcontrôleur avec succès!

1. Apprendre à piloter une LED en tout ou rien (digitalWrite)

Le montage

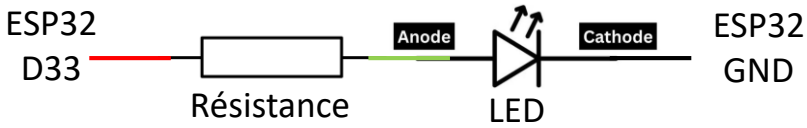
Liste du matériel
- 1 ESP32 Wroom DevKit
- 1 ou 2 LED
- 1 ou 2 résistance de 100 Ohms
- 1 breadboard
- des fils



Petite astuce

Pour brancher correctement ta LED, il faut que **la patte la plus longue** (appelé anode) soit reliée **à la résistance,** et **la patte la plus courte** (appelé cathode) soit reliée **sur le G de la carte**.

Si ton montage ne clignote pas après avoir téléversé ton programme**, inverse les pattes de ta LED**, il y a une forte chance que le problème vienne de là.



Le code (fichier LEDToR.ino)

Cette ligne de code te permet de dire au microcontrôleur qu'il faut allumer ta LED en mettant la sortie 33 à l'état haut (*high* en anglais)

Cette ligne de code te permet de dire au microcontrôleur qu'il faut éteindre ta LED en mettant la sortie 33 à l'état bas (*low* en anglais)

```
void setup() {
    pinMode(33, OUTPUT);
}

void loop() {
  digitalWrite(33, HIGH);
  delay(1000);
  digitalWrite(33, LOW);
  delay(1000);
 }
```

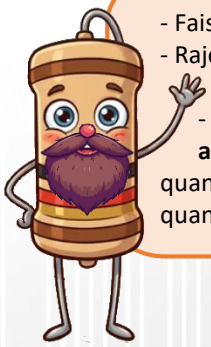
Cette ligne de code te permet de dire au microcontrôleur que ta LED est connectée sur la patte 33 et qu'il faut donc configurer 33 en sortie (*output* en anglais)

Ces lignes de code te permettent de dire au microcontrôleur qu'il faut qu'il fasse à chaque fois une pause d'une seconde (*1000 millisecondes*) avant de continuer le déroulement de son

Le savais-tu ?

Par tradition, faire clignoter une LED est souvent le **premier code** que teste un électronicien lorsqu'il reçoit un nouveau microcontrôleur. C'est l'équivalent en électronique du « Hello World » des informaticiens, cela permet de saluer le nouveau monde de possibilité qui s'ouvre à nous et de tester que les outils de programmations sont bien configurés avant de démarrer des codes plus complexes !

Les défis de Résisti

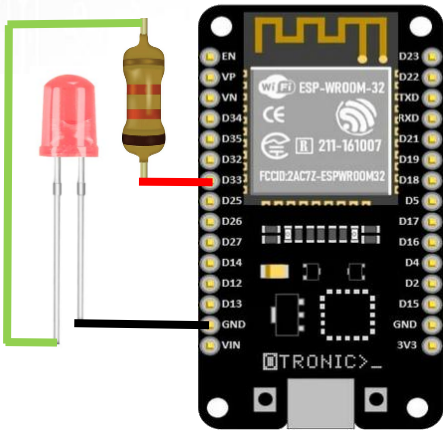


- Fais clignoter la LED **plus vite**
- Rajoute une **deuxième LED** et fait la clignoter **en même temps**
- Maintenant fait les clignoter **alternativement** c'est-à-dire quand **LED1 allumée**, **LED2 éteinte** et quand **LED1 éteinte**, **LED2 allumée**

2. Apprendre à piloter une LED en proportionnel (analogWrite)

Le montage

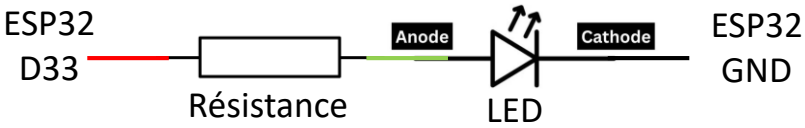
Liste du matériel
- 1 ESP32 Wroom DevKit
- 1 ou 2 LED
- 1 ou 2 résistance de 100 Ohms
- 1 breadboard
- des fils



Petite astuce

Pour brancher correctement ta LED, il faut que **la patte la plus longue** (appelé anode) soit reliée **à la résistance,** et **la patte la plus courte** (appelé cathode) soit reliée **sur le G de la carte**.

Si ton montage ne s'allume pas après avoir téléversé ton programme**, inverse les pattes de ta LED**, il y a une forte chance que le problème vienne de là.



Le code (fichier LEDPropor.ino)

Cette ligne de code te permet de dire au microcontrôleur qu'il faut éteindre ta LED car on lui envoie 0 (voir le savais-tu?)

Cette ligne de code te permet de dire au microcontrôleur qu'il faut allumer la led à moitié

Cette ligne de code te permet de dire au microcontrôleur qu'il faut allumer à 100% la led

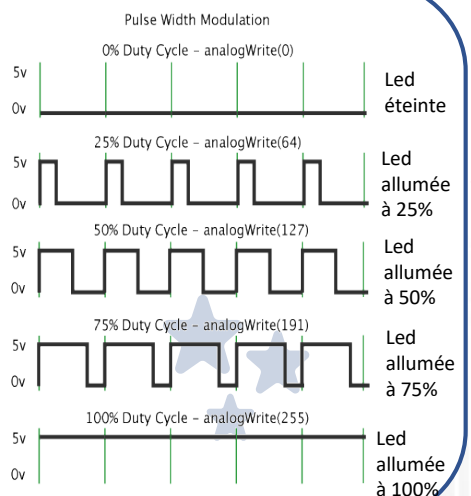
```
void setup() {
    pinMode(33, OUTPUT);
}

void loop() {
  analogWrite (33, 0);
  delay(3000);
  analogWrite(33, 125);
  delay(3000);
  analogWrite(33, 255);
  delay(3000);

 }
```

Cette ligne de code te permet de dire au microcontrôleur que ta LED est connectée sur la patte 33 et qu'il faut donc configurer 33 en sortie (*output* en anglais)

Ces lignes de code te permettent de dire au microcontrôleur qu'il faut qu'il fasse à chaque fois une pause d'une seconde (*3000 millisecondes*) avant de continuer le déroulement de son

Le savais-tu ?

