



جامعة سيدي محمد بن عبد الله
Université Sidi Mohamed Ben Abdellah

FACULTÉ DES SCIENCES DHAR EL MAHRAZ
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH

Internet des Objets

Chapitre 3

Programmation des Cartes de développement

Pr. Bouayad Anas

A.U : 2022/2023

Cours adapté pour la 1^{ème} année Master BDSaS- FSDM

Objectifs du chapitre

- ❑ **Programmer les cartes de développement en utilisant :**
 - ❑ **Arduino**
 - ❑ **Python**

Plan

1. Introduction
2. Langage Arduino
3. Python pour RBPi

Introduction

Langages de Programmation pour l'IoT

- ❑ Pour qu'un micro puisse remplir n'importe quelle fonction, nous devons le programmer.
- ❑ Pour ce faire, nous utilisons un langage particulier, généralement choisi en fonction de notre niveau de **connaissance**, de sa **facilité d'utilisation**, des **bibliothèques** disponibles et de la tâche que nous souhaitons.

Langages de Programmation pour l'IoT

❑ Langages de Programmation

TOP IoT PROGRAMMING LANGUAGES

Devices



Gateways



Cloud



Arduino

Présentation de la plateforme Arduino

- ❑ Arduino est le nom d'une entreprise, une communauté et un projet libre (opensource).
- ❑ Ce projet avait été initié en Italie, dans le but de concevoir et fabriquer des solutions matérielles et logiciel à faible coût qui faciliteront le développement d'objets interactifs qui peuvent interagir avec le monde physique.
- ❑ Les cartes Arduino peuvent être achetées préassemblées, ou réalisés par soi-même.



Présentation de la plateforme Arduino

- ❑ Construire des dispositifs qui interagissent avec l'environnement qui les entoure

Capteurs

Températures

Humidité

Pression

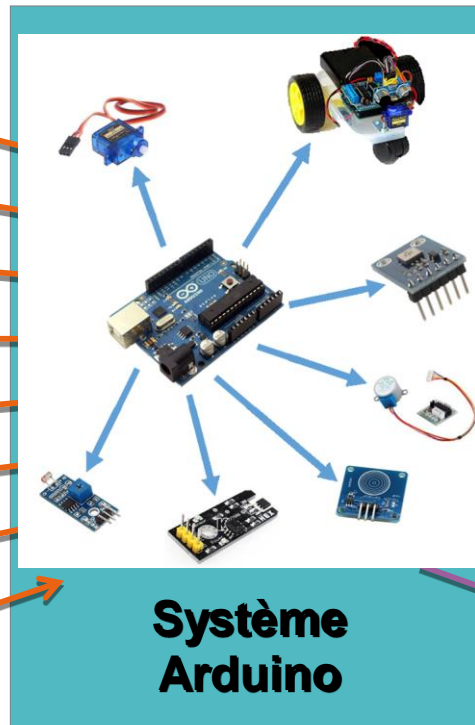
Présence

Distance

Position

Luminosité

...



Actionneurs

Moteurs

Servos

Electrovannes

Pompes

Verins

Résistances chauffantes

Eclairages

...

Présentation de la plateforme Arduino

Diverses Cartes Arduino : Les microcontrôleurs



- ✓ *Adaptée pour les petits projets*
- ✓ *Mise à disposition de nombreux tutoriels*
- ✓ *Limitée en nombre de périphériques externes*

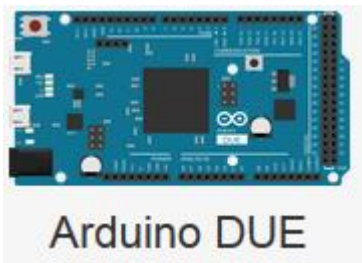
Présentation de la plateforme Arduino

Diverses Cartes Arduino : Les microcontrôleurs

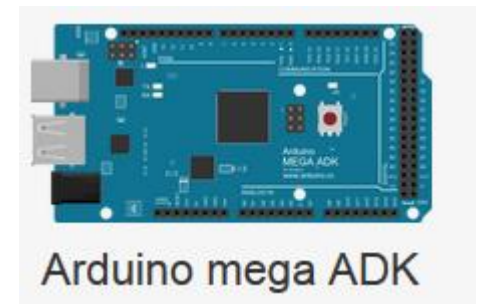


Plus évoluée

- ✓ Plus d'entrées et sorties (54 contre 14),
- ✓ Plus de mémoire (programmes plus complexes)
- ✓ Processeur, plus puissant.



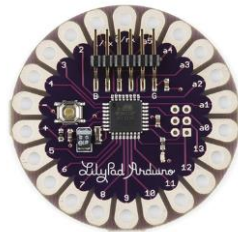
- ✓ Arduino équivalent Mega mais plus puissante pour des calculs plus rapides.



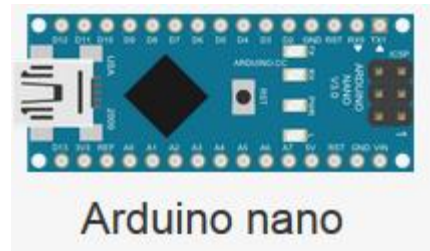
- ✓ caractéristiques l'Arduino Mega)
- ✓ Interagir avec un smartphone Android grâce à un port USB.

Présentation de la plateforme Arduino

Diverses Cartes Arduino : Les microcontrôleurs



Lilipad



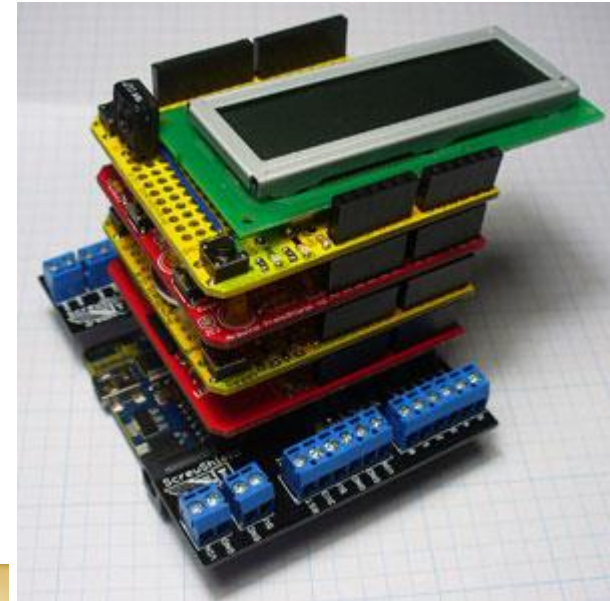
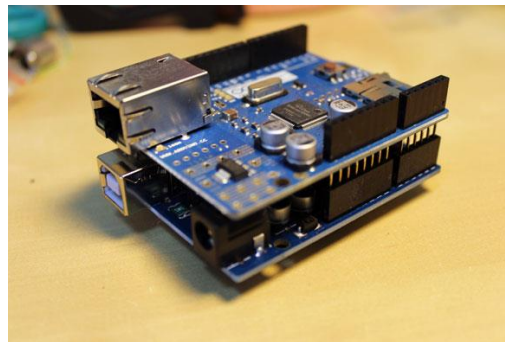
Arduino nano

- ✓ *Permet la miniaturisation,*
- ✓ *utilisées dans des espaces réduits (vêtement).*

Présentation de la plateforme Arduino

❑ Divers Shields Arduino

- ❑ Rajouter des fonctionnalités nouvelles à votre carte Arduino
 - ❑ Relais, commande de moteurs, lecteur carte SD, ...
 - ❑ Ethernet, WIFI, GSM, GPS, ...
 - ❑ Afficheurs LCD, Écran TFT, ...



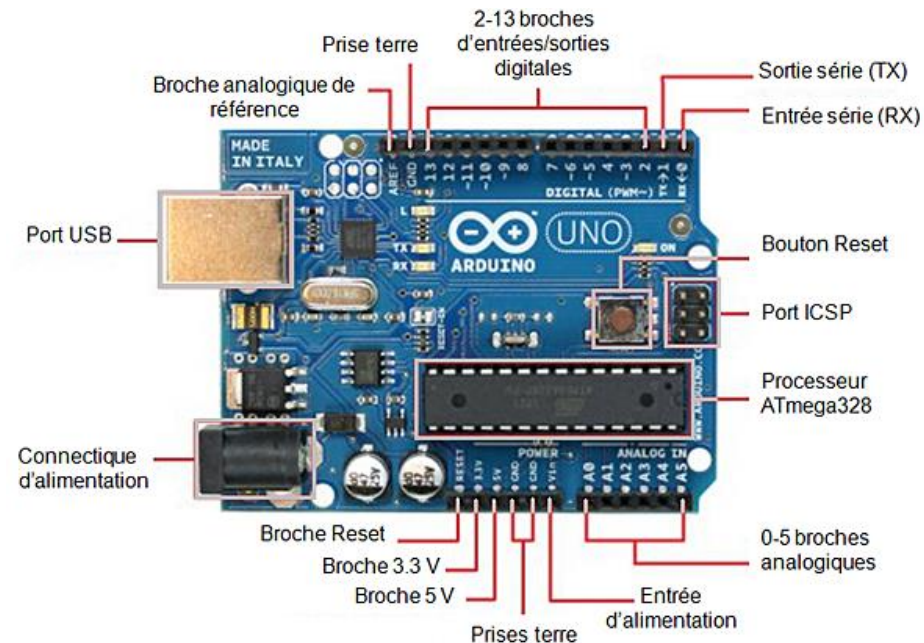
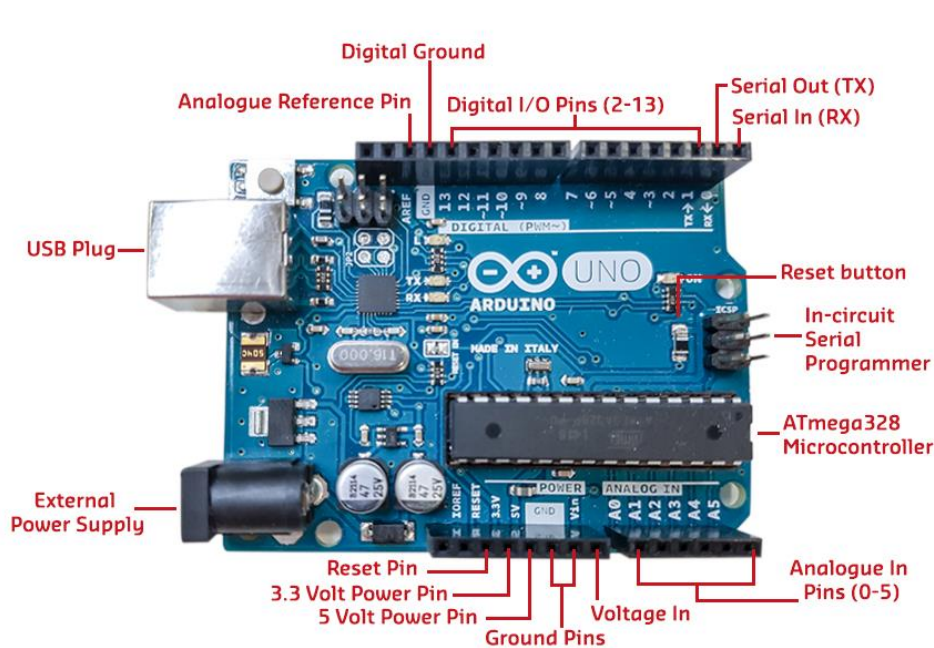
Présentation de la carte Arduino UNO

☐ Les Spécifications Techniques

- ☐ Microcontroller: Microchip ATmega328P (8 bits)
- ☐ Clock Speed: 16 MHz
- ☐ Operating Voltage: 5 Volts
- ☐ Input Voltage: 7 to 20 Volts
- ☐ Digital I/O Pins: 14 (of which 6 can provide PWM output)
- ☐ Analog Input Pins: 6
- ☐ DC Current per I/O Pin: 20 mA
- ☐ DC Current for 3.3V Pin: 50 mA
- ☐ Flash Memory: 32 KB of which 0.5 KB used by bootloader
- ☐ SRAM: 2 KB
- ☐ EEPROM: 1 KB
- ☐ Length: 68.6 mm
- ☐ Width: 53.4 mm
- ☐ Weight: 25 g

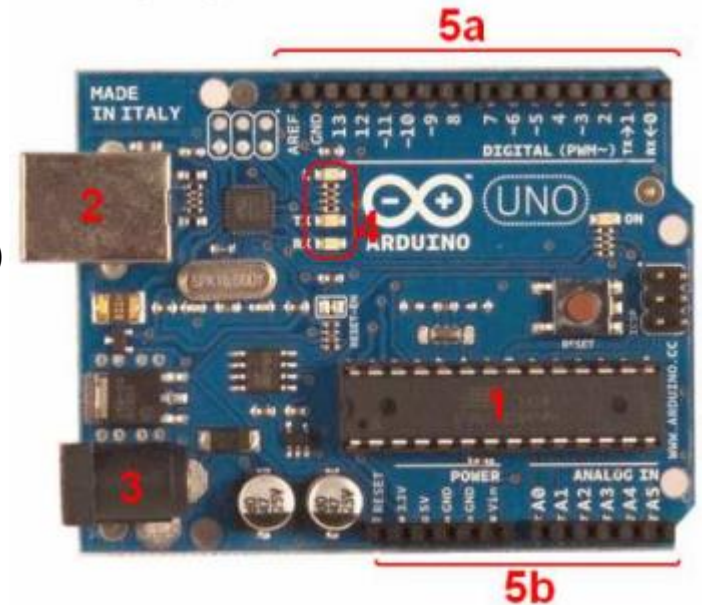
Présentation de la carte Arduino UNO

❑ Les composants principaux de la carte (Pins)



Présentation de la carte Arduino UNO

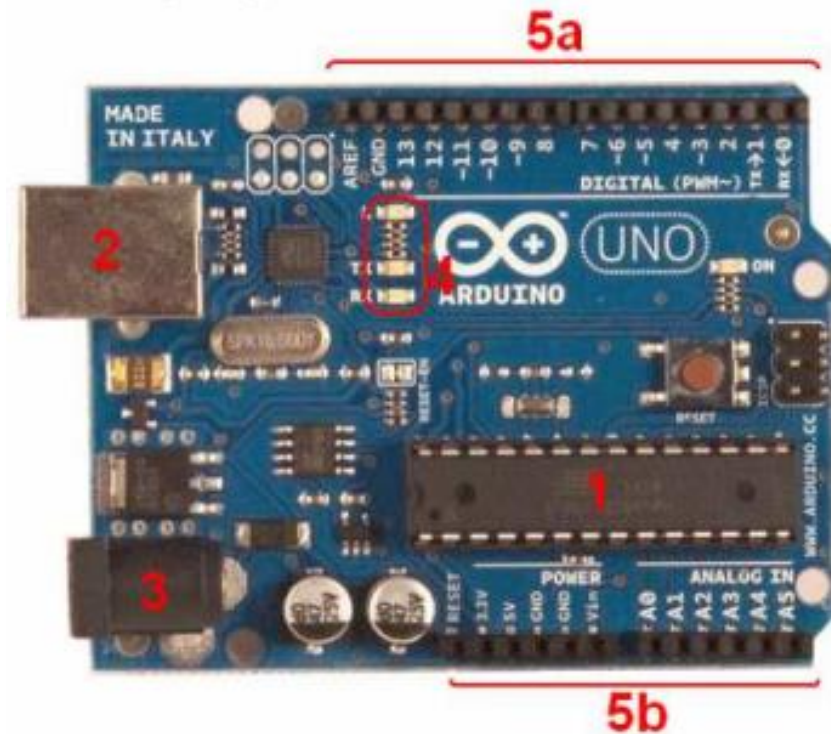
- ❑ **Le microcontrôleur (1)**
- ❑ C'est lui qui va recevoir le programme que vous aurez créé et qui va le stocker dans sa mémoire puis l'exécuter.
- ❑ **L'alimentation (2,3)**
- ❑ la carte peut être alimentée en 5V continue par le port USB (2) ou bien par une alimentation externe (3) qui est comprise entre 7V et 12V. Cette tension peut par exemple être fournie par une pile 9V. Un régulateur se charge ensuite de réduire la tension à 5V.
- ❑ **La visualisation (4)**
- ❑ Les trois "points blancs" entourés en rouge sont en fait des LED. Ces derniers servent à deux choses : elle clignote quelques secondes quand on branche la carte au PC. Les deux LED du bas du cadre : servent au téléchargement du programme dans le microcontrôleur.



