

Chapitre2 : Les objets IoT à base de μ -contrôleur

A/ Arduino

Le système Arduino est une plate-forme de prototypage basée sur un microcontrôleur ATMEL. C'est un circuit de commande capable de piloter des capteurs et des actionneurs afin de simuler ou créer des systèmes automatisés. Elle dispose d'un logiciel de programmation qui permet de programmer la carte en fonction du comportement désiré. Il existe une grande variété de cartes Arduino : Mega 2560, Uno R3, YUN, Mini, Nano et Leonardo.

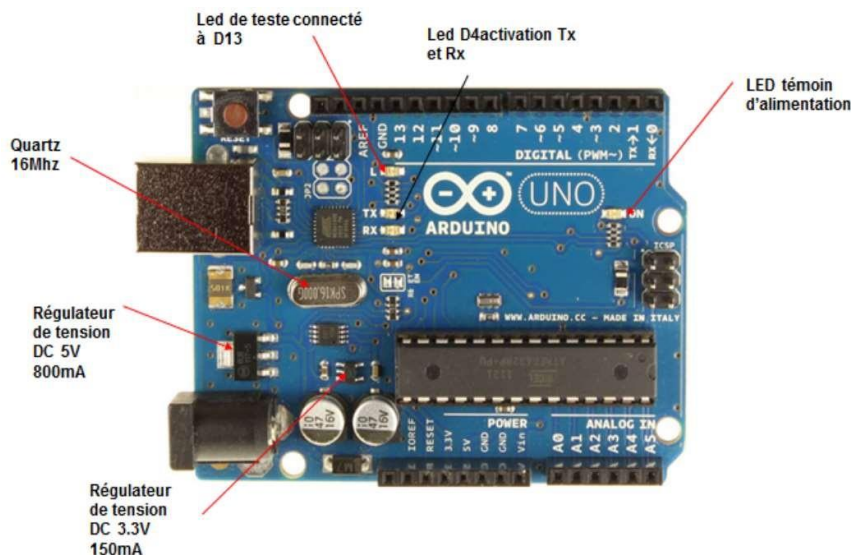


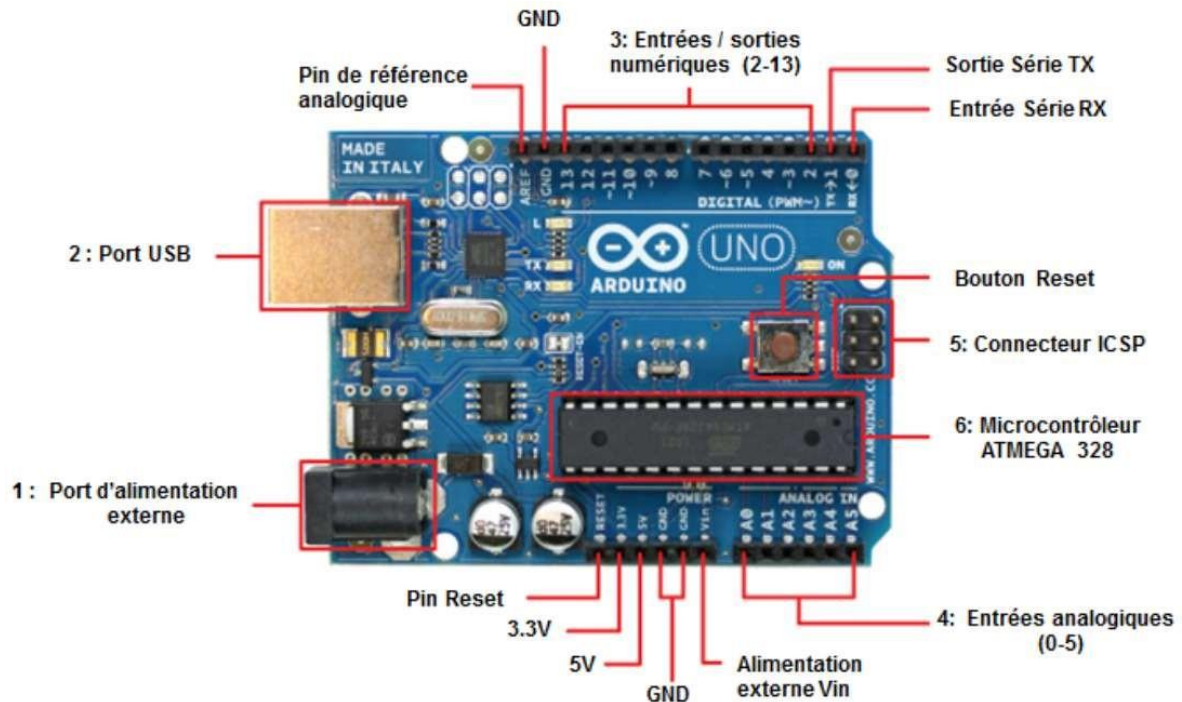
- la carte Arduino est peu coûteuse, par exemple la carte Arduino UNO a un prix qui ne dépasse pas 50 dinars. Les environnements matériel et logiciel sont sous licence libre.
- Le matériel : cartes électroniques dont les schémas sont en libre circulation sur internet
- Le logiciel : gratuit et open source, développé en Java.

Une large communauté d'amateurs et de passionnés contribuent à développer des applications et à les partager.

1. Anatomie de la carte Arduino UNO

Les deux schémas suivants détaillent l'anatomie de la carte Arduino Uno :





port d'alimentation externe Pour fonctionner, la carte a besoin d'une alimentation qui est comprise entre 6 et 20 V. Onconseille en général de l'alimenter plutôt entre 7 V et 12Vpour garder une marge en basse tension et éviter que le circuit ne chauffe trop (car le régulateur de tension disperse toute surtension en chaleur). Cette tension doit être continue et peut par exemple être fournie par une pile 9V. Un régulateur se charge ensuite de réduire la tension à 5V pour le bon fonctionnement de la carte.

1 : port USB Permet de communiquer avec la carte et de l'alimenter en 5V.

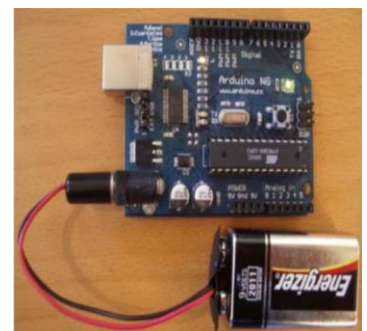
2 : Les entrées/sorties numériques 14 entrées/sorties numériques dont 6 peuvent assurer une sortie