Ici on utilise une PWM signifiant « Pulse Width Modulation ». En français, on le traduit par « Modulation de Largeur d'Impulsion » (MLI). En fait, un signal PWM est un signal électrique de fréquence donnée, et dont le rapport cyclique (Duty cycle) peut varier dans le temps. Le rapport cyclique étant le rapport entre le temps où le signal sera à « l'état haut », et le temps d'une période complète de ce signal (la somme du temps passé « à l'état haut » + le temps passé à « l'état bas », en fait). C'est tellement rapide que l'on ne voit pas les clignotements de la led mais la moyenne du signal. Astuce, pour un microcontrôleur la valeur 0 = 0% et la valeur 255 = 100%



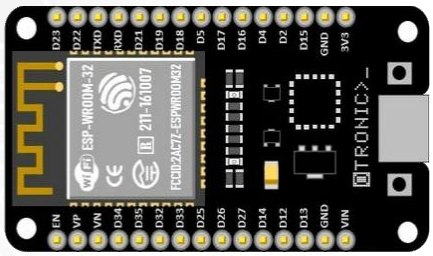
Les défis de Résisti

- Ajoute une deuxième led et allume la avec la même proportion que la première
- Ajoute une deuxième led et allume la avec la même proportion inverse de la première

3. Apprendre à communiquer avec le PC (fonction Serial + Serial Monitor)

Le montage

Liste du matériel
- 1 ESP32 Wroom DevKit



Petite astuce

Si tu ne reçoit pas de signal ou des caractères bizarre, vérifie le baud rate, c'est souvent de là que vient le problème. (voir le savais-tu?)

Le code (fichier TestEnvoiSerial.ino)

On initialise un compteur à zéro pour montrer chaque boucle faite par la suite

Cette ligne permet de dire à quelle vitesse on communique avec le PC et de démarrer la communication (voir le Savais-tu?)

Cette ligne permet d'écrire sur un message

Cette ligne permet d'écrire sur un message et de revenir à la ligne

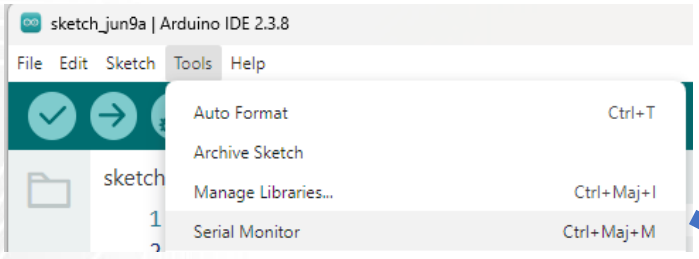
Ici on ajoute 1 au total du compteur

Ces lignes de code te permettent de dire au microcontrôleur qu'il faut qu'il fasse à chaque fois une pause d'une seconde (*3000 millisecondes*) avant de continuer le déroulement de son

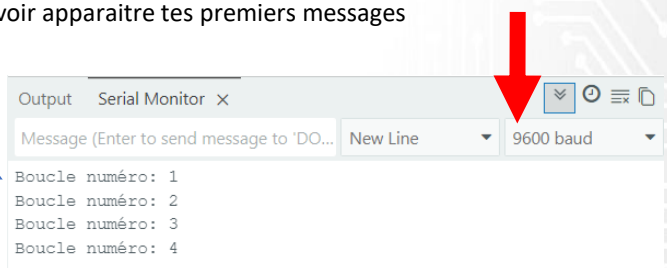
```
int compteur = 0;

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

void loop() {
  Serial.print(F("Boucle numéro: "));
  Serial.println(compteur);
  compteur = compteur + 1;
  delay(3000);
}
```

Et du côté du PC?

Clique sur Tools puis Serial Monitor



Choisis le bon baudrate et normalement tu devrais voir apparaitre tes premiers messages



Le savais-tu ?

Le « Baudrate » est la vitesse de communication entre deux appareils plus il est grand plus tu peux envoyer d'information par secondes et c'est important que ce soit le même pour les deux. Pour illustrer, si nous avions deux baudrates différents entre les appareils cela reviendrait à faire une balade avec un de tes amis où toi tu marche et ton ami court à fond donc cela serait impossible de discuter ensemble.

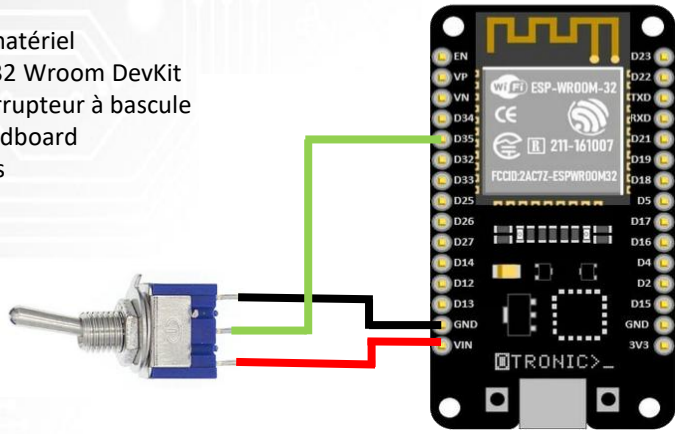
Les défis de Résisti

- Change la vitesse de communication (Baudrate)
- Essaie de rajouter des lignes

4. Apprendre à lire une entrée numérique tout ou rien (digitalRead)

Le montage

Liste du matériel
- 1 ESP32 Wroom DevKit
- 1 interrupteur à bascule
- 1 breadboard
- des fils



Petite astuce

Il faut bien brancher toutes les pattes de l'interrupteur et non pas que le VCC ou le GND car une patte en l'air ne veux pas dire 0 ou 1, c'est un état indéterminé que le microcontrôleur ne sais pas gérer en temps normal.