Présentation de la carte Arduino UNO

❑ Les connectiques

- ❑ La carte Arduino possède un seul composant (LED connectée à la broche 13) qui peut être utilisés par un programme,
- ❑ Pour rajouter d'autres, il faut les connecter à la carte via les ports 5a et 5b.
- ❑ Par exemple, on veut connecter une LED sur une sortie du microcontrôleur. Il suffit juste de la connecter, avec une résistance en série, à la carte, sur les fiches de connections de la carte.



Présentation de la carte Arduino UNO

- ❑ Les entrées / les sorties digitales
- ❑ Ce sont des connecteurs (D0 à D13) qui permettent la connexion de la carte au monde extérieur.
- ❑ Chacun des connecteurs D0 à D13 peut être configuré dynamiquement par programmation en entrée ou en sortie.
- ❑ Les signaux véhiculés par ces connecteurs sont des signaux logiques compatibles TTL (deux états HAUT (5 Volts) ou BAS (0 Volt)).
- ❑ En pratique, les connecteurs D0 et D1 réservés pour la liaison série asynchrone (port COM virtuel via le câble USB) ne sont pas exploités pour d'autres utilisations.
- ❑ Certains connecteurs peuvent être spécialisés comme sorties PWM (repérées par un ~) .
- ❑ Attention : chacun des connecteurs ne peut fournir ou absorber un courant supérieur à 40 mA



Présentation de la carte Arduino UNO

☐ Les entrées analogiques

- ☐ Par défaut et contrairement aux entrées/sorties digitales qui ne peuvent prendre que deux états HAUT et BAS, ces six entrées peuvent admettre toute tension analogique comprise entre 0 et 5 Volts.

☐ Les bornes d'alimentation

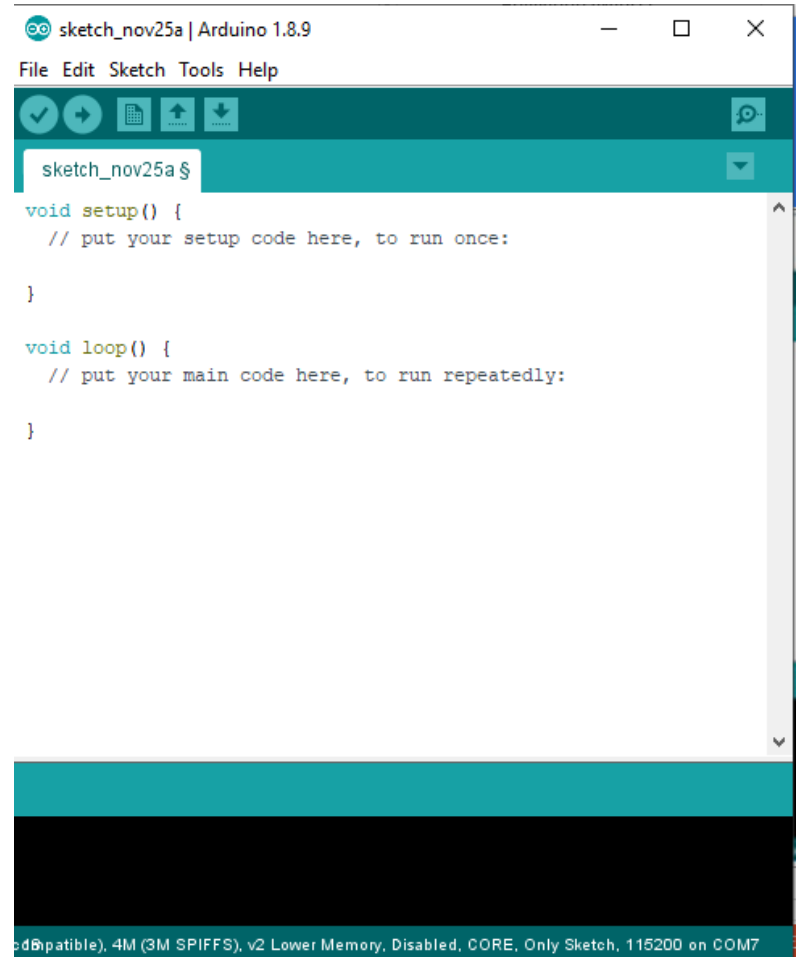
- ☐ GND : Terre (0V)
- ☐ 5V : Alimentation +5V
- ☐ 3.3V : Alimentation +3.3V
- ☐ Vin : Alimentation non stabilisée (= même tension que celle à l'entrée de la carte)
- ☐ IOREF : Cette broche fournit la tension de référence avec laquelle le microcontrôleur fonctionne



L'environnement de programmation d'Arduino

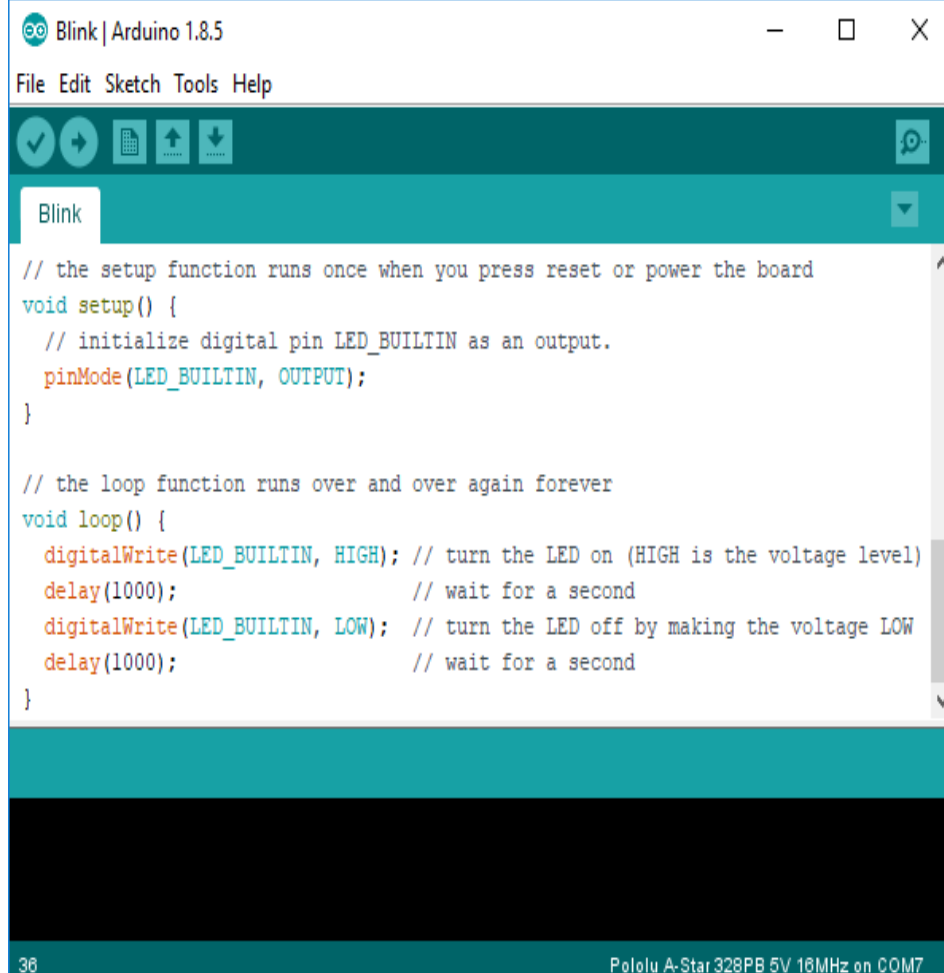
☐ IDE Arduino

- ☐ Environnement de développement intégré fonctionnant sur divers systèmes d'exploitation qui permet d'éditer le programme sur un ordinateur et de le transférer via le port usb.



L'environnement de programmation d'Arduino

- ❑ **Coloration syntaxique de l'IDE**
- ❑ **En orange**, apparaissent les mots-clés reconnus par le langage Arduino comme les fonctions .
- ❑ **En bleu**, apparaissent les mots-clés reconnus par le langage Arduino comme des constantes.
- ❑ **En gris**, les commentaires qui ne seront pas exécutés dans le programme.



The screenshot shows the Arduino IDE interface for the 'Blink' sketch. The title bar reads 'Blink | Arduino 1.8.5'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for saving, running, uploading, and downloading. The sketch name 'Blink' is displayed in a tab. The code editor shows the following code with syntax highlighting: comments in grey, keywords in blue, and function names in orange.

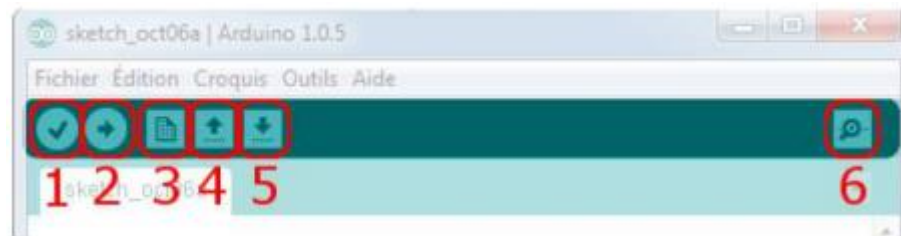
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}







// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

At the bottom of the IDE, the status bar shows the line number '36' and the hardware specification 'Pololu A-Star 328PB 5V 16MHz on COM7'.

L'environnement de programmation d'Arduino

❑ IDE Arduino



1		Verify : Ce bouton permet de vérifier le programme, il actionne un module qui cherche les erreurs dans votre programme
2		Upload : Charge (téléverse) le programme dans la carte Arduino
3		New : pour créer un nouveau programme (sketch)
4		Open : pour ouvrir un programme existant
5		Save : pour sauvegarder le programme
6		Verify : pour visualiser les données transmises par le port série

Programmation de la carte Arduino

- ❑ Pour pouvoir programmer une carte Arduino , il nous faut trois choses :
 - ❑ Un ordinateur avec IDE
 - ❑ Une carte Arduino (avec un câble USB)
 - ❑ Connaitre le **langage Arduino** : langage très proche du C et du C++.

Programmation de la carte Arduino

☐ La structure d'un programme

- ☐ Un programme Arduino est structuré en 3 grandes parties, exécutées successivement:
 - ☐ 1- Déclaration des librairies externes et initialisation des variables et des constantes
 - ☐ 2- Initialisation du programme: la fonction setup()
 - ☐ Fonction est exécutée une seule fois au démarrage du Arduino.
 - ☐ Configuration des entrées/sorties : Les broches configurées en entrée peuvent recevoir des signaux en provenance des capteurs. Les broches configurées en sortie peuvent commander des moteurs ou une LED.
 - ☐ Initialisation des variables
 - ☐ 3- Exécution du programme: la fonction loop()
 - ☐ C'est la fonction principale: c'est ici que s'exécute l'algorithme de travail
 - ☐ Exécuté en continu
 - ☐ Les instructions sont exécutées en boucle jusqu'à l'appui sur « Reset »

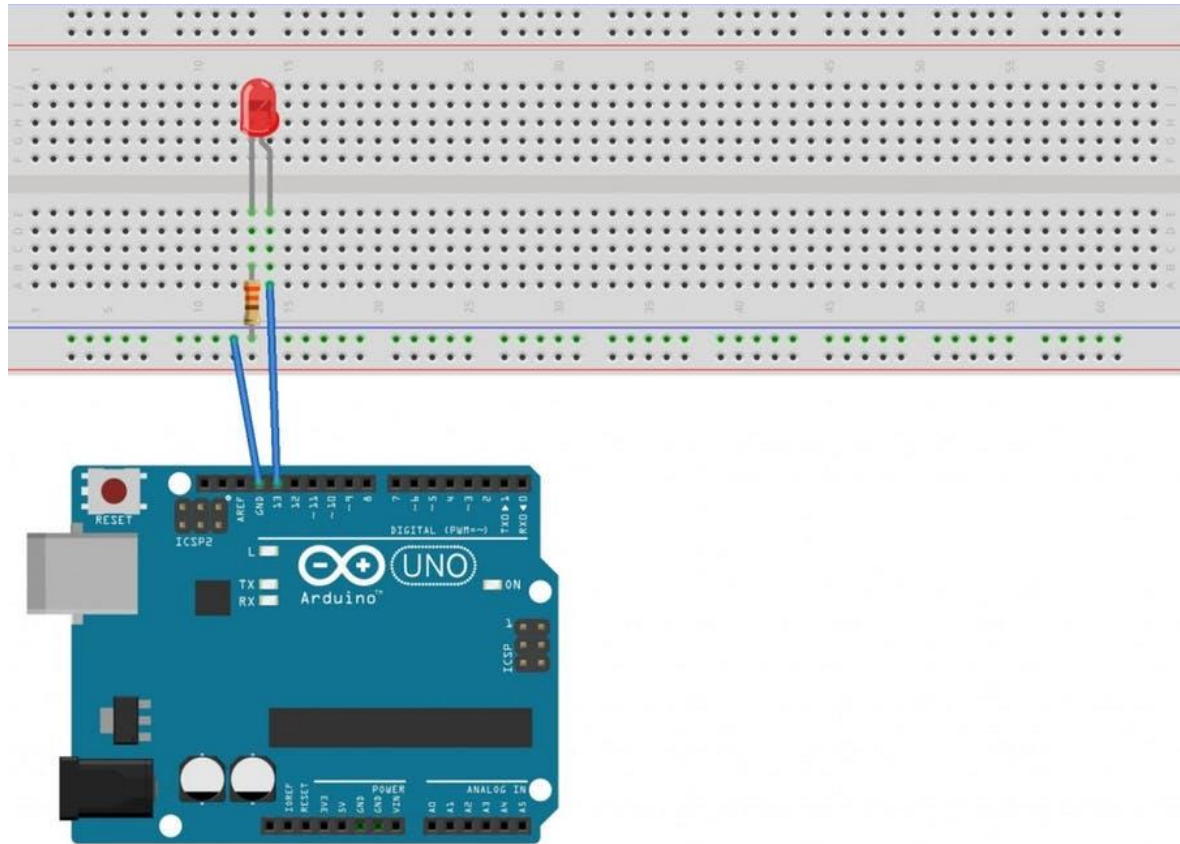
Programmation de la carte Arduino

❑ Premier Programme : Led Blink

```
const int led = 13;
```

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

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



Langage Arduino

☐ La ponctuation

- ☐ Toute ligne de code se termine par un point-virgule « ; »
- ☐ Le contenu d'une fonction est délimité par des accolades « { } »
- ☐ Les paramètres d'une fonction sont contenus entre parenthèses « () »
- ☐ Un commentaire commence par /* et fini avec */ et peut faire plusieurs ligne.
- ☐ Une seule ligne de commentaire commence par // et fini à la fin de la ligne de code.