Le code (fichier LectureDigitale.ino)

Cette ligne de code te permet de dire au microcontrôleur que l'entrée à lire sera la 14

Cette ligne permet de lire l'état du bouton et de l'enregistrer dans une variable pour la garder

```
int Button = 35;

void setup() {
  Serial.begin(9600);
  pinMode(Button, INPUT);
}

void loop() {
  int buttonState = digitalRead(Button);
  Serial.println(buttonState);
  delay(3000);
}
```

Cette ligne permet de dire à quelle vitesse on communique avec le PC et de démarrer la communication

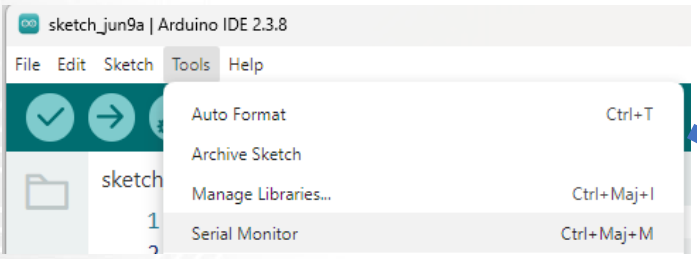
Cette ligne permet de dire que l'on passe la broche 14 en mode entrée

Cette ligne permet d'envoyer l'état du bouton au PC

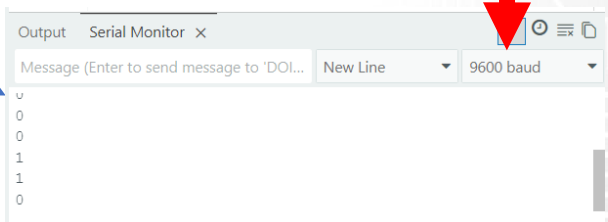
Ici on fait une pause de 3 secondes avant de recommencer un cycle programme

Et du côté du PC?

Clique sur Tools puis Serial Monitor



Choisis le bon baudrate et normalement tu devrais voir apparaitre tes premiers messages.
Bouge l'interrupteur et voie ce qu'il se passe



Le savais-tu ?

Ce code n'est pas idéal pour piloter un système à partir d'un bouton poussoir. Il est souvent utilisé pour connaitre l'état d'un actionneur

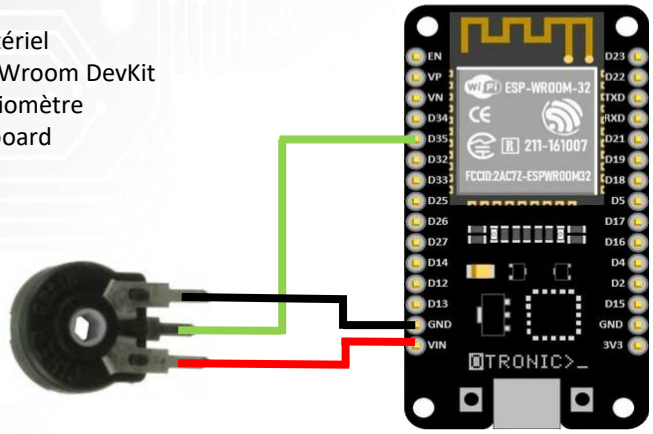
Les défis de Résisti

- Ajoute une led à ton montage et allume là quand le bouton est à l'état 1 et éteint la quand il est à l'état 0
- Rajoute un deuxième bouton à ton montage et envoie son état.

5. Apprendre à lire une entrée analogique (AnalogRead)

Le montage

Liste du matériel
- 1 ESP32 Wroom DevKit
- 1 potentiomètre
- 1 breadboard
- des fils



Petite astuce

Il faut bien brancher VCC et GND aux extrémités et la patte du milieu sur la broche qui va nous permettre de lire la valeur de la tension. Si cela n'est pas fait correctement nous verrons soit toujours 0 soit toujours 5V.

Le code (fichier LectureAnalogique.ino)

Cette ligne de code te permet de dire au microcontrôleur que l'entrée à lire sera la 35

```
int sensorPin = 35;

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

void loop() {
  int sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);
  delay(2000);
}
```

Cette ligne permet de dire à quelle vitesse on communique avec le PC et de démarrer la communication

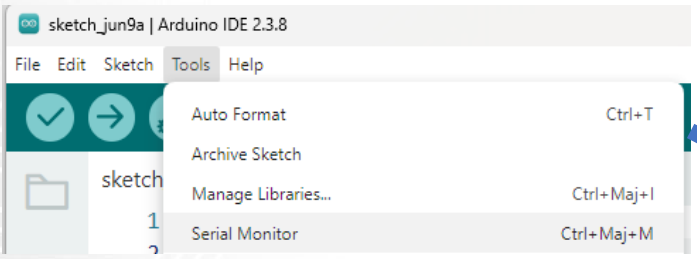
Cette ligne permet de lire la valeur de l'entrée et de l'enregistrer dans une variable pour la garder

Cette ligne permet d'envoyer la valeur du bouton au PC

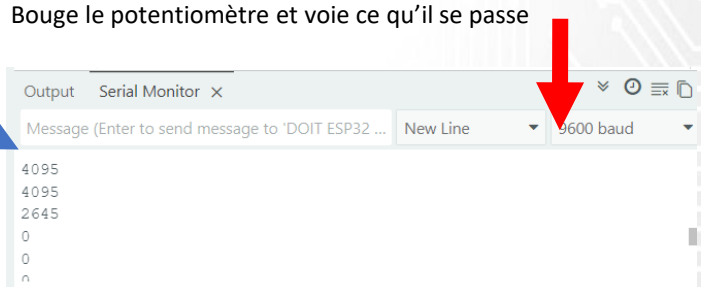
Ici on fait une pause de 2 secondes avant de recommencer un cycle programme

Et du côté du PC?

Clique sur Tools puis Serial Monitor

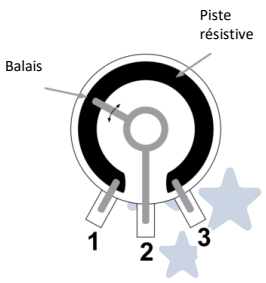


Choisis le bon baudrate et normalement tu devrais voir apparaitre tes premiers messages.
Bouge le potentiomètre et voie ce qu'il se passe



Le savais-tu ?

L'intérieur d'un potentiomètre est composé d'une bande résistive qui est parcourue par un balais. La valeur de la résistance entre les pattes 1 et 3 est toujours fixe. On l'utilise souvent comme un diviseur de tension.



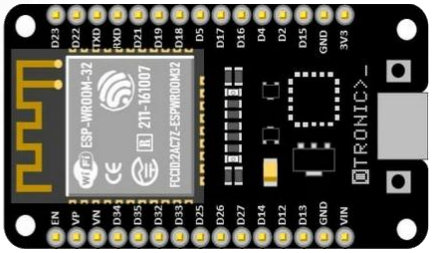
Les défis de Résisti

- Ajoute un deuxième potentiomètre
- Essaie de piloter une led avec les données issues du potentiomètre

6. Faire des répétitions (la boucle For)

Le montage

Liste du matériel
- 1 ESP32 Wroom DevKit
- 1 interrupteur à bascule
- 1 breadboard
- des fils



Petite astuce

La boucle for est une base de la programmation et tiens en une ligne
S'il y a une erreur vérifie bien les conditions que tu as mis souvent c'est la condition de fin qui peux poser problème

Le code (fichier BoucleFor.ino)

**Cette ligne est le cœur de la boucle for et se décompose de la façon suivant (L'initialisation de l'index; La condition de fin; La façon dont on augmente ou diminue les index)
Ici, on démarre de zéro et on va jusqu'à 5 en augmentant l'index de 1 à chaque fois et on exécutera le code situé entre ses deux accolades (ici en rouges) à chacune de ses boucles**

Ici on fait une pause de 5 secondes avant de recommencer le programme complet

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

void loop() {
  for (int i = 0; i <= 5; i=i+1){
    Serial.println(i);
    delay(1000);
  }
delay(5000);
}
```

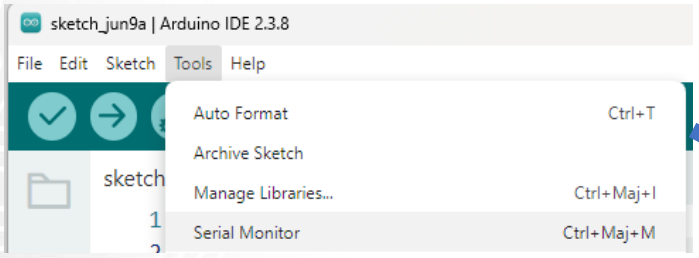
Cette ligne permet de dire à quelle vitesse on communique avec le PC et de démarrer la communication

Cette ligne permet d'envoyer l'état de l'index au PC

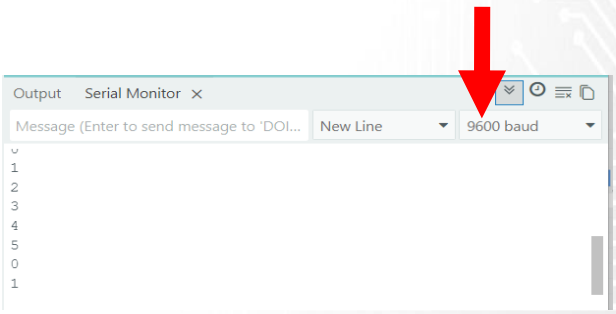
Ici on fait une pause de 1 seconde avant de recommencer un cycle dans la boucle for

Et du côté du PC?

Clique sur Tools puis Serial Monitor



Choisis le bon baudrate et observe ce qu'il se passe



Le savais-tu ?

La boucle for est utilisée afin de faire des taches répétitives et ainsi économiser de l'espace de programmation qui est précieux dans un microcontrôleur.

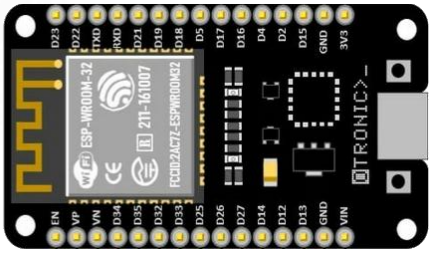
Les défis de Résisti

- Fais en sorte d'aller jusqu'à 10
- Fais un décompte de 10 à 0.
- Compte de 2 en 2 de 0 à 10

7. Vérifier une situation avec une condition (la fonction If… Else…)

Le montage

Liste du matériel
- 1 ESP32 Wroom DevKit
- 1 interrupteur à bascule
- 1 breadboard
- des fils



Petite astuce

La fonction If Else (appelez aussi en français Si Sinon) est utilisée pour vérifier des situations.
On peux mettre un If sans avoir forcément de Else derrière mais pas de Else sans If avant

Le code (fichier TestIfElse.ino)

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  for (int i = 0; i <= 5; i = i + 1) {
    if (i == 2){
      Serial.println("Tadam");
    }
    else {
      Serial.println("Oups");
    }
    delay(1000);
  }
  delay(5000);
}
```

Cette ligne permet de dire à quelle vitesse on communique avec le PC et de démarrer la communication

Cette ligne permet de comparer une situation et d'exécuter le code qui se trouve entre les accolades (rouge ici)

Ici si i est égale à 2 on envoie le message Tadam

Cette ligne intervient si la condition précédente n'a pas été remplie et donc exécute le code qui se trouve entre les accolades (rose ici)

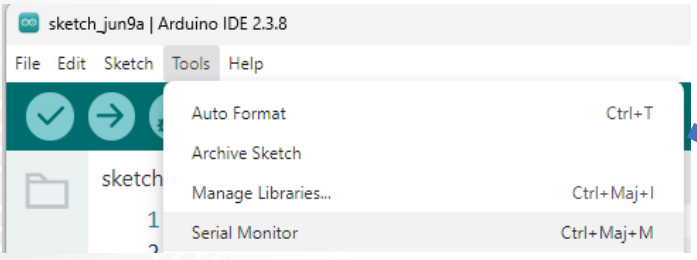
Ici si i n'est pas égale à 2 on envoie le message Oups

Ici on fait une pause de 1 seconde avant de recommencer un cycle dans la boucle for

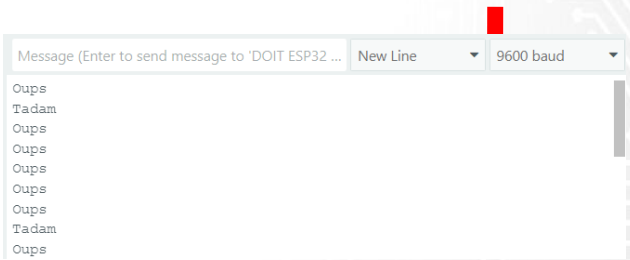
Ici on fait une pause de 5 secondes avant de recommencer le programme complet

Et du côté du PC?