Langage Arduino

❑ Les types de données

- ❑ **byte** : permet de stocker une valeur entière sur 8 bits (pas de chiffre à virgule). Cela permet d'aller de 0 à 255
- ❑ **int** : est le type de base pour les chiffres (sans virgule) des variables. Le stockage se fait sur 16 bits, permettant d'aller de - 32 768 à 32 767 .
- ❑ **long** : Il s'agit d'un type de variable de grande taille : la donnée (chiffre sans virgule) est stockée sur 32 bits. Cela permet d'aller de - 2 147 483 648 à 2 147 483 647 .
- ❑ **float** : Ce type permet de stocker les chiffre "flottant" ou chiffre à virgule. Le float bénéficie d'une meilleure résolution que les entiers (int) : ils sont stockés sur 32 bits, soit de $3,4028235^{38}$ à $- 3,4028235^{38}$.

Langage Arduino

❑ Les types de données

❑ **bool** : Le type bool prend 2 valeurs : Vrai (true) ou Faux (false). Il est stocké sur 8 bits et se comporte ainsi :

❑ 0 = FAUX = FALSE

❑ 1 = VRAI = TRUE

❑ **char** : Le type char est utilisé pour stocker une lettre. Le stockage se fait sur 8 bits et permet de stocker une seule lettre.

❑ Note : Pour assigner une valeur à la variable char, on l'écrira entre ' '.

❑ **string** : Le type string est comparable à une variable possédant ses propres fonctions. String permet de stocker une chaîne de caractères, comme pourrait le faire un tableau de caractère. String permet de réaliser des opérations sur les chaînes de caractères plus complexes que ce qui se ferait sur un tableau de caractères.

Langage Arduino

❑ Les opérations arithmétique

```
y = y + 3;
```

```
x = x - 7;
```

```
r = r / 5;
```

```
i = j * 6;
```

❑ Les opérations d'incrémentation

```
x ++ //équivalent à x = x + 1 ou d'incrémenter x par 1
```

```
x -- //équivalent à x = x - 1 ou de décrémenter x par 1
```

❑ Les opérations de comparaison

```
x == y //x est égal à y
```

```
x != y //x n'est pas égal à y
```

```
x < y //x est plus petit que y
```

```
x > y //x est plus grand que y
```

```
x <= y //x est plus petit ou égal à y
```

```
x >= y //x est plus grand ou égal à y
```

Langage Arduino

❑ Les opérations logiques

- ❑ sont utilisés pour comparer deux expressions/conditions. Ils renvoient TRUE (vrai) ou FALSE (faux) selon l'opération.

AND

`if (x > 0 && x < 5) //vrai seulement si les deux conditions sont vraies`

OR

`if (x > 0 || y > 0) //vrai si une des condition est vraie`

NOT

`if (!x > 0) //vrai seulement si la condition est fausse`

Langage Arduino

❑ Les constantes

- ❑ Le langage Arduino possède des valeurs prédéfinies. Elles sont utilisées pour rendre un programme plus simple à lire. Il en existe différents groupes.
- ❑ **TRUE / FALSE** : Il s'agit de variable du type booléen. FALSE signifie que la constante est à 0 alors que TRUE désigne l'état différent de 0 (1, -1, 2,).
- ❑ **HIGH/LOW** : Il s'agit des constantes utilisées pour lire ou pour écrire sur les broches, soit le niveau HIGH (ON, 1, 5 volts) soit le niveau LOW (OFF, 0, 0 volts).
- ❑ **INPUT/OUTPUT** : Ces constantes sont utilisées pour définir le type de broche digitale :
 - ❑ INPUT pour entrée
 - ❑ OUTPUT pour sortie

```
if (b == TRUE)
{
.....;
}
```

```
digitalWrite(13,HIGH);
```

```
pinMode(13, OUTPUT);
pinMode(12, INPUT);
```

Langage Arduino

❑ Les structures de contrôle

- ❑ **if** : La structure de test "if" vérifie si une certaine condition est atteinte.
- ❑ **if ... else** : utilisée dans le cas où on veut tester une entrée digitale et faire une action si elle vaut HIGH et d'autres actions si elle vaut LOW.

```
if( x == 10 )  
{  
  Faire quelque chose  
}
```

```
if (inputPin == HIGH)  
{  
  doThingA;  
}  
else  
{  
  doThingB;  
}
```


Langage Arduino

❑ Les structures de contrôle

- ❑ **Switch ... case** : Cette instruction permet de tester si une expression coïncide avec un certain nombre de constantes, et d'exécuter une action par défaut dans le cas où aucune valeur ne correspond à celle de l'expression.
- ❑ La principale différence entre le switch... case et le if... else est que le switch... case permet de continuer les tests même si un des cas est vrai, contrairement au if...else qui quittera dans le même cas.

```
switch (someVariable) {  
  case A : doSomethingA; //si someVariable vaut A, doSomethingA est exécuté  
           break; //une fois le code exécuté, on sort de la structure  
  case B : doSomethingB; //en l'absence de "break", si someVariable est égale à B puis à C  
  case C : doSomethingC; //doSomethingB et doSomethingC seront exécutés  
  default : doSomethingByDefault //dans le cas où aucun cas ne serait vérifié,  
           //doSomethingByDefault sera exécuté puis on sortira de la structure  
}
```

Langage Arduino

❑ Les structures de contrôle

- ❑ **for** : Une boucle for commence par l'initialisation, puis exécute le bloc de code tant que la condition est vérifiée et en appelant l'itération après chaque bloc.

```
for (initialisation; condition; expression d'incrément) {  
doSomething;  
}
```

```
for (int i = 0; i<20; i++) {  
    digitalWrite(13, HIGH);  
    delay(250);  
    digitalWrite(13, LOW);  
    delay(250);  
}
```

Langage Arduino

❑ Les structures de contrôle

- ❑ **while** : Une boucle while exécute le bloc d'instructions tant que la condition est vérifiée.
- ❑ **do ... while** : Cette boucle exécute donc le bloc au moins une fois, et ce jusqu'à ce que la condition soit fausse (ou tant que la condition est vraie).

```
while (someVariable < 200) //test si "someVariable" est inférieur à 200
{
doSomething; //exécution du code
someVariable++ //incrémenter de "someVariable" par 1
}
```

```
do {
  x = lireCapteur(); //assignation de la valeur de lireCapteur à x
  delay(50); //pause durant 50 milli-secondes
} while (x < 100); //on recommence la boucle si x est inférieur à 100
```

Langage Arduino

❑ Les entrées/sorties numériques

- ❑ **pinMode(pin, mode)** : Utilisée dans le setup(), cette fonction permet de configurer une broche soit en entrée (INPUT) soit en sortie (OUTPUT).

```
pinMode(broche, OUTPUT); //fixe la broche en sortie
```

- ❑ **digitalRead(pin)** : Cette fonction permet de lire la valeur à la broche digitale indiquée. La valeur lue étant soit HIGH soit LOW. La broche ciblée ne peut être qu'une variable ou une constante comprise entre 0 et 13 (pour les Arduino Uno)

```
valeur = digitalRead(broche); //assigne à "valeur" la valeur lue à la broche
```

Langage Arduino

❑ Les entrées/sorties numériques

- ❑ **digitalWrite(pin, value)** : Il s'agit du contraire de digitalRead() : cette fonction permet de fixer une sortie digitale soit à HIGH soit à LOW. Cette sortie est comprise entre 0 et 13 sur Arduino Uno et doit être spécifiée soit par une variable soit par une constante.

```
digitalWrite(broche, HIGH) //fixe la sortie "broche" à HIGH
```

Langage Arduino

☐ Les entrées/sorties analogiques

- ☐ Les cartes Arduino disposent tous d'un convertisseur A/N dont le nombre d'entrée varie entre 6 et 16 (selon le type de la carte).
- ☐ Ce convertisseur travaille sur 10 bits donc la donnée numérique en sortie est comprise entre 0 et 1023.
- ☐ La résolution dépend de la tension de référence.
- ☐ Le compilateur propose deux fonctions pour gérer ce convertisseur
 - ☐ Une pour la lecture de la donnée convertie
 - ☐ Une pour définir la tension de référence
- ☐ **analogReference(type)** : Permet de préciser la tension de référence utilisée par le CAN. *type* ne peut prendre que les valeurs suivantes:
 - ☐ DEFAULT (5V); INTERNAL(1,1V); INTERNAL2V56(2,56V); EXTERNAL(tension de l'entrée externe)

Langage Arduino

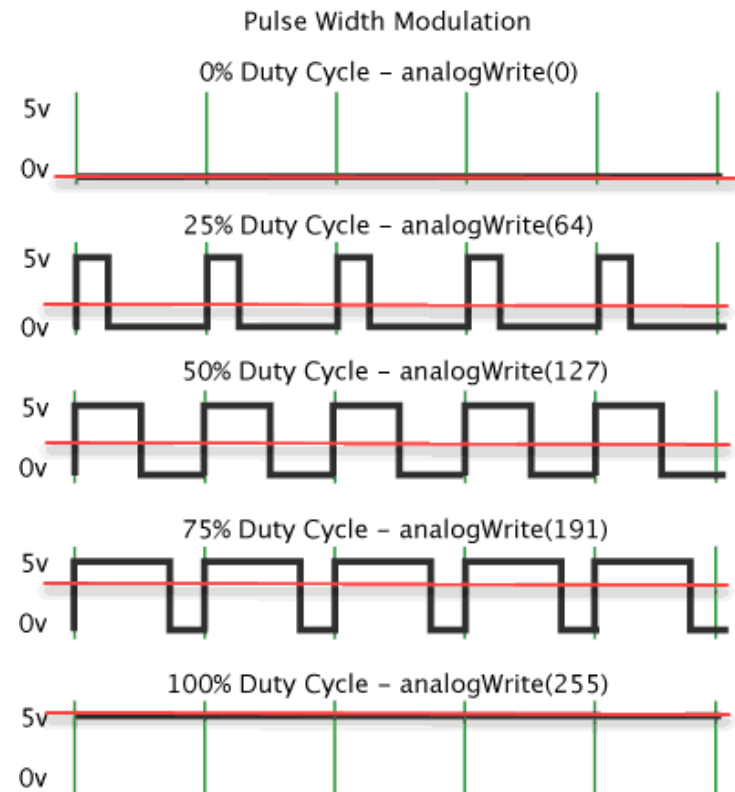
❑ Les entrées/sorties analogiques

- ❑ **`analogRead(broche)`** : Cette fonction permet d'aller lire la valeur sur une sortie analogique avec 10 bits de résolution. Elle ne fonctionne que sur les broches analogiques (0-5 ou 0-15 selon la carte) et renvoie un résultat compris entre 0 et 1023.
- ❑ Les broches analogiques ne sont pas des broches digitales, elle ne nécessite pas d'être déclarée comme des sorties ou des entrées.

Langage Arduino

❑ Les entrées/sorties analogiques

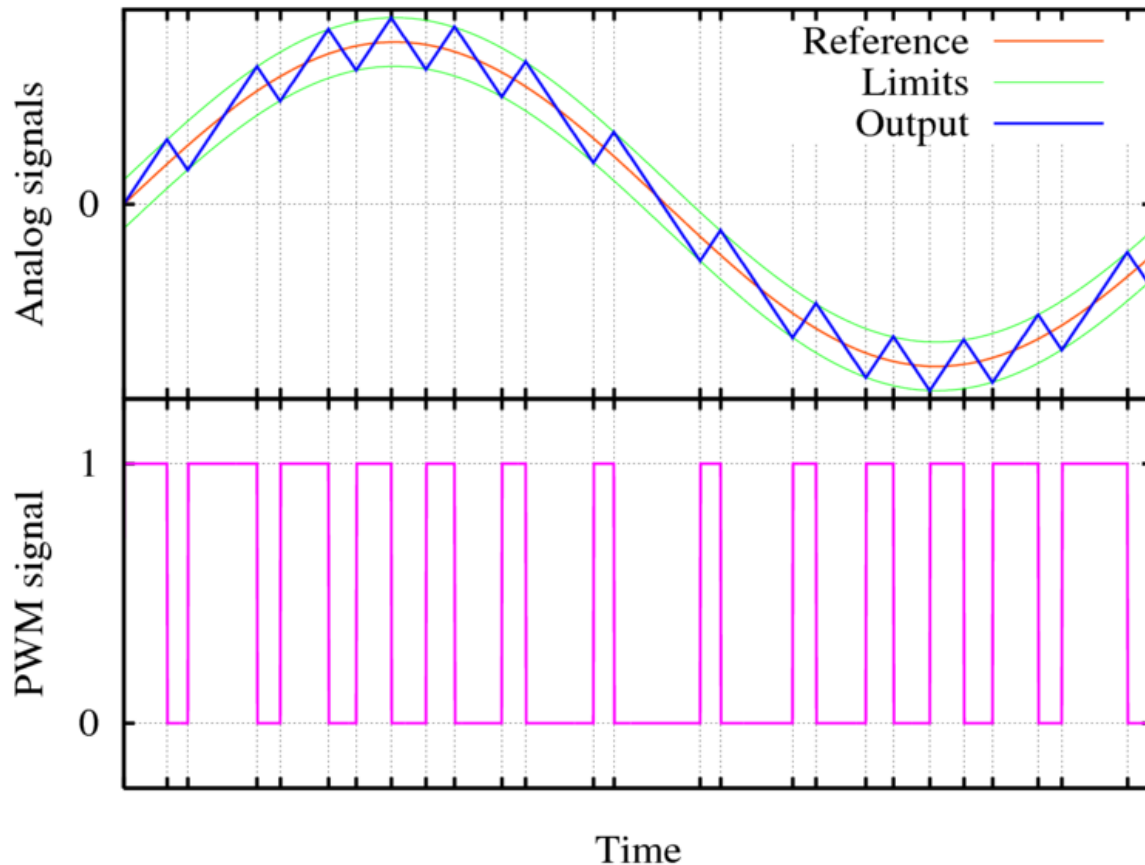
- ❑ Les cartes arduino ne disposent pas de CNA au sens strict du terme.
- ❑ Par contre il sont capables de générer des signaux à modulation de largeur d'impulsion (PWM : Pulse Width Modulation).
- ❑ Ce sont des signaux rectangulaires (490 Hz pour arduino) et dont le rapport cyclique est variable entre 0 et 100% .



Langage Arduino

❑ Les entrées/sorties analogiques

❑ Exemple de signal PWM :



Langage Arduino

❑ Les entrées/sorties analogiques

- ❑ `analogWrite(sortiePWM, rapportcyclique)` : permet d'envoyer un pseudo-signal analogique en utilisant les broches possédant un "pulse width modulation" (PWM).
- ❑ Cette fonctionnalité se trouve sur les broches 3, 5, 6, 9, 10 et 11 de la carte UNO.
- ❑ `rapportcyclique` : variable de type byte (au minimum) compris ici entre 0 et 255.
- ❑ Après avoir appelé l'instruction `analogWrite()`, la broche générera une onde carrée stable avec un "duty cycle" de longueur spécifiée (en largeur 0-255), jusqu'à l'appel suivant de l'instruction `analogWrite()`, `digitalRead()`, ou `digitalWrite()`.

Langage Arduino

❑ La Gestion du temps

- ❑ Les fonctions de gestion de temps sont utilisés pour une variété de tâches, par exemple, pour gérer le temps d'appui sur un bouton ou pour générer des signaux modulés en largeur d'impulsion (PWM) pour le contrôle moteur à titre d'exemple. Le langage Arduino fournis quelques fonctions permettant de gérer le temps:
- ❑ La fonction `delay()`: suspend le programme durant un laps de temps (en millisecondes) spécifié comme paramètre.
- ❑ La fonction `delayMicroseconds()`: suspend le programme pour un laps de temps (en microsecondes) spécifié comme paramètre.
- ❑ La fonction `millis()`: retourne le nombre de millisecondes depuis que la carte Arduino a commencé à exécuter le programme jusqu'au moment de l'exécution de la fonction `millis()`.
- ❑ La fonction `micros()`: retourne le nombre de microsecondes depuis que la carte Arduino a commencé l'exécution du programme en cours.