Clique sur Tools puis Serial Monitor



Choisis le bon baudrate et observe ce qu'il se passe



Le savais-tu ?

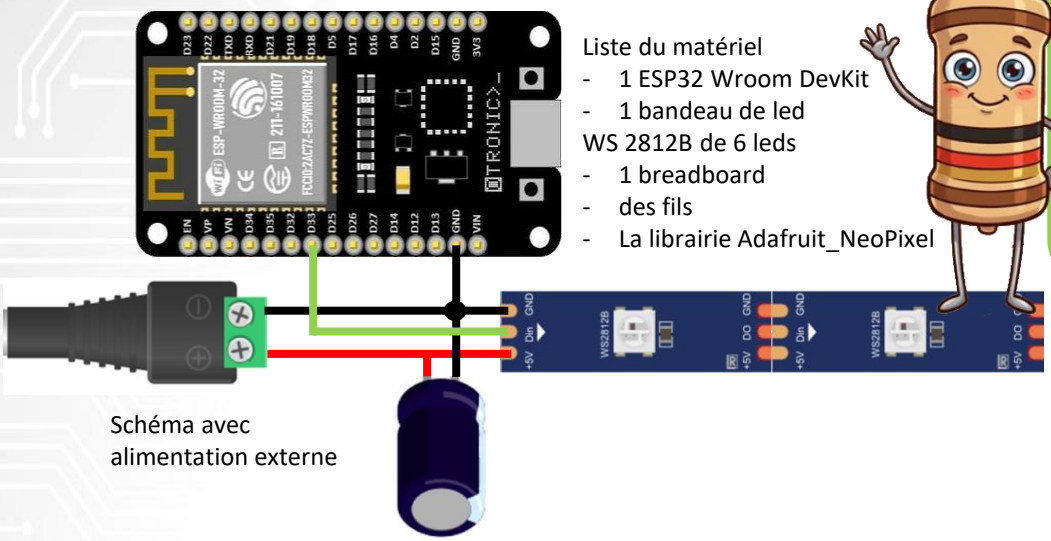
La fonction If Else peut s'imbriquer et dans un If on peux avoir un sous if qui lui aussi peux avoir un soussous if mais attention au bazar et au bug qui peux trainer et qui attend le bon moment pour faire planter ton microcontrôleur

Les défis de Résisti

- Ajoute des conditions et des messages différents
- ajoute une led et allume la à 4.

7. Apprendre à piloter un bandeau de LED RGB adressable de type WS2812B

Le montage



Liste du matériel
- 1 ESP32 Wroom DevKit
- 1 bandeau de led WS 2812B de 6 leds
- 1 breadboard
- des fils
- La librairie Adafruit_NeoPixel

Schéma avec alimentation externe

Petite astuce

Dans l'exercice comme il a moins de 10 leds, tu peux directement brancher le bandeau leds de la façon suivante sans avoir besoin d'alimentation externe ni de condensateur.

ESP32	Bandeau
GND	GND
Vin	+5V
D33	Din

Le code (fichier BandeauLED.ino)

Cette ligne appelle la librairie de pilotage des leds

```
#include <Adafruit_NeoPixel.h>

#define PIN_WS2812B 33
#define NUM_PIXELS 6

int rouge = 0;
int bleu = 0;
int vert = 0;

Adafruit_NeoPixel ws2812b(NUM_PIXELS, PIN_WS2812B, NEO_GRB + NEO_KHZ800);

void setup() {
  ws2812b.begin();
}

void loop() {
  rouge = 255;
  vert = 0 ;
  bleu = 0;
  for (int pixel = 0; pixel < NUM_PIXELS; pixel++) {
    ws2812b.setPixelColor(pixel, ws2812b.Color(rouge, vert, bleu));
  }
  ws2812b.show();
  delay(5000);
}
```

Cette ligne de code te permet de dire au microcontrôleur que le bandeau LED est connectée sur la patte 33

Cette ligne de code te permet de dire combien de LED à la bandeau

Ici on déclare les variables qui nous servirons dans le programme

Cette ligne de code te permet à la librairie de savoir quel bandeau elle doit piloter

Cette ligne de code te permet d'initialiser le bandeau de leds

Ici on défini l'intensité de chaque couleur 0 = éteinte et 255 = allumée au maximum

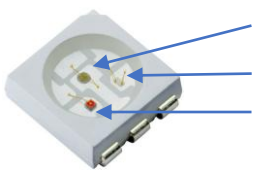
Ces trois lignes permettent de parcourir tous les leds les uns après les autres pour leur appliquer la couleur choisie.
Petite astuce, la première led est à l'index 0, la seconde à l'index 1, etc...

Ici on indique au microcontrôleur qu'il doit appliquer les modifications de couleur

Ici on fait une pause de 5 secondes

Le savais-tu ?

Une led RGB est composée en réalité de trois petites leds que l'on pilote indépendamment.