Langage Arduino

☐ La Gestion des interruptions

- ☐ Moyen performant pour surveiller un évènement extérieur.
- ☐ Exp : Application pendant laquelle l'appui sur un poussoir doit provoquer un changement de comportement.
 - ☐ En programmation classique, on va devoir faire plusieurs tests de ce poussoir sans être certain que l'on manquera pas un changement de comportement.
 - ☐ Avec les interruptions, il faut câbler le poussoir de façon à ce qu'il agisse sur une entrée d'interruption. Dans ce cas il n'y aura plus besoin d'aucun test dans le programme et on ne risquera pas de manquer un appui sur le poussoir, car ce dernier provoquera toujours une interruption.
 - ☐ Il y a deux broches d'interruption externes sur l'Arduino UNO, INT0 et INT1, et ils sont mappés sur les broches 2 et 3 de la carte.

Langage Arduino

❑ La Gestion des interruptions

- ❑ La fonction `attachInterrupt()`: détermine une routine d'interruption à exécuter quand une interruption se produit.
- ❑ Syntaxe de la fonction: `attachInterrupt(entrée, routine, mode)`
 - ❑ `entrée`: Numéro de l'interruption (0 (pin2) ou 1 (pin3))
 - ❑ `routine`: fonction à appeler une fois que l'interruption se produit. Cette fonction ne doit pas prendre de paramètres et ne retourne rien.
 - ❑ `mode`: LOW, CHANGE, RISING, ou FALLING.
 - ❑ `LOW`: Lorsque la broche est à l'état 0 (0V).
 - ❑ `RISING`: Lorsque la broche passe de l'état 0 (0V) à l'état 1 (5V) (front montant).
 - ❑ `FALLING`: Lorsque la broche passe de l'état 1 (5V) à l'état 0 (0V) (front descendant).
 - ❑ `CHANGE`: Lorsque la broche change d'état (front montant et front descendant).
- ❑ La fonction `detachInterrupt()`: Désactive une interruption.
- ❑ Syntaxe de la fonction: `detachInterrupt(entrée)`.

Langage Arduino

☐ La Gestion des interruptions

☐ Tout microcontrôleur qui se respecte offre la possibilité d'interdire ou autoriser les interruptions.

☐ Exp: programme dont le timing doit être précisément respecté et qui, de ce fait, ne doit pas pouvoir être interrompu.

☐ Dans Arduino, les interruptions sont autorisées par défaut, car ils sont utilisées par certaines fonctions (`delay()`, `Serial()` par exemples). Pour cela deux fonctions sont disponibles.

☐ La fonction `noInterrupts()`: interdire les interruptions

☐ La fonction `interrupts()`: autoriser les interruptions

Langage Arduino

☐ La Gestion du port Série

- ☐ Utilisée pour la communication entre la carte Arduino et un ordinateur ou d'autres appareils.
- ☐ Toutes les cartes Arduino ont au moins un port série. Il communique sur les broches numériques 0 (Rx) et 1 (TX) et avec l'ordinateur via USB.
- ☐ Le port série de l'Arduino Uno s'appelle **Serial**
- ☐ Serial fait appel à un certain nombre de fonctions :
 - ☐ begin(), ...

Langage Arduino

❑ La Gestion du port Série

- ❑ **Serial.begin(débit)** : cette fonction doit être appelée au moins une fois dans le setup().
- ❑ Permet d'activer le port série et de définir la vitesse de transmission (débit, en bits par seconde ou bauds).
- ❑ La vitesse usuelle est de 9600 bauds avec les ordinateurs mais d'autres vitesses sont supportées.

```
void setup() {  
  Serial.begin(9600); //Ouvre le port série et fixe le débit à 9600 bauds  
}
```

- ❑ **Note :** Lorsque la liaison série est utilisée, les broches digitales 0 (RX) et 1 (TX) ne peuvent être utilisées en même temps.

Langage Arduino

❑ La Gestion du port Série

- ❑ **Serial.println(données)** : cette fonction Permet d'envoyer des données sur le port série et d'automatiquement retourner à la ligne ensuite. La commande Serial.print() fonctionne de la même façon mais sans le retour à la ligne, pouvant être moins facile à lire via le moniteur série.

```
void setup() {  
  Serial.begin(9600); //affecte le débit à 9600 bauds  
}  
  
void loop() {  
  Serial.println(analogRead(0)); //envoi la valeur analogique  
  delay(1000); //marque une pause d'une seconde  
}
```

Langage Arduino

❑ La Gestion du port Série

- ❑ le port série des arduino dispose d'une mémoire tampon capable de mémoriser jusqu'à 128 octets.
- ❑ `Serial.available()` : permet d'obtenir le nombre d'octets disponibles pour lecture sur port série (disponible dans la mémoire tampon).
- ❑ Le retour de cette fonction est de type int.

Langage Arduino

❑ La Gestion du port Série

- ❑ `Serial.read()` : permet de lire le premier caractère disponible dans la mémoire tampon de réception et donc le premier caractère à avoir été reçu.

Langage Arduino

☐ Bibliothèques Arduino

- ☐ Comme Arduino est un projet communautaire, vous trouverez des bibliothèques sur le web en général.
- ☐ Quelques bibliothèques sont recensées sur le site officiel .
- ☐ les bibliothèques officielles sont installées dans le répertoire :
 - ☐ sous Windows 32 bits, dans C :\Program Files\Arduino\libraries ;
 - ☐ sous Windows 64 bits, dans C :\Program Files(x86)\Arduino\libraries ;
- ☐ Le dossier : C:\Users\nom du PC\Documents\Arduino\libraries est réservé pour les autres librairies (non officielles).
- ☐ Pour les installer, vous avez le menu Sketch/Include Library /add library .ZIP...
- ☐ L'IDE doit être redémarré.

Cartes ESP

Carte ESP8266

- ❑ L'ESP8266 est un circuit intégré à microcontrôleur par le fabricant chinois Espressif.
- ❑ L'ESP8266 d'Espressif Systems n'est autre qu'un microcontrôleur à cœur Tensilica Xtensa LX106 (processeur RISC 32 bits) complété de fonctionnalités Wifi associées à une mémoire flash contenant le firmware.
- ❑ En gros, c'est une carte Arduino connecté en WIFI.

Carte ESP8266

- ☐ L'ESP8266 possède autant de versions que de clones.



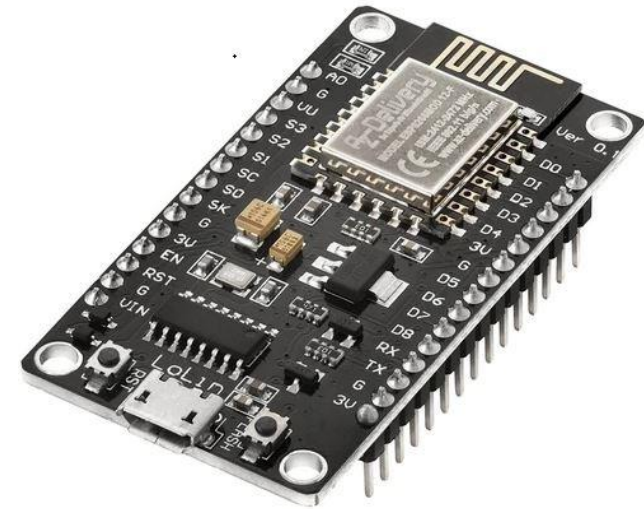
Carte ESP8266

- ❑ **L'ESP8266 peut se programmer de plusieurs façons :**
 - ❑ En Lua, interprétés ou compilés, avec le firmware NodeMCU (langage par défaut) ;
 - ❑ En C, avec le SDK esp-open-sdk basé sur la chaîne de compilation GCC ou avec le SDK d'Espressif ;
 - ❑ En C++, avec l'IDE Arduino ;
 - ❑ En Go, avec le framework Gobot ;
 - ❑ En JavaScript, avec le firmware Espruino ;
 - ❑ En MicroBlocks, notamment sur NodeMCU ;
 - ❑ En MicroPython, avec le firmware MicroPython

Carte ESP8266 NodeMCU

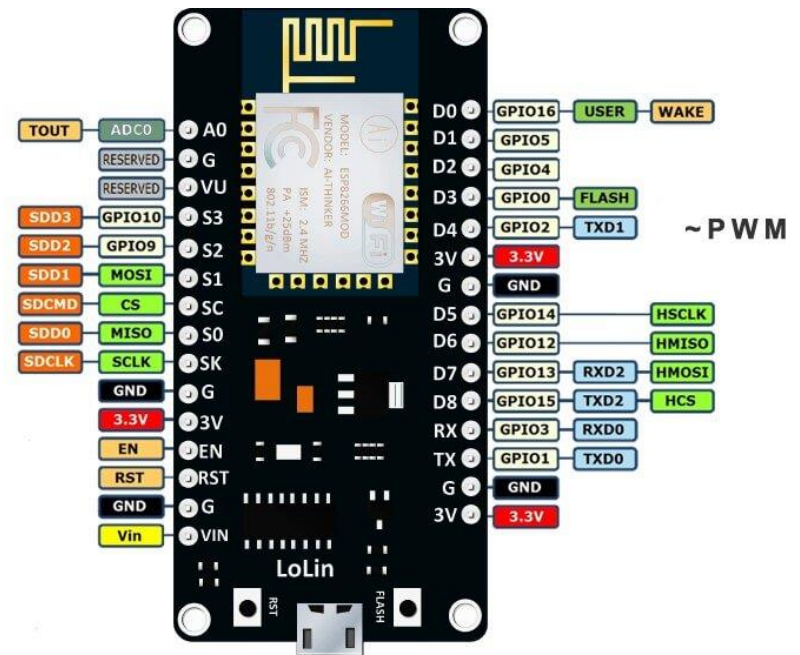
❑ Caractéristiques

Nom:	ESP8266
Marque:	Espressif
Caractéristiques	
CPU:	Tensilica 32-bit RISC CPU Xtensa LX106
Tension d'alimentation :	7-12V
Tension logic:	3.3V
E/S digitales:	16
Entrées analogiques:	1
Flash:	4000kB
SRAM:	64kB
EEPROM:	NCKB
Fréquence d'horloge:	80 MHz
Wifi:	Yes
Bluetooth:	No
SD card:	No
Touch:	Yes
UART/SPI/I2C/I2S:	Yes/Yes/Yes/Yes



Carte ESP8266 NodeMCU

- ❑ Analog I/O : 1 (A0)
- ❑ Digital I/O : 5 (D0, D1, D3, D4, D7)
- ❑ Broches PWM : 4 (D2, D5, D6, D8)
- ❑ Communication Serial: 10 (D0, D1, D2, D3, D4, D5, D12, D13, D14, D15)
- ❑ Communication I2C : 1 (('D1', 'D2'))
- ❑ Communication SPI : 1 (('D8', 'D5', 'D6', 'D7'))
- ❑ Communication I2S: 1 (('D15', 'D2', 'D3'))
- ❑ Interrupt : 6 (D1, D2, D5, D6, D7, D8)



Carte ESP32 NodeMCU

- ☐ Successeur de la carte ESP8266
- ☐ Il présente plusieurs intérêts :
- ☐ Prix faible ~5€ pour le module ~8€ pour une petite carte de développement
- ☐ Intègre du Wifi 802.11 b/g/n/e/i (WPA)/WPA2/WPA2-Enterprise/Wi-Fi Protected Setup (WPS)
- ☐ Intègre Bluetooth 4.2 - BLE Bluetooth low Energy
- ☐ Compatible avec l'environnement de développement ARDUINO
- ☐ Intègre un microcontrôleur 32 bits performants et de nombreux périphériques (ADC 12bit, DAC, 3xUART, PWM, I2C, SPI, etc ...)
- ☐ S'alimente directement en USB
- ☐ etc.



Carte ESP32 NodeMCU

❑ Caractéristiques de la Carte ESP32 :

CPU:	ESP-WROOM-32 (Tensilica Xtensa LX6)
Tension d'alimentation :	7-12V
Tension logic:	3.3V
E/S digitales:	14
Entrées analogiques:	6
Flash:	4000kB
SRAM:	520kB
EEPROM:	448kB
Fréquence d'horloge:	240 MHz
Wifi:	Yes
Bluetooth:	Yes
SD card:	No
Touch:	Yes
UART/SPI/I2C/I2S:	Yes/Yes/Yes/Yes

Raspberry Pi

Présentation de la carte RBPi

- ❑ Le Raspberry Pi est un Micro-ordinateur peu coûteux.
- ❑ Micro-ordinateur à processeur ARM conçu par David Braben (UK) 2011-2012.
- ❑ Elle a la taille d'une carte de crédit, mais elle est capable de gérer un système d'exploitation complet et de faire tout ce que fait un ordinateur de bureau.



Présentation de la carte RBPi

- ❑ Destiné à encourager l'apprentissage de la programmation informatique, et d'explorer des projets électroniques et IoT.
- ❑ Plus de 20 millions d'exemplaires vendus
- ❑ disponible à moins de 45 € (entre 10 et 40 €)
- ❑ Il est généralement vendu "nu" (carte mère seule, sans boîtier, alimentation, clavier, souris ni écran) dans l'objectif de diminuer les coûts et de permettre l'utilisation de matériel de récupération.



Présentation de la carte RBPi

- ❑ Plusieurs générations de Raspberry Pi ont été produites.
- ❑ Tous les modèles disposent d'un système SoC (System on a Chip) Broadcom avec un CPU et GPU d'architecture ARM.
- ❑ Les architectures ARM sont des architectures de type RISC 32 bits (ARMv1 à ARMv7) et 64 bits (ARMv8) développées par ARM Ltd depuis 1990.