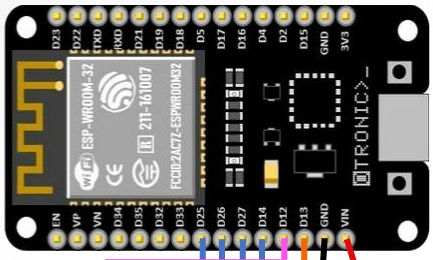
Les défis de Résisti

- Change la couleur de toutes les leds
- Essaie de baisser l'intensité des leds
- Essaie de faire en sorte qu'il n'y ait une couleur différentes pour chaque led
- Essaie de faire clignoter les leds

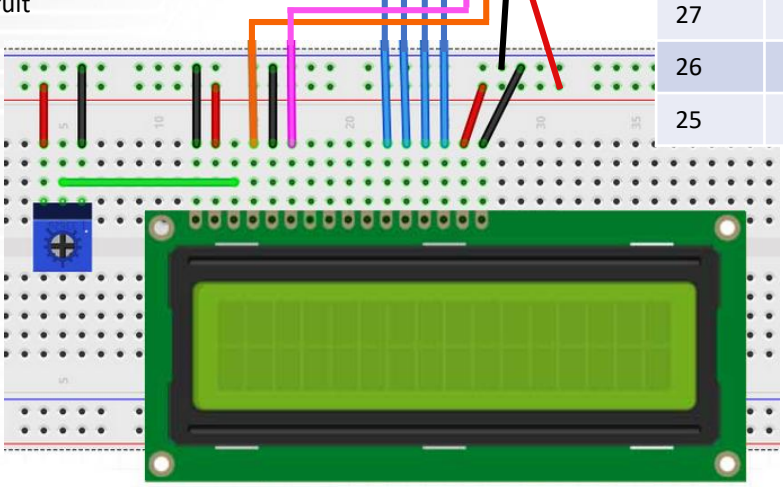
8. Apprendre à piloter un écran LCD 2 lignes avec un ESP32

Le montage

Liste du matériel
- 1 ESP32 Wroom DevKit
- Un écran LCD 2x16
- 1 potentiomètre de 10kOhms ou une résistance de 1kOhms
- La librairie LiquidCrystal de AdaFruit

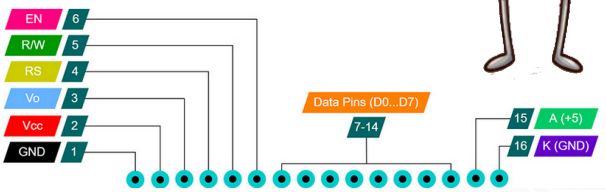


ESP32	LCD
13	RS
12	EN
14	D4
27	D5
26	D6
25	D7



Petite astuce

Le potentiomètre sert à régler le contraste de l'écran finement mais tu peux le remplacer par une résistance de 1kOhms entre l'écran et le GND. Tu peux également piloter le rétroéclairage si tu mets celui-ci sur une pin que tu pilotera en PWM.



Noms des pins de l'écran LCD

Le code (fichier TestLCD.ino)

Cette ligne appelle la librairie de pilotage du LCD

Cette ligne de code te permet de dire au microcontrôleur où sont connectées les pattes de l'écran

On initialise un compteur à zéro pour montrer chaque boucle faite par la suite

Cette ligne permet de dire à la librairie que nous avons un écran de 2 lignes avec 16 caractères à chaque fois

Cette ligne sert à « nettoyer » l'écran, c'est-à-dire le vider

Ces lignes servent à marquer ce que l'on veux écrire

Ici on ajoute 1 au total du compteur

On indique ici au microcontrôleur où l'on veux commencer à écrire avec la logique suivante (colonne, ligne) Par exemple le point 0,0 indique (première colonne, première ligne) et 0,1 indique (première colonne, deuxième ligne)

Ici on fait une pause de 1 seconde

```
#include <LiquidCrystal.h>

LiquidCrystal My_LCD(13, 12, 14, 27, 26, 25);

int compteur = 0;

void setup() {
  My_LCD.begin(16, 2);
}

void loop() {
  My_LCD.clear();
  My_LCD.setCursor(0,0);
  My_LCD.print("Bonjour Resisti");
  My_LCD.setCursor(0,1);
  My_LCD.print(compteur);
  compteur = compteur +1;
  delay(1000);
}
```

Le savais-tu ?

LCD veux dire Liquid Crystral Display ou Affichage à Cristaux Liquides. Cette technologie existe depuis les années 70 et les Game & Watch étaient les premières consoles de jeux vidéos portables à utiliser cette technologie car elle ne consomme quasiment pas d'énergie donc idéale pour des appareils à piles.

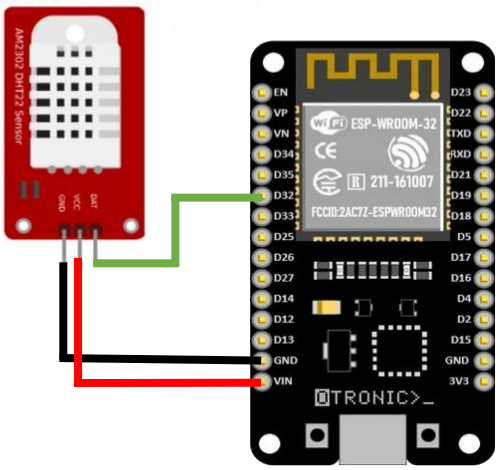
Les défis de Résisti

- Inverse la ligne du haut et la ligne du bas
- Change de place le compteur pour qu'il soit au milieu
- Essaie d'écrire de droite à gauche

9. Apprendre à lire un capteur de température/humidité de type DHT22

Le montage

Liste du matériel
- 1 ESP32 Wroom DevKit
- 1 capteur DHT22
- 1 breadboard
- des fils
- La librairie DHT sensor de Adafruit



Petite astuce

Attention, le DHT22 à trois pattes! Deux pour son alimentation (VCC en rouge et GND en noir) et une pour la lecture des données (Data souvent d'une autre couleur par exemple verte). Ne les confond pas car sinon cela peut détruire le capteur ou le microcontrôleur

Le code (fichier TestDHT22.ino)

```
#include "DHT.h"

#define DHTPIN 32
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

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

void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println(F("Probleme de lecture!"));
    return;
  }

  Serial.print(F("Humidite: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
}
```

Cette ligne appelle la librairie de lecture du capteur

Cette ligne de code te permet de dire au microcontrôleur que le capteur est sur la broche 32

Cette ligne permet de dire au microcontrôleur quel type de DHT il s'agit (ici un DHT22)

Cette ligne permet au microcontrôleur de définir la bonne fonction à utiliser

Cette ligne permet de dire au microcontrôleur de lancer la communication avec le DHT22

Cette ligne permet au microcontrôleur d'initialiser la communication avec le PC

Ici on fait une pause de 2 secondes entre chaque lecture

Ici on demande une lecture de l'humidité

Ici on demande une lecture de la température

Ces trois lignes sont là pour vérifier que le capteur a bien été lu et de renvoyer un message d'erreur si ce n'est pas le cas

Ces 5 lignes permettent d'envoyer au PC les données lues par le capteur

Le savais-tu ?