Présentation de la carte RBPi

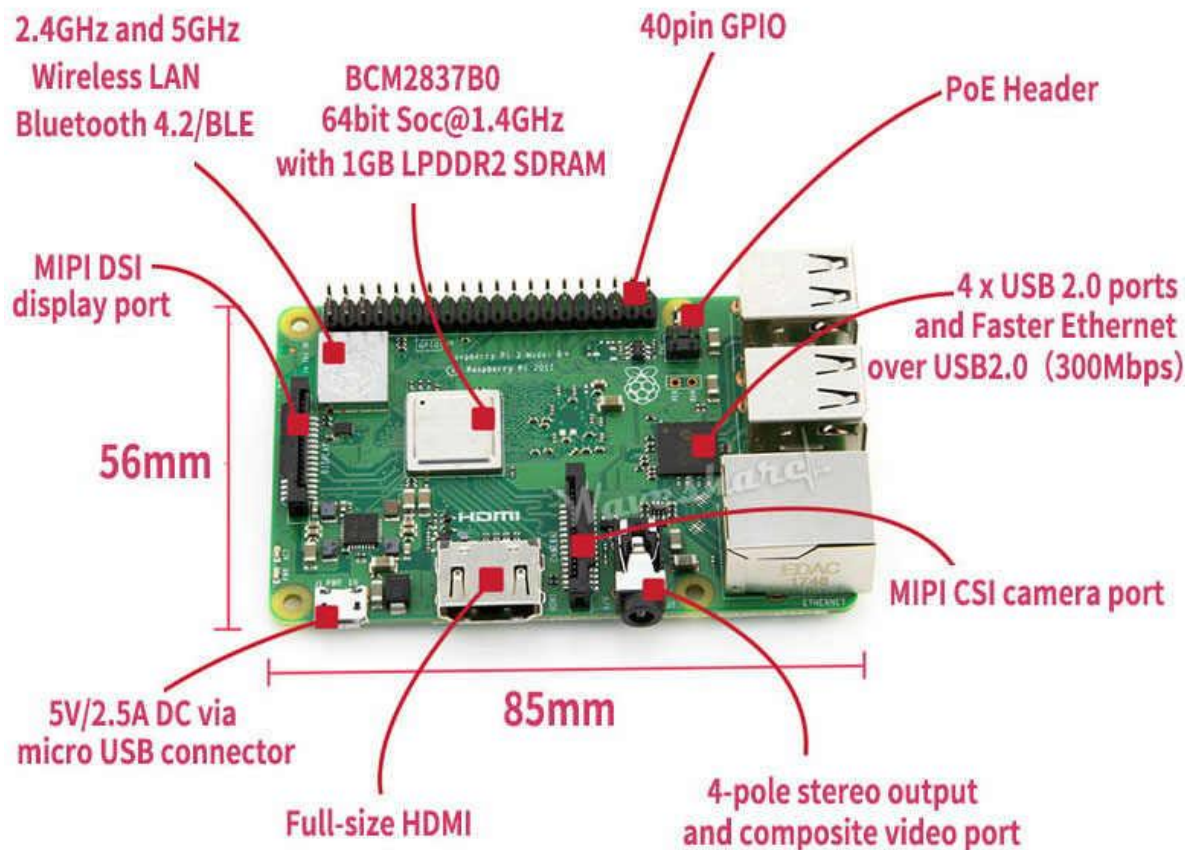
☐ Caractéristiques :

- ☐ Processeur ARM
- ☐ RAM
- ☐ Sorties vidéo composite, HDMI et Micro HDMI (suivant les modèles)
- ☐ USB, Ethernet, Wifi et Bluetooth (suivant les modèles)
- ☐ GPIO, UART, SPI, I2C
- ☐ Audio
- ☐ DSI/CSI (Display/Camera Serial Interface)













Présentation de la carte RBPi 3 B+

❑ Caractéristiques :



Versions du Raspberry Pi

	Raspberry Pi 4 B	Raspberry Pi 3 Model A+	Raspberry Pi 3 B+	Raspberry Pi Zero WH	Raspberry Pi Zero W	Raspberry Pi 3	Raspberry Pi Zero	Raspberry Pi 2	Raspberry Pi A+	Raspberry Pi B
										
Date de sortie	24 juin 2019	15 nov. 2018	14 mars 2018	12 janv. 2018	28 févr. 2017	29 févr. 2016	30 nov. 2015	1 févr. 2015	10 nov. 2014	15 févr. 2012
Description				connecteur GPIO déjà soudé						
Caractéristiques										
Prix	37,95 €	25,00 \$US	39,59 €	14,60 €	10,44 €	35,00 \$US	5,00 \$US	35,00 \$US	35,00 \$US	35,00 \$US
SOC										
SOC Type	Broadcom BCM2711	Broadcom BCM2837B0	Broadcom BCM2837B0	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2837	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2835	Broadcom BCM2835
Core Type	Cortex-A72 (ARM v8) 64-bit	Cortex-A53 64-bit	Cortex-A53 64-bit	ARM1176JZF-S	ARM1176JZF-S	Cortex-A53 64-bit	ARM1176JZF-S	Cortex-A7	ARM1176JZF-S	ARM1176JZF-S
Nombre de coeurs	4	4	4	1	1	4	1	4	1	1
GPU	VideoCore VI	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV 1080p@30	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV 1080p@30
Fréquence processeur	1,5 GHz	1,4 GHz	1,4 GHz	1 GHz	1 GHz	1,2 GHz	1 GHz	900 MHz	700 MHz	700 MHz
Mémoire vive	1 GB , 2 GB, 4 GB	512 MB DDR2	1 GB DDR2	512 MB	512 MB	1 GB DDR2	512 MB	1 GB	256 MB	512 MB
Wired Connectivity										
USB	2x USB3.0 + 2x USB2.0	1xUSB 2.0	4x USB2.0	micro & micro OTG	1 x micro OTG	4x USB2.0 + micro OTG	micro + micro OTG	4 + OTG	1	2x USB 2.0
Ethernet	Gigabit	✗	Gigabit - Over USB 2.0	✗ module supp (14.89€)	✗ module supp (14.89€)	10/100M	✗	10/100M	✗	✗
Port HDMI	2x micro HDMI	✓	HDMI	mini	mini	✓	mini	✓	✓	✓
Analog Video Out	shared with audio jack	shared with audio jack	shared with audio jack	via unpopulated pin	via unpopulated pin	shared with audio jack	via unpopulated pin	shared with audio jack	shared with audio jack	Composite video
Analog Audio Out	3.5mm jack	3.5mm jack	3.5mm jack	HDMI audio	HDMI audio	3.5mm jack	HDMI audio	3.5mm jack	3.5mm jack	3.5mm jack
I2C	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GPIO	✓	✓	✓	✓	✓	40-pin	✓	✓	✓	26-pins
LCD Panel	✓	✓	✓	✗	✗	✓	✗	✓	✓	DSI
Camera	✓	✓	✓	✓	✓	✓	latest version include a camera connector	✓	✓	DSI
SD/MMC	microSD	microSD	microSD	microSD	microSD	microSD	microSD	microSD	microSD	SD, SDHC and SDXC up to 2TB
Wireless Connectivity (On-Board)										
Wi-Fi	2.4GHz and 5GHz 802.11 b/g/n/ac	2.4GHz and 5GHz 802.11 b/g/n/ac	2.4GHz and 5GHz 802.11 b/g/n/ac	802.11n	802.11n	802.11n	✗	✗	✗	✗
Bluetooth®	5.0	4.2. BLE	4.2. BLE	4.1	4.1	4.1 LE	✗	✗	✗	✗

Option d'affichage Raspberry Pi

- ❑ Il existe trois options de connexion pour l'affichage :
 - ❑ HDMI (haute définition) : Les téléviseurs HD et LCD peuvent être connectés à l'aide d'un câble HDMI "mâle" standard ou d'un adaptateur DVI ou VGA. L'audio et la vidéo sont transmis via HDMI, mais les entrées HDMI ne sont pas supportées.
 - ❑ Composite (définition standard) : Les téléviseurs plus anciens peuvent être connectés à l'aide de la vidéo composite (câble RCA jaune). L'audio est disponible à partir de la prise jack 3,5 mm et peut être utilisé avec un téléviseur, des écouteurs ou un amplificateur.
 - ❑ DSI (Display Serial Interface) :
<https://www.raspberrypi.org/products/raspberry-pi-touch-display/>



Systemes d'exploitation pour Raspberry Pi

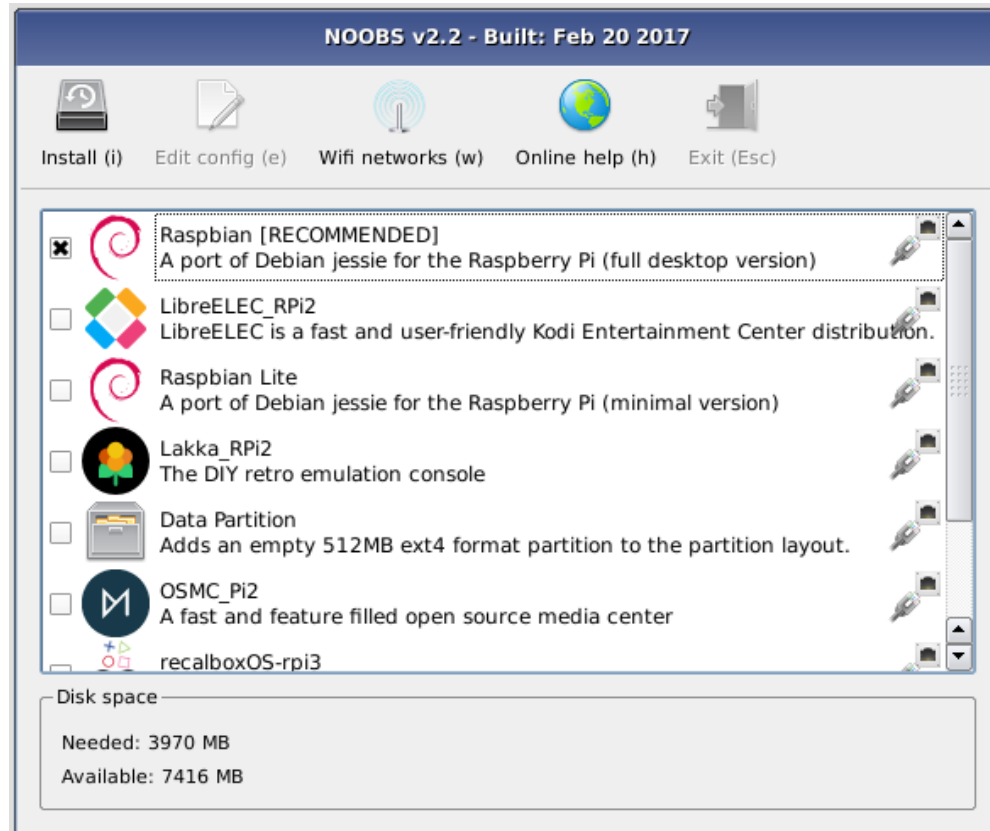
- ❑ Depuis sa commercialisation, la Raspberry Pi exécute des variantes du système d'exploitation libre GNU/Linux-Debian.
- ❑ Suivant les modèles, on pourra aussi exécuter :
 - ❑ Plusieurs variantes du système d'exploitation libre GNU/Linux (Raspbian est recommandé par la fondation raspberry pi)
 - ❑ Windows 10 IoT Core
(developer.microsoft.com/en-us/windows/iot/downloads)
 - ❑ Android Pi (androidpi.wikia.com/wiki/Android_Pi_Wiki)

Premiers pas Raspberry Pi

- ❑ **le Raspberry Pi a été conçu pour être aussi simple et rapide à configurer et à utiliser mais, comme tout ordinateur, il repose sur divers périphériques externes :**
 - ❑ **une alimentation USB** - Une alimentation nominale à 2,5 ampères (2,5A) ou 12,5 watts (12,5 W) et avec un connecteur micro USB.
 - ❑ **Carte microSD** - La carte microSD sert de stockage permanent au Raspberry Pi; tous les fichiers que vous créez et les logiciels que vous installez, avec l'OS, sont stockés sur la carte microSD (Taille recommandée est > 16 Go).
 - ❑ **Clavier et souris USB** - Le clavier et la souris vous permettent de contrôler le Raspberry Pi. Presque tous les claviers et souris filaires ou sans fil dotés d'un connecteur USB fonctionnent avec le Raspberry Pi.
 - ❑ **Câble HDMI** - Le câble HDMI transmet le son et des images du Raspberry Pi sur votre téléviseur ou écran . Si vous utilisez un écran d'ordinateur sans prise HDMI, vous pouvez acheter des adaptateurs HDMI vers DVI-D, ou VGA.
 - ❑ **Boîtier** : Pour la protection de la carte
 - ❑ **Câble réseau** : Pour la connexion filaire

Premiers pas Raspberry Pi

- ❑ Préparation et Configuration du système d'exploitation.
- ❑ Deux méthodes d'installation de **Raspbian** (SE recommandé) :
 - ❑ Utiliser NOOBS
 - ❑ New Out-Of-Box Software, outil conçu pour simplifier au maximum la tâche en vous permettant de choisir parmi plusieurs systèmes d'exploitation et de les installer automatiquement. Mieux encore, vous pouvez faire tout cela en quelques clics de la souris.



Premiers pas Raspberry Pi

- ❑ **Préparation et Configuration du système d'exploitation.**
 - ❑ Deux méthodes d'installation de **Raspbian** (SE recommandé) :
 - ❑ Installer Raspbian avec la méthode classique
 - ❑ Etapes :
 - ❑ 1. Télécharger directement une version de Raspbian
(<https://www.raspberrypi.org/downloads/>)
 - ❑ 2. installer un outil permettant de créer une carte SD bootable
 - ❑ 3. insérer la carte SD dans le lecteur.
 - ❑ 4. copier Raspbian dans la carte SD.
 - ❑ 5. insérer la carte dans le port SD de la carte Raspberry PI

Premiers pas Raspberry Pi

☐ Configuration du matériel

☐ Brancher :

- ☐ Le clavier
- ☐ La souris
- ☐ Le câble HDMI
- ☐ La câble réseau
- ☐ Câble d'alimentation

- ☐ Le processus d'installation commence (Le RPi démarre dès qu'il est branché sur l'alimentation USB)

Premiers pas Raspberry Pi

☐ Paramétrage du système

- ☐ Sélectionner le pays, la langue et le fuseau horaire
- ☐ Changer le mot de passe par défaut
- ☐ Sélectionner un réseau Wifi (s'il existe)
- ☐ lancer le « Raspberry PI configuration tool » :

Premiers pas Raspberry Pi

☐ Paramétrage du système

- ☐ Raspberry PI configuration tool :

- ☐ Situé dans Préférences du menu du bureau, il vous permet de configurer les paramètres matériel et logiciel de votre Raspberry Pi.

- ☐ Accessible via la commande : `sudo raspi-config`

- ☐ L'outil de configuration de Raspberry Pi affiche quatre onglets:

- ☐ **Système**. Les options permettant d'étendre le système de fichiers et de changer le mot de passe et le nom d'hôte sont associées à divers choix de connexion.

- ☐ **Interfaces** Prise en charge des diverses fonctionnalités matérielles et logicielles, telles que Camera Module, SSH et VNC.

- ☐ **Performance**. Les options de mémoire Overclocking et GPU peuvent améliorer les performances d'un Raspberry Pi.

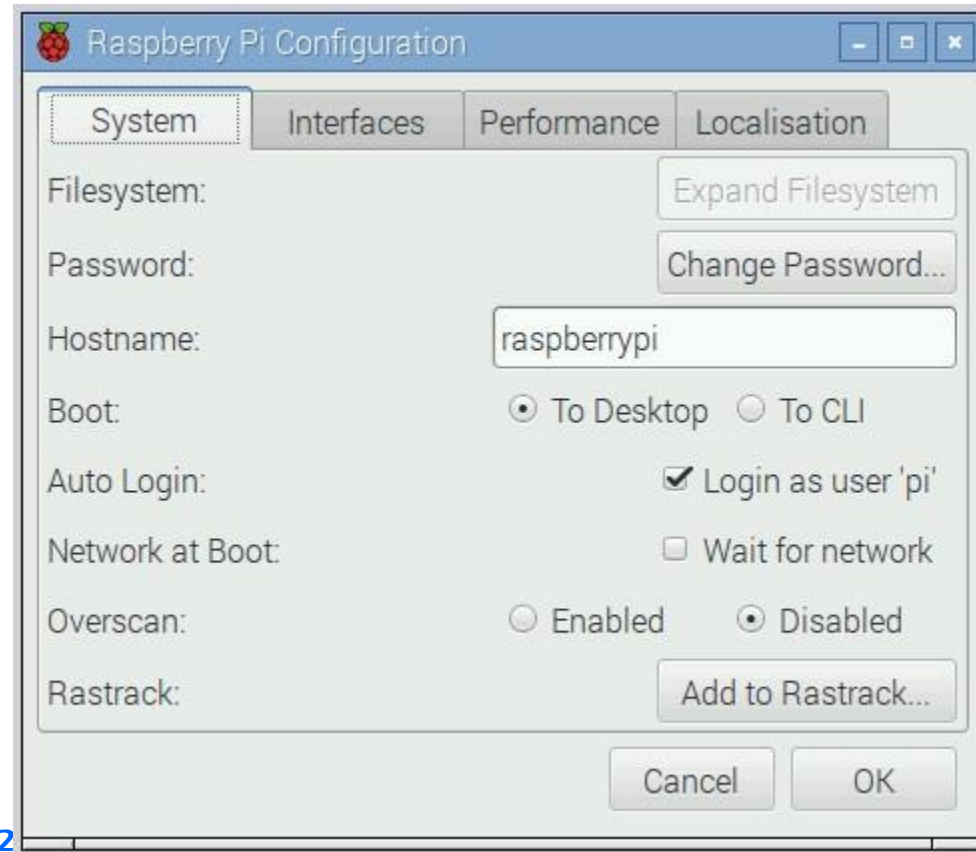
- ☐ **Localisation** : Configurez un clavier international, des options WiFi globales, et ajustez les paramètres régionaux et le fuseau horaire.