En Europe, nous utilisons les degrés Celsius avec comme repères pour 0°C le point de congélation de l'eau et pour 100°C le point d'ébullition de l'eau au niveau de la mère mais aux Etats-Unis, ils utilisent les degrés Fahrenheit avec comme échelle pour 0°F la température à la plus basse qu'il ait observée en 1709 dans la ville de Dantzig et pour 100°F la température corporelle d'un cheval sain… Bonjour la précision!

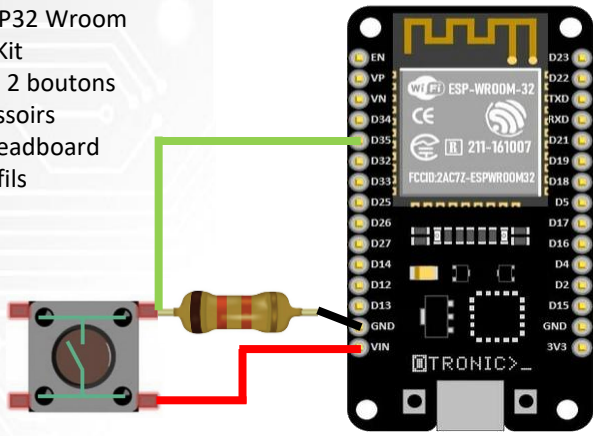
Les défis de Résisti

- Essaie de rajouter une alerte au message quand la température ou l'humidité dépasse une certaine valeur.
- Essaie de rajouter une led à ton montage et de l'allumer quand la température ou l'humidité dépasse une certaine valeur

10. Apprendre à lire l'état d'un bouton avec un anti-rebond

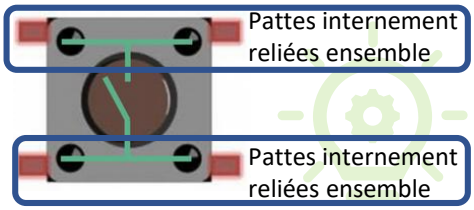
Le montage

Liste du matériel
- 1 ESP32 Wroom DevKit
- 1 ou 2 boutons poussoirs
- 1 breadboard
- des fils



Petite astuce

Si ton bouton poussoir à quatre pattes il y a de fortes chance quelles soient reliées 2 à 2 comme pour le schéma suivant:



Tu peux utiliser un ohmètre pour déterminer comment il est fait à l'intérieur si tu n'as pas sa documentation technique

Le code (fichier Testbutton.ino)

Cette ligne permet de dire au microcontrôleur sur quelle broche est le bouton

Ces trois lignes permettent de donner un état initial du bouton au microcontrôleur

Cette ligne permet de régler le délai de Debounce (voir le savais tu?)

Cette ligne permet de dire que l'on passe la broche du bouton en entrée

Cette ligne initialise la communication avec le PC

Cette ligne va lire l'état du bouton

Si on détecte que l'état actuelle du bouton est différent de celui enregistré sur la boucle précédente, on active le mode Debounce en réinitilisant le timer

On attend que le bouton soit enfin stabilisé

Si après le temps de Debounce le signal lu est toujours différent de l'état du bouton enregistré.

On vient enregistrer le nouvel état

On vient faire une action en fonction de l'état du bouton

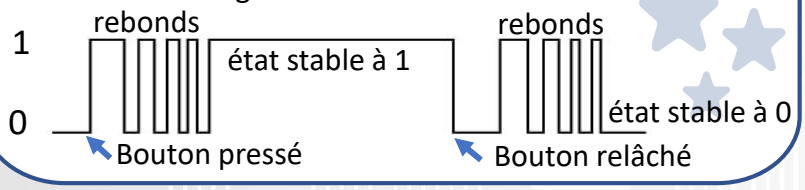
On vient enregistrer l'état du bouton en fin de boucle afin de le comparer sur la prochaine boucle

```
const int buttonPin = 35;
int buttonState = HIGH;
int lastButtonState = HIGH;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 20;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  int reading = digitalRead(buttonPin);

  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;
      if (buttonState == HIGH) {
        Serial.println("Bouton appuye");
      }
      if (buttonState == LOW) {
        Serial.println("Bouton relache");
      }
    }
  }
  lastButtonState = reading;
}
```

Maintenant vas voir ce qu'il se passe du côté du PC avec le serial monitor

Le savais-tu ?

Un bouton mécanique lorsqu'on le presse ou qu'on le relâche, il rebondit un peu comme un ressort et donc son signal oscille entre 0 et 1 ce qui peux apporter un bug. C'est pour cela que l'on rajoute une fonction Debounce qui va attendre pendant quelques millisecondes la stabilisation du signal avant de confirmer l'état du bouton.



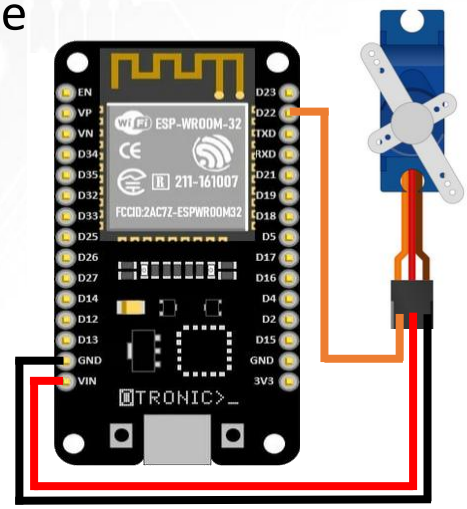
Les défis de Résisti

- Ajoute une led à ton montage et allume là quand tu appuie sur le bouton et éteint la quand tu relance
- Ajoute une led à ton montage et allume là quand tu appuie sur le bouton et éteint la quand tu rappuie dessus
- Rajoute un deuxième bouton poussoir à ton montage