Premiers pas Raspberry Pi

❑ Paramétrage du système

❑ L'outil de configuration de Raspberry Pi affiche quatre onglets:

❑ **Système**. Les options permettant d'étendre le système de fichiers et de changer le mot de passe et le nom d'hôte sont associées à divers choix de connexion.

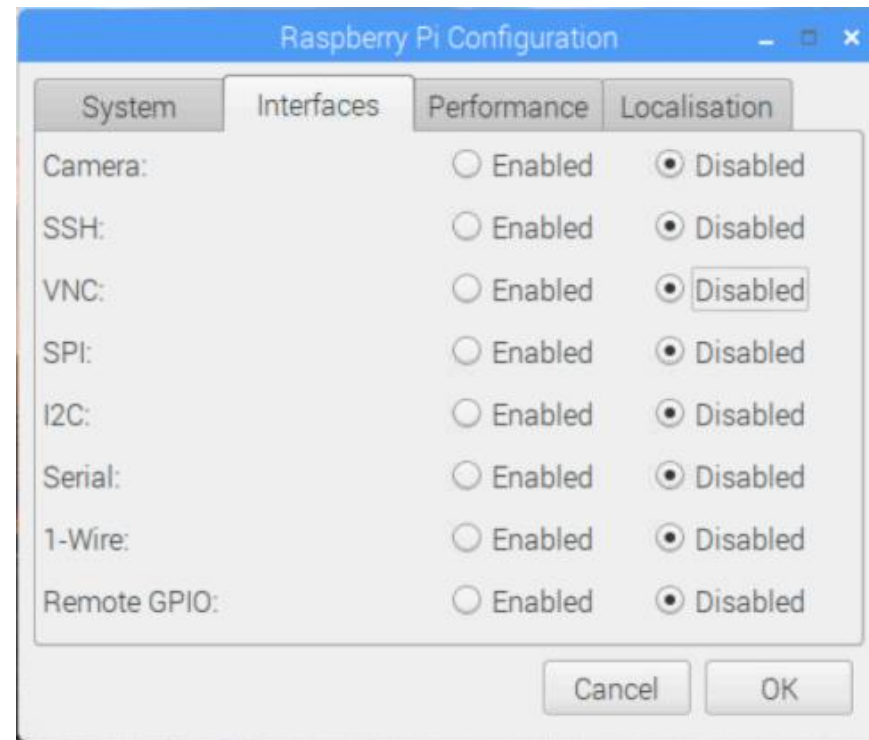


Premiers pas Raspberry Pi

❑ Paramétrage du système

❑ L'outil de configuration de Raspberry Pi affiche quatre onglets:

❑ **Interfaces** : Prise en charge des diverses fonctionnalités matérielles et logicielles, telles que le Module du Camera, SSH, VNC....

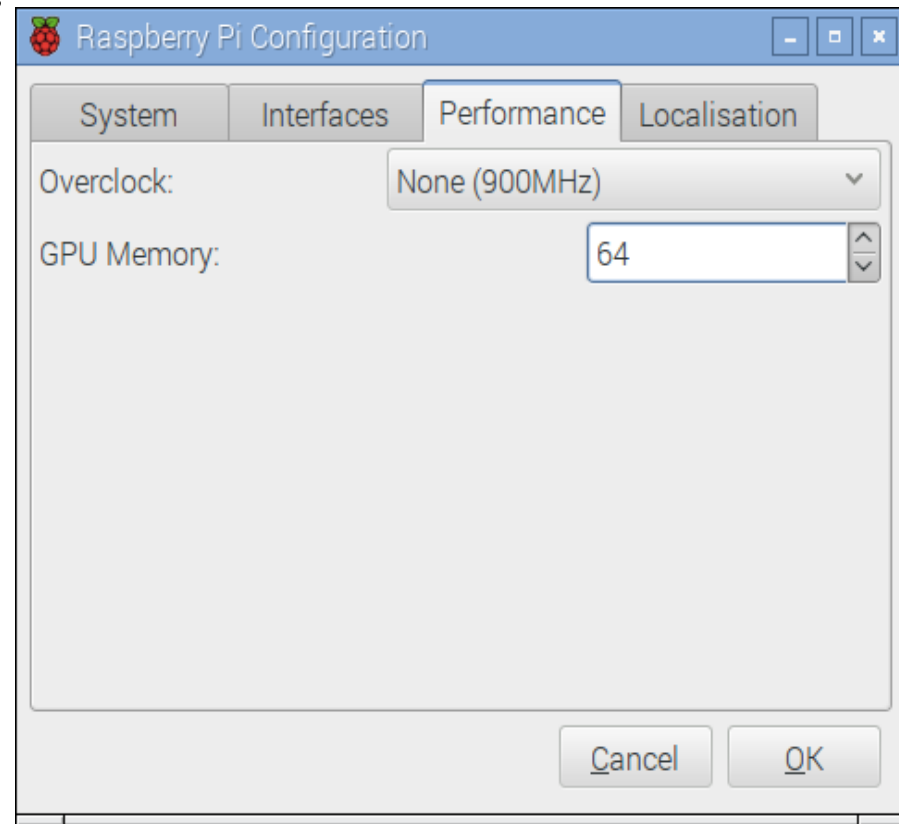


Premiers pas Raspberry Pi

❑ Paramétrage du système

❑ L'outil de configuration de Raspberry Pi affiche quatre onglets:

❑ **Performance**. configurer les options de l'Overclocking et la mémoire GPU qui améliorèrent les performances d'un Raspberry Pi.

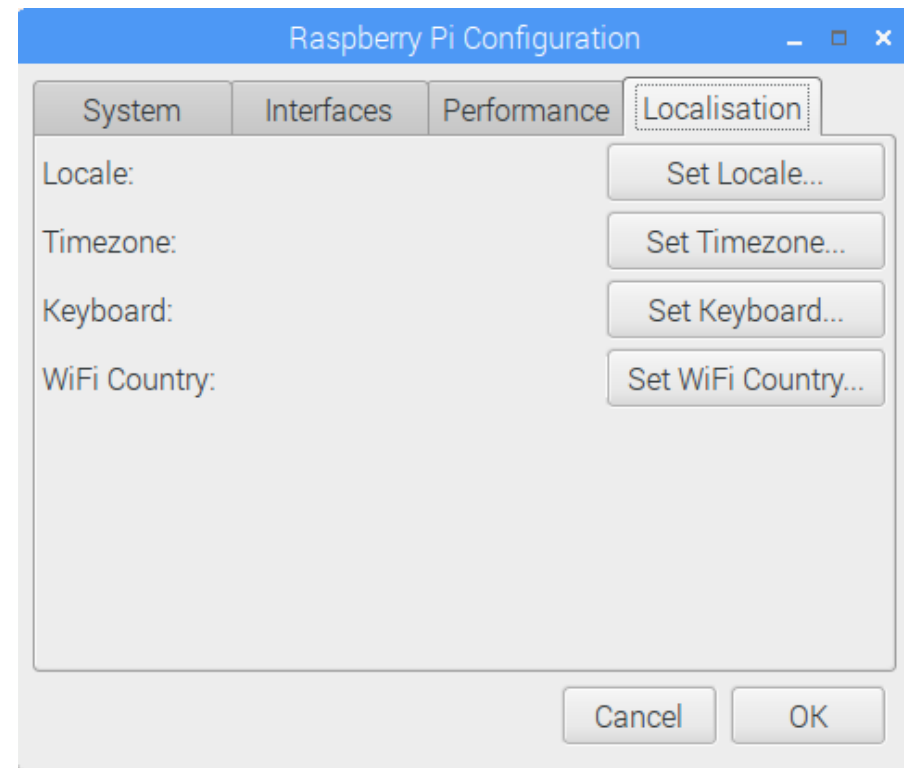


Premiers pas Raspberry Pi

❑ Paramétrage du système

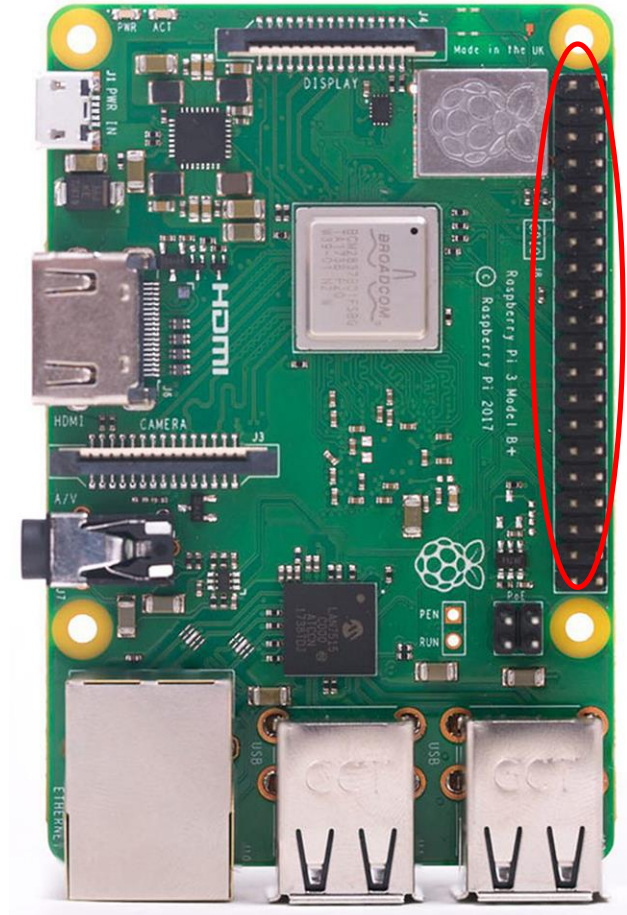
❑ L'outil de configuration de Raspberry Pi affiche quatre onglets:

- ❑ **Localisation** : pour Configurer le clavier, ajuster les paramètres régionaux le fuseau horaire et le Wifi country.

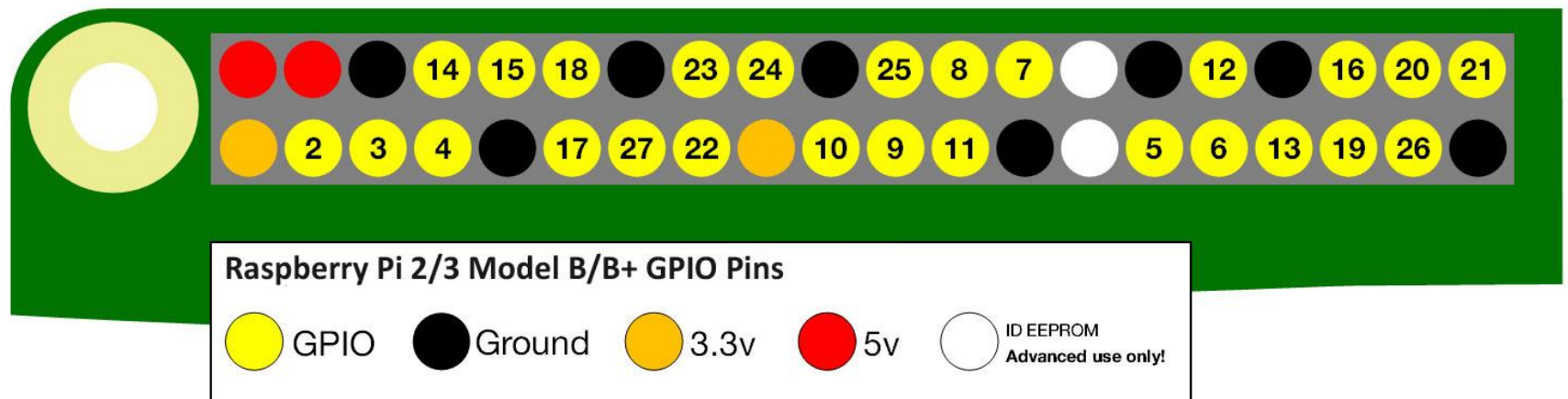


GPIO

- ❑ Une caractéristique puissante du Raspberry Pi est la rangée de broches GPIO (entrée / sortie à usage général) le long du bord supérieur de la carte.
- ❑ Un en-tête GPIO à 40 broches se trouve sur toutes les cartes Raspberry Pi actuelles (non peuplées sur Pi Zero et Pi Zero W).
 - ❑ Avant le Pi 1 Model B + (2014), les cartes comprenaient un en-tête à 26 broches plus court.

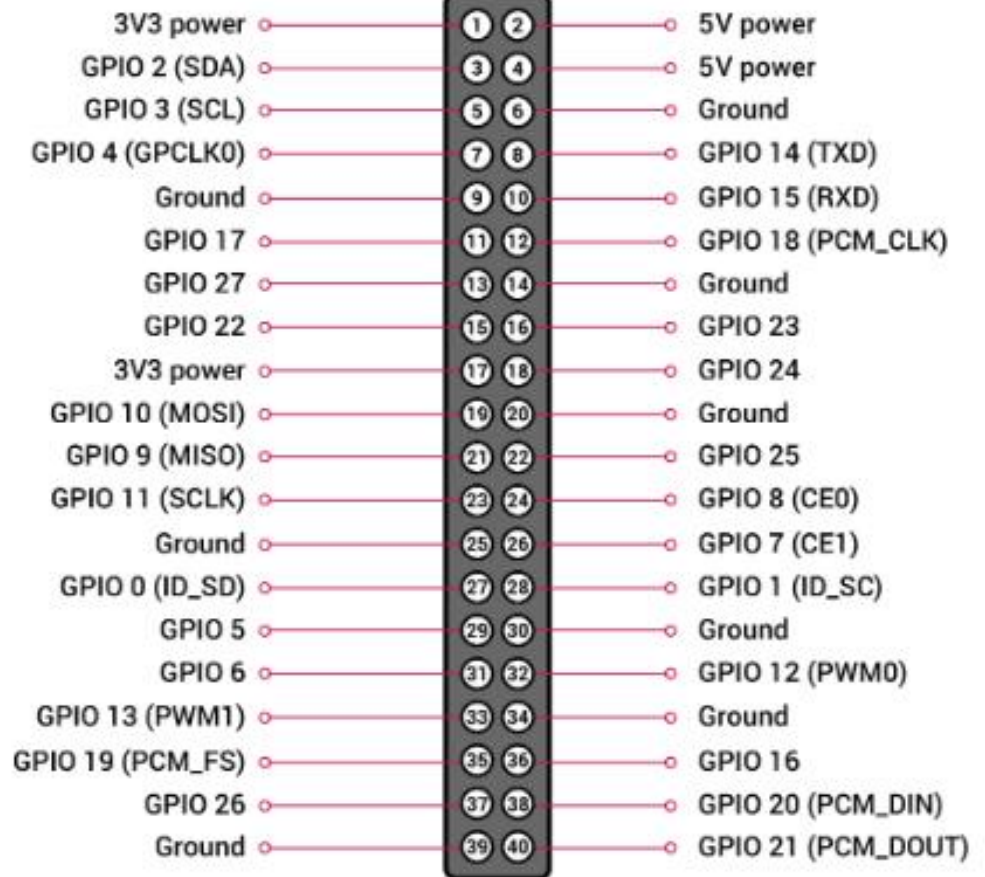
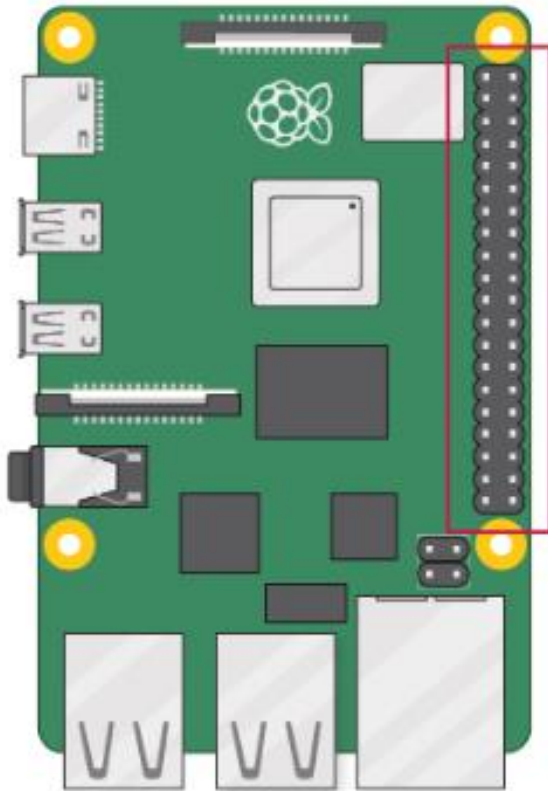


GPIO



- ❑ N'importe quelle des broches GPIO peut être désignée (par programmation) comme une broche d'entrée ou de sortie et utilisée pour un large éventail d'utilisations.