11. Apprendre à piloter un servomoteur

Le montage

Liste du matériel
- 1 Wemos D1 mini
- 1 ou 2 LED
- 1 ou 2 résistance de 100 Ohms
- 1 breadboard
- des fils
- La librairie ESP32servo de Kevin Harrington



Petite astuce

Attention, le servomoteur à trois pattes! Deux pour son alimentation (VCC en rouge et GND en noir/marron) et une pour son contrôle (d'une autre couleur par exemple orange). Ne les confond pas car sinon cela peut détruire le servomoteur ou le microcontrôleur

Le code (fichier Servomoteur.ino)

```
#include <ESP32Servo.h>
#define PIN_SG90 22
Servo sg90;
int pos = 0;

void setup() {
  sg90.setPeriodHertz(50);
  sg90.attach(PIN_SG90, 500, 2400);}

void loop() {
  pos = 0;
  sg90.write(pos);
  delay(5000);
  pos = 180;
  sg90.write(pos);
  delay(5000);
}
```

Cette ligne appelle la librairie de pilotage du servomoteur

Cette ligne sert à définir le type de servomoteur utilisé

Cette ligne de code te permet de dire au microcontrôleur que le servomoteur est sur la broche 22

Cette ligne sert à définir à quelle fréquence travaille le servomoteur (voir le savais-tu?)

Cette ligne permet de donner la largeur minimale et maximale de l'impulsion (en µs) pour aller de 0° à 180°

Cette ligne indique que nous voulons la position 0°

Ici on fait une pause de 5 secondes

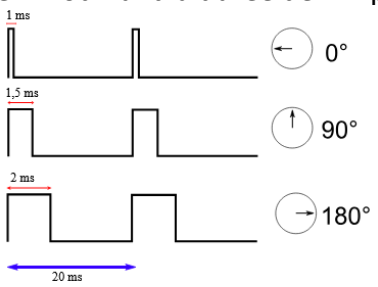
Cette ligne indique que nous voulons la position 180°

Ces deux lignes servent à envoyer au servomoteur la position que l'on veux qu'il prenne

Ici on fait une pause de 5 secondes

Le savais-tu ?

Le servomoteur est piloté en utilisant des petites impulsions que l'on appelle PWM avec une fréquence de 50 Hz. Cela veux dire que l'on produit une impulsion toutes les 20ms. On peut ajuster la position du servomoteur en modifiant la durée de l'impulsion envoyée.



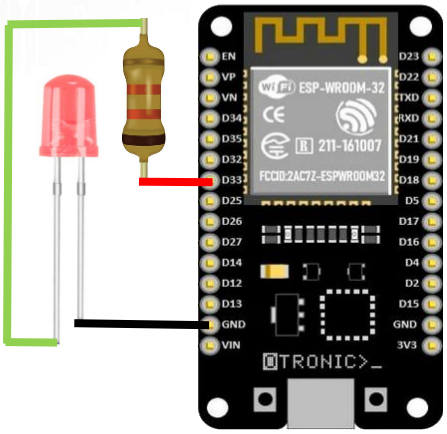
Les défis de Résisti

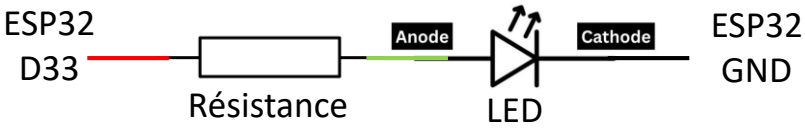
- Essaie de faire bouger le servomoteur de 0° à 180° puis de 180° à 0° sans à-coup.
- Rajoute un bouton et pilote le servomoteur en fonction de son état
- Rajoute un capteur de température et pilote le servomoteur en fonction de sa valeur

11. Apprendre à faire une fonction non bloquante

Le montage

Liste du matériel
- 1 ESP32 Wroom DevKit
- 1 ou 2 LED
- 1 ou 2 résistance de 100 Ohms
- 1 breadboard
- des fils





Petite astuce

La dernière fois nous avions utilisé la fonction delay() pour faire une pause. Le problème est que lorsque l'on utilise cette fonction le microcontroleur se met en pause et cela peut-être critique car celui-ci doit pouvoir gérer parfois plusieurs fonctions en même temps. C'est pour cela que l'on va tester une autre méthode utilisant l'horloge interne du microcontrôleur

Le code (fichier TestDelayMillis.ino)

```
unsigned long currentTime=0;
unsigned long previousTime=0;
bool ledState=LOW;

void setup() {
  pinMode(33,OUTPUT);
}

void loop() {
  currentTime=millis();
  if((currentTime-previousTime)>2000){
    previousTime=currentTime;
    ledState=!ledState;
    digitalWrite(33,ledState);
  }
}
```

Ici on initialise à 0 la variable du temps actuel

Ici on initialise à 0 la variable du temps de la dernière modification

Ici on initialise à l'état éteint la led

Cette ligne de code te permet de dire au microcontrôleur que ta LED est connectée sur la patte 33 et qu'il faut donc configurer 33 en sortie

Ici on appelle la fonction millis() qui nous indique le nombre de millisecondes passée depuis le démarrage du microcontrôleur et on l'enregistre dans la variable temps actuel

Ici on fait une comparaison entre le temps de la dernière fois où l'état de la led a été modifiée et le temps actuel si celui-ci dépasse 2 s, on exécute le code entre les accolades

On enregistre le temps actuel comme nouveau temps de la dernière modification

On écrit sur la sortie le nouvel état

On décide d'inverser l'état de la led par rapport au précédent

Le savais-tu ?

Les microcontrôleurs modernes sont quasiment 1 milliard de fois plus puissant de celui utilisé pour envoyer l'Homme sur le lune. Alors imagine toi le casse-tête que cela devait être quand il fallait générer la poussée, les boosters, le tangage sans que tout plante…

Les défis de Résisti

- Fais clignoter la LED **plus vite**
- Rajoute une **deuxième LED** et fait la clignoter **en même temps**
- Rajoute une **deuxième LED** et fait la clignoter à une vitesse différente