GPIO



GPIO : Numérotation

- ❑ Le Raspberry Pi autorise deux numérotations : celle de la sérigraphie du connecteur de la carte (BOARD), ou la numérotation électronique de la puce (BCM). À vous de choisir celle que vous voulez.
- ❑ Exemple :
- ❑ Si on prend la broche qui se trouve dans la deuxième ligne de la première colonne et on choisit la numérotation BOARD, on accède à cette broche par le numéro 3 et si on choisit la numérotation BCM on y accède par le numéro 2.

3.3V	1	2	5V
GPIO2 (SDA1)	3	4	5V
GPIO3 (SCL1)	5	6	GND
GPIO4 (GPIO_GCLK)	7	8	GPIO14 (UART_TXD0)
GND	9	10	GPIO15 (UART_RXD0)
GPIO17 (GPIO_GEN0)	11	12	GPIO18 (GPIO_GEN1)
GPIO27 (GPIO_GEN2)	13	14	GND
GPIO22 (GPIO_GEN3)	15	16	GPIO23 (GPIO_GEN4)
3.3V	17	18	GPIO24 (GPIO_GEN5)
GPIO10 (SPI0_MOSI)	19	20	GND
GPIO9 (SPI0_MISO)	21	22	GPIO25 (GPIO_GEN6)
GPIO11 (SPI0_CLK)	23	24	GPIO8 (SPI_CE0_N)
GND	25	26	GPIO7 (SPI_CE1_N)
ID_SD (I2C EEPROM)	27	28	ID_SC (I2C EEPROM)
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

GPIO : Voltage

- ❑ Deux broches 5V et deux broches 3V3 sont présentes sur la carte, ainsi qu'un certain nombre de broches de mise à la terre (0V), qui ne sont pas configurables.
- ❑ Les broches restantes sont toutes des broches 3V3 à usage général, ce qui signifie que les sorties sont définies sur 3V3 et les entrées sont tolérantes à 3V3.

3.3V	1	2	5V
GPIO2 (SDA1)	3	4	5V
GPIO3 (SCL1)	5	6	GND
GPIO4 (GPIO_GCLK)	7	8	GPIO14 (UART_TXD0)
GND	9	10	GPIO15 (UART_RXD0)
GPIO17 (GPIO_GEN0)	11	12	GPIO18 (GPIO_GEN1)
GPIO27 (GPIO_GEN2)	13	14	GND
GPIO22 (GPIO_GEN3)	15	16	GPIO23 (GPIO_GEN4)
3.3V	17	18	GPIO24 (GPIO_GEN5)
GPIO10 (SPI0_MOSI)	19	20	GND
GPIO9 (SPI0_MISO)	21	22	GPIO25 (GPIO_GEN6)
GPIO11 (SPI0_CLK)	23	24	GPIO8 (SPI_CE0_N)
GND	25	26	GPIO7 (SPI_CE1_N)
ID_SD (I2C EEPROM)	27	28	ID_SC (I2C EEPROM)
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

GPIO : Entrées/Sorties

- ❑ Les sorties
- ❑ Une broche GPIO désignée comme broche de sortie peut être réglée sur l'état HIGH (3V3) ou LOW (0V).
- ❑ Les entrées
- ❑ Une broche GPIO désignée comme broche d'entrée peut être lue comme HIGH (3V3) ou LOW (0V).

3.3V	1	2	5V
GPIO2 (SDA1)	3	4	5V
GPIO3 (SCL1)	5	6	GND
GPIO4 (GPIO_GCLK)	7	8	GPIO14 (UART_TXD0)
GND	9	10	GPIO15 (UART_RXD0)
GPIO17 (GPIO_GEN0)	11	12	GPIO18 (GPIO_GEN1)
GPIO27 (GPIO_GEN2)	13	14	GND
GPIO22 (GPIO_GEN3)	15	16	GPIO23 (GPIO_GEN4)
3.3V	17	18	GPIO24 (GPIO_GEN5)
GPIO10 (SPI0_MOSI)	19	20	GND
GPIO9 (SPI0_MISO)	21	22	GPIO25 (GPIO_GEN6)
GPIO11 (SPI0_CLK)	23	24	GPIO8 (SPI_CE0_N)
GND	25	26	GPIO7 (SPI_CE1_N)
ID_SD (I2C EEPROM)	27	28	ID_SC (I2C EEPROM)
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

GPIO :

- ❑ En plus des simples périphériques d'entrée et de sortie, les broches GPIO peuvent être utilisées avec une variété de fonctions alternatives, certaines sont disponibles sur toutes les broches, d'autres sur des broches spécifiques.
- ❑ PWM (modulation de largeur d'impulsion)
 - ❑ Logiciel PWM disponible sur toutes les broches
 - ❑ PWM matériel disponible sur GPIO12, GPIO13, GPIO18, GPIO19
- ❑ SPI
 - ❑ SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)
 - ❑ SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)
- ❑ I2C
 - ❑ Données: (GPIO2); Horloge (GPIO3)
 - ❑ Données EEPROM: (GPIO0); Horloge EEPROM (GPIO1)
- ❑ série
 - ❑ TX (GPIO14); RX (GPIO15)

3.3V	1	2	5V
GPIO2 (SDA1)	3	4	5V
GPIO3 (SCL1)	5	6	GND
GPIO4 (GPIO_GCLK)	7	8	GPIO14 (UART_TXD0)
GND	9	10	GPIO15 (UART_RXD0)
GPIO17 (GPIO_GEN0)	11	12	GPIO18 (GPIO_GEN1)
GPIO27 (GPIO_GEN2)	13	14	GND
GPIO22 (GPIO_GEN3)	15	16	GPIO23 (GPIO_GEN4)
3.3V	17	18	GPIO24 (GPIO_GEN5)
GPIO10 (SPI0_MOSI)	19	20	GND
GPIO9 (SPI0_MISO)	21	22	GPIO25 (GPIO_GEN6)
GPIO11 (SPI0_CLK)	23	24	GPIO8 (SPI_CE0_N)
GND	25	26	GPIO7 (SPI_CE1_N)
ID_SD (I2C EEPROM)	27	28	ID_SC (I2C EEPROM)
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

GPIO :

❑ Note : bien que la connexion de composants simples aux broches GPIO soit parfaitement sûre, il est important de faire attention à la façon de câbler les composants.

- ❑ Les LED doivent avoir des résistances pour limiter le courant qui les traverse.
- ❑ N'utilisez pas 5V pour les composants de 3V3.
- ❑ Ne connectez pas les moteurs directement aux broches GPIO, utilisez plutôt un circuit en pont H ou une carte contrôleur de moteur.

3.3V	1	2	5V
GPIO2 (SDA1)	3	4	5V
GPIO3 (SCL1)	5	6	GND
GPIO4 (GPIO_GCLK)	7	8	GPIO14 (UART_TXD0)
GND	9	10	GPIO15 (UART_RXD0)
GPIO17 (GPIO_GEN0)	11	12	GPIO18 (GPIO_GEN1)
GPIO27 (GPIO_GEN2)	13	14	GND
GPIO22 (GPIO_GEN3)	15	16	GPIO23 (GPIO_GEN4)
3.3V	17	18	GPIO24 (GPIO_GEN5)
GPIO10 (SPI0_MOSI)	19	20	GND
GPIO9 (SPI0_MISO)	21	22	GPIO25 (GPIO_GEN6)
GPIO11 (SPI0_CLK)	23	24	GPIO8 (SPI_CE0_N)
GND	25	26	GPIO7 (SPI_CE1_N)
ID_SD (I2C EEPROM)	27	28	ID_SC (I2C EEPROM)
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

GPIO :

- ❑ Une référence pratique est accessible sur le Raspberry Pi en ouvrant une fenêtre de terminal et en exécutant la commande « pinout ».

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ pinout  
  
Revision : a02082  
SoC : BCM2837  
RAM : 1024Mb  
Storage : MicroSD  
USB ports : 4 (excluding power)  
Ethernet ports : 1  
Wi-fi : True  
Bluetooth : True  
Camera ports (CSI) : 1  
Display ports (DSI) : 1  
  
J8:  
3V3 (1) (2) 5V  
GPIO2 (3) (4) 5V  
GPIO3 (5) (6) GND  
GPIO4 (7) (8) GPIO14  
GND (9) (10) GPIO15  
GPIO17 (11) (12) GPIO18  
GPIO27 (13) (14) GND  
GPIO22 (15) (16) GPIO23  
3V3 (17) (18) GPIO24  
GPIO10 (19) (20) GND  
GPIO9 (21) (22) GPIO25  
GPIO11 (23) (24) GPIO8  
GND (25) (26) GPIO7  
GPIO0 (27) (28) GPIO1  
GPIO5 (29) (30) GND  
GPIO6 (31) (32) GPIO12  
GPIO13 (33) (34) GND  
GPIO19 (35) (36) GPIO16  
GPIO26 (37) (38) GPIO20  
GND (39) (40) GPIO21  
  
For further information, please refer to https://pinout.xyz/  
pi@raspberrypi:~ $
```


GPIO : Programmation avec Python

- ❑ Pourquoi Python?
- ❑ C'est le standard "de facto" pour la programmation sur Raspberry Pi.
- ❑ C'est un langage de programmation qui reste simple et lisible.
- ❑ Il permet de facilement programmer les GPIO's de la carte
- ❑ L'écosystème (les bibliothèques ou modules additionnels) est énorme
- ❑ Python 3



Contrôle du GPIO en python

- ❑ Raspberry-gpio-python ou RPi.GPIO, est un module Python pour contrôler l'interface GPIO sur le Raspberry Pi.
- ❑ Il a été développé par Ben Croston et publié sous une licence de logiciel libre du MIT.
- ❑ Cette bibliothèque est également installée par défaut sur Raspbian OS.
- ❑ Pour utiliser cette bibliothèque, vous devez importer dans l'environnement de programmation Python:
import RPi.GPIO as GPIO
- ❑ En procédant de cette façon, vous pouvez vous y référer uniquement en tant que GPIO dans le reste de votre script.

Contrôle du GPIO en python

- ❑ Spécifier la numérotation des pins utilisée :

GPIO.setmode(GPIO.BOARD)

or

GPIO.setmode(GPIO.BCM)

- ❑ Pour détecter le système de numérotation des broches défini (par exemple, par un autre module Python):

mode = GPIO.getmode()

Contrôle du GPIO avec python

- ❑ Configurer le canal (Input or Output).
- ❑ Vous devez configurer chaque canal (pin) que vous utilisez comme entrée ou sortie.
- ❑ Pour configurer un canal comme entrée:

`GPIO.setup(# of pin , GPIO.IN)`

- ❑ Pour configurer un canal comme sortie:

`GPIO.setup(# of pin , GPIO.OUT)`

`GPIO.setup((# of pin , GPIO.OUT, initial=GPIO.HIGH)`

- ❑ Configurer plusieurs canaux par appel :

`pin_list = [11,12] # ajoutez autant de pins que vous le souhaitez!`

`GPIO.setup(pin_list, GPIO.OUT)`

Contrôle du GPIO avec python

❑ Lecture des entrées et écriture des sorties

❑ Pour définir l'état de sortie d'une broche GPIO:

`GPIO.output(pin, state)`

- ❑ (où pin est le numéro de canal basé sur le système de numérotation que vous avez spécifié (BOARD ou BCM)).
- ❑ où state : peut être `0 / GPIO.LOW / False` ou `1 / GPIO.HIGH / True`.

❑ Sortie sur plusieurs pins

- ❑ Vous pouvez émettre vers de nombreux pins dans le même appel.
Par exemple:

`pin_list = [11, 12]` # also works with tuples

`GPIO.output(pin_list, GPIO.LOW)` # sets all to GPIO.LOW

`GPIO.output(pin_list, (GPIO.HIGH, GPIO.LOW))` # sets first HIGH and second LOW

Contrôle du GPIO avec python

☐ Lecture des entrées et écriture des sorties

☐ Pour lire la valeur d'une broche GPIO:

GPIO.input(pin)

- ☐ (où pin est le numéro de canal basé sur le système de numérotation que vous avez spécifié (BOARD ou BCM)). Cela renverra 0 / GPIO.LOW / False ou 1 / GPIO.HIGH / True.

☐ Nettoyage

- ☐ À la fin de tout programme, il est recommandé de nettoyer toutes les ressources que vous pourriez avoir utilisées.
- ☐ Pour nettoyer à la fin de votre script:

GPIO.cleanup ()

Contrôle du GPIO avec python

- ❑ **Module de temporisation**

- ❑ Le module time de python fournit différentes fonctions liées au temps. Pour l'utiliser :

import time

- ❑ Il existe plusieurs fonctions importante dans ce module.
- ❑ **sleep()** : Suspend l'exécution du thread appelant pendant le nombre de secondes indiqué.

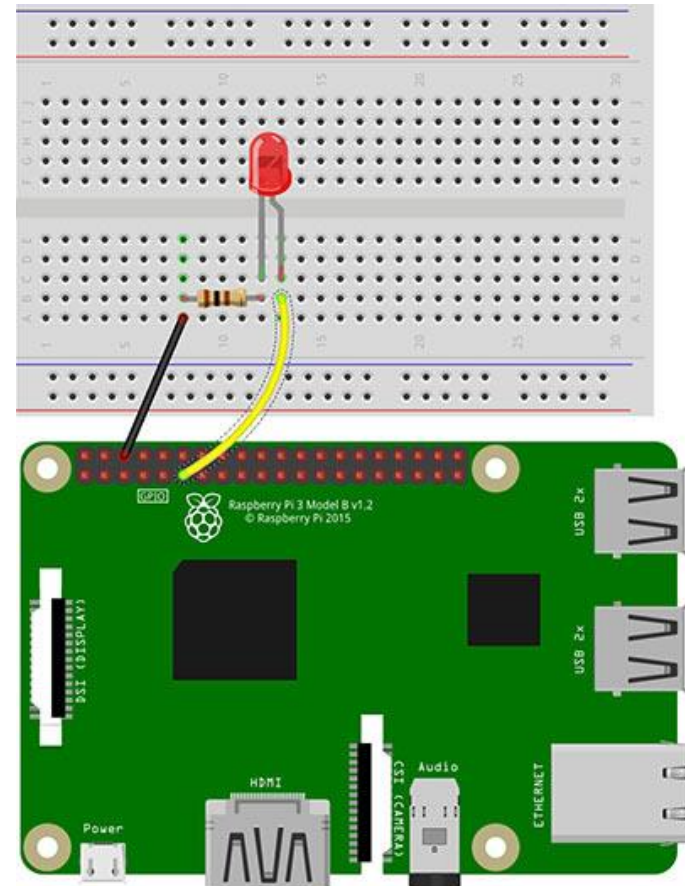
Premier exemple

*import RPi.GPIO as GPIO
from time import sleep*

*GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)*

*GPIO.output(17, True)
sleep(2)
GPIO.output(17, False)*

❑ Allumer et éteindre la LED avec RPi.GPIO



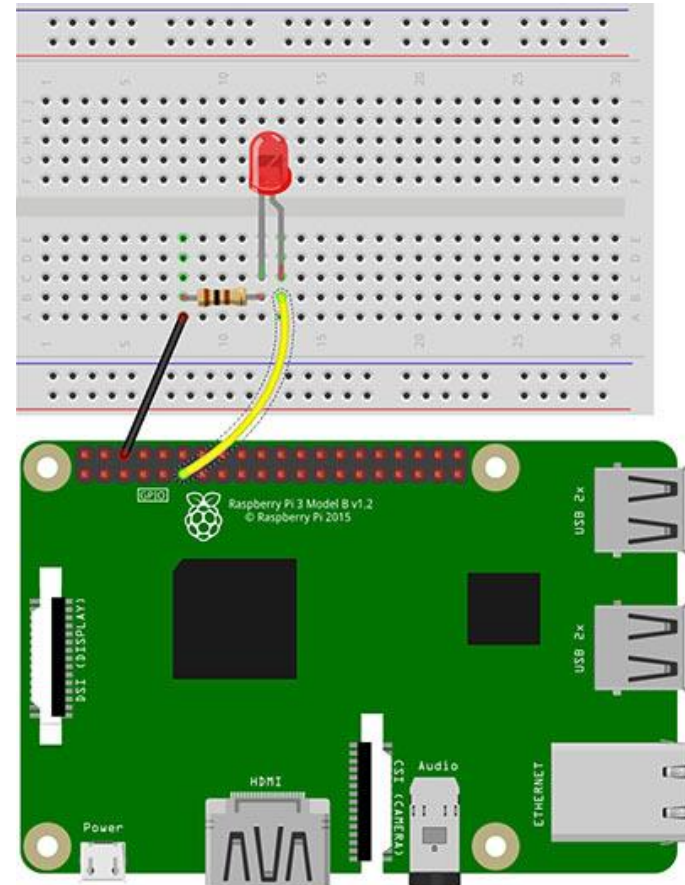
Premier exemple

```
import RPi.GPIO as GPIO  
from time import sleep
```

```
GPIO.setmode(GPIO.BCM)  
GPIO.setup(17, GPIO.OUT)
```

```
while True:  
    GPIO.output(17, True)  
    sleep(1)  
    GPIO.output(17, False)  
    sleep(1)
```

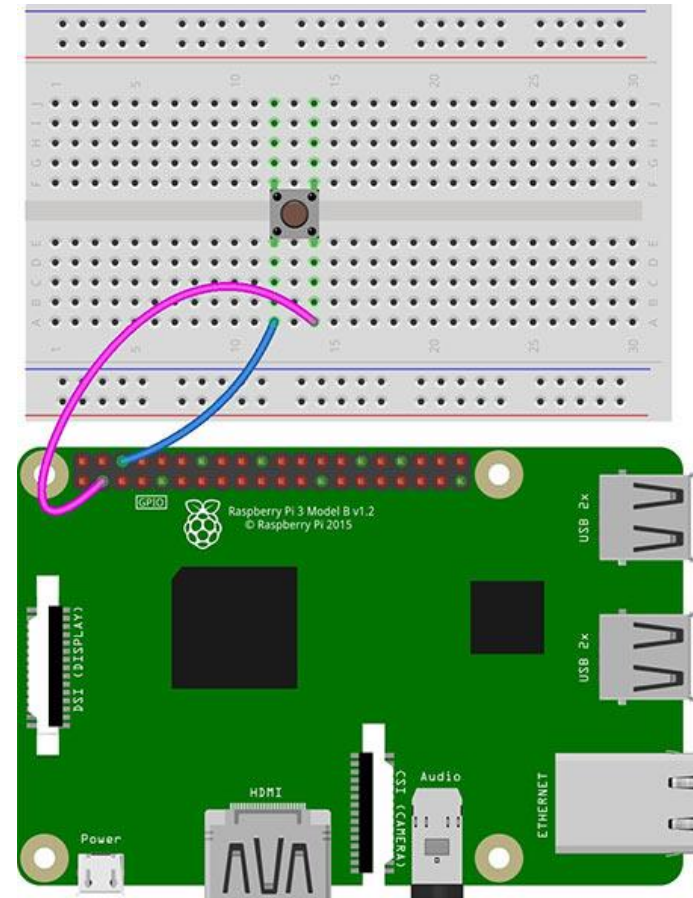
❑ Clignoter la LED avec RPi.GPIO



Deuxième exemple

- ❑ Vérifiez si un bouton est enfoncé avec RPi.GPIO

```
import RPi.GPIO as GPIO  
from time import sleep  
  
GPIO.setmode(GPIO.BCM)  
GPIO.setup(2, GPIO.IN)  
while True:  
    if GPIO.input(2) == 1 :  
        print("Button is pressed")  
        sleep(0.5)
```



Contrôle du GPIO avec python

☐ Module de `gpiozero`

- ☐ Cette bibliothèque est l'une des bibliothèques les plus complètes disponibles pour contrôler les broches d'entrée et de sortie du Raspberry Pi. Gpiozero est en langage Python.
- ☐ De nombreux composants ont déjà été définis dans cette bibliothèque. (comme une LED, un bouton, etc.);
- ☐ Par conséquent, travailler avec cette bibliothèque est assez facile et vous pouvez simplement contrôler différents composants.

Contrôle du GPIO avec python

❑ Module de gpiozero

❑ Installation

- ❑ Cette bibliothèque est disponible sur Raspbian par défaut et
- ❑ si votre SE n'est pas Raspbian, vous pouvez installer la bibliothèque à l'aide des commandes suivantes:
- ❑ Python 3: `sudo apt-get install python3-gpiozero`

❑ Utilisation

- ❑ Pour utiliser cette bibliothèque, il vous suffit de l'importer dans l'environnement de programmation Python.
- ❑ Méthode 1 : *`from gpiozero import [name of class]`*
- ❑ Méthode 2: *`import gpiozero`*

Contrôle du GPIO avec python

❑ Module de `gpiozero`

❑ Commandes définies dans la bibliothèque

❑ Commandes liées aux LED:

❑ Définition de la broche connectée à la LED: `led = LED (# of pin)`

❑ Activation: `on()`

❑ Désactivation: `off()`

❑ Modification du statut *: `toggle()`

❑ Clignotant *: `blink()`

❑ * Allumer la LED si elle est éteinte et vice-versa.

```
>>> led = LED(17)
```

```
>>> led = LED("GPIO17")
```

```
>>> led = LED("BCM17")
```

```
>>> led = LED("BOARD11")
```

Contrôle du GPIO avec python

❑ Module de gpiozero

❑ Commandes définies dans la bibliothèque

❑ Commandes liées aux LED:

❑ Exemple : Faire clignoter la LED avec gpiozero

```
from gpiozero import LED
```

```
from time import sleep
```

```
red = LED(17)
```

```
while True:
```

```
    red.on() #turn led on
```

```
    sleep(1) #delay for 1 second
```

```
    red.off() #turn led off
```

```
    sleep(1):
```

Contrôle du GPIO avec python

❑ Module de `gpiozero`

❑ Commandes définies dans la bibliothèque

❑ Commandes liées aux boutons :

- ❑ Définition de la broche connectée au bouton: `button = Button (# of pin)`
- ❑ Attendez jusqu'à ce qu'un bouton soit enfoncé: `wait_for_press ()`
- ❑ Attendez que le bouton soit relâché: `wait_for_release ()`
- ❑ Si le bouton est enfoncé retourne 1 sinon retourne 0: `is_pressed`
- ❑ Exécuter une fonction spécifique lorsque le bouton est enfoncé:
`when_pressed`
- ❑ Exécuter une fonction spécifique lorsque le bouton est relâché:
`when_released`

Contrôle du GPIO avec python

❑ Module de gpiozero

❑ Commandes définies dans la bibliothèque

❑ Commandes liées aux boutons :

❑ Exemple :

```
from gpiozero import Button
```

```
from time import sleep
```

```
button = Button(2)
```

```
while True:
```

```
    if button.is_pressed:
```

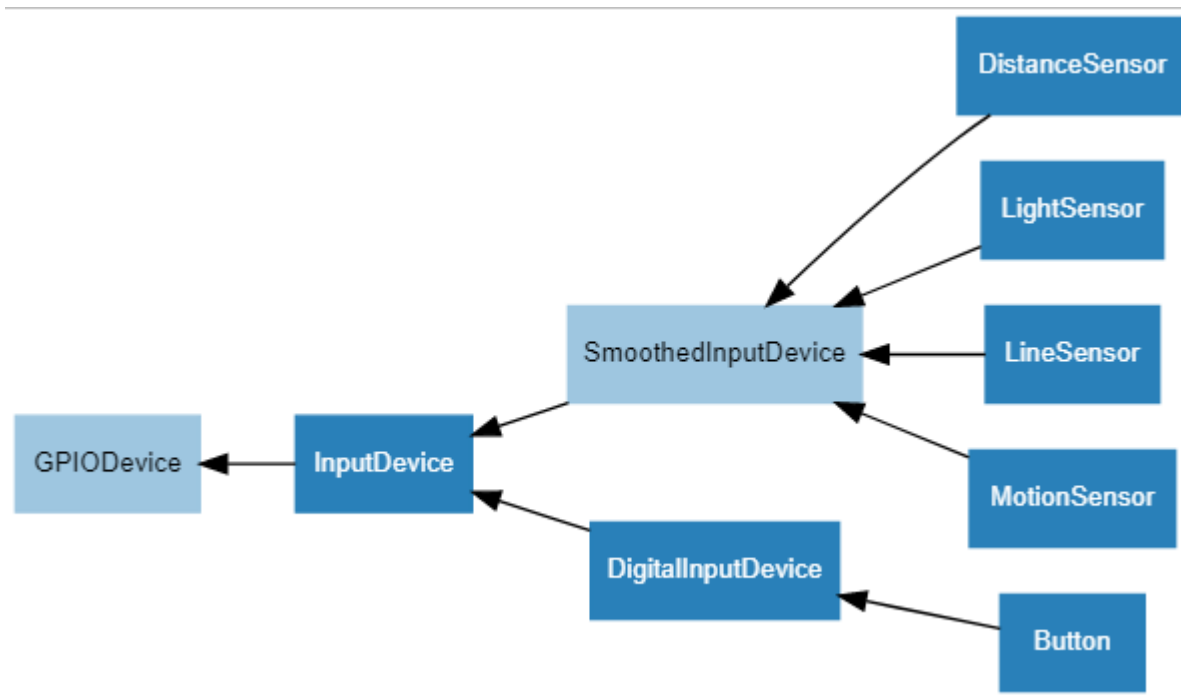
```
        print("Button is pressed")
```

```
        sleep(0.5)
```


Contrôle du GPIO avec python

❑ Module de gpiozero

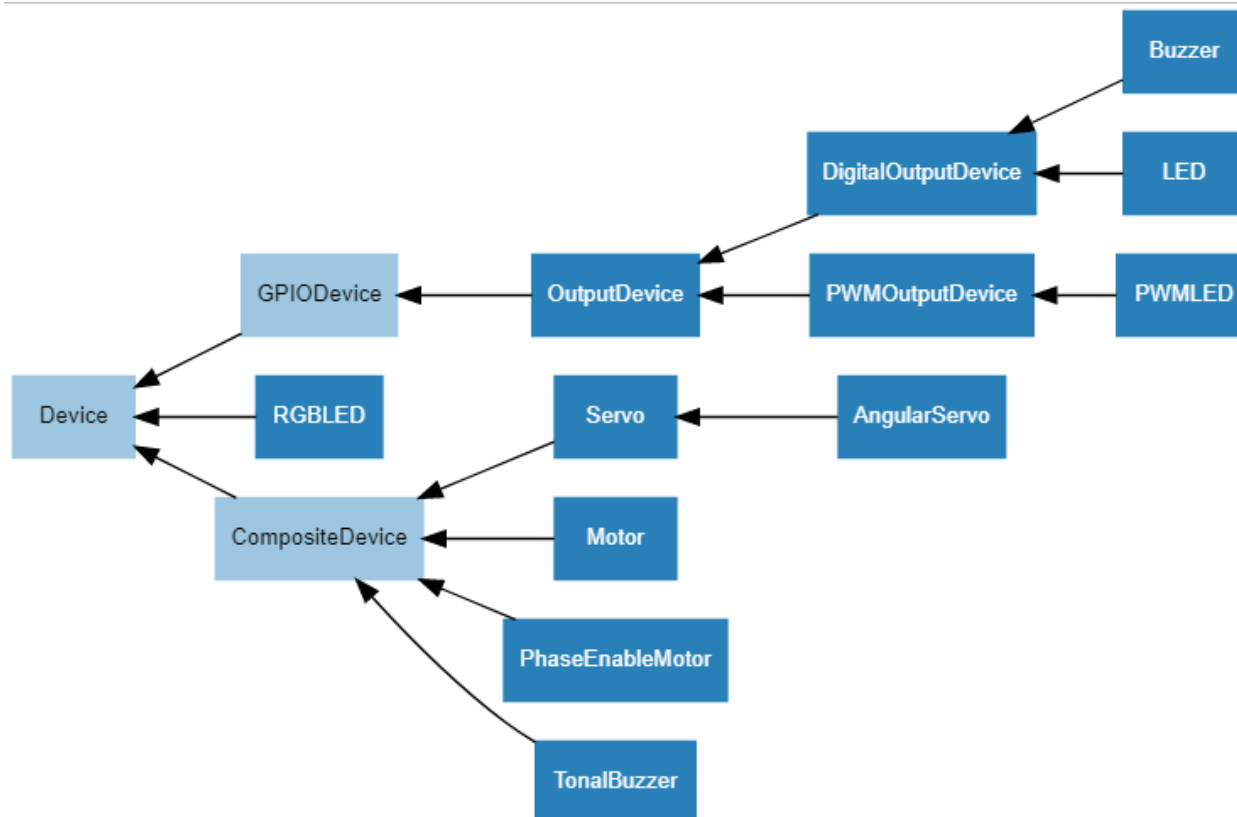
❑ API – périphériques d'entrées (classes de base)



Contrôle du GPIO avec python

❑ Module de gpiozero

❑ API – périphériques de sorties (classes de base)



Questions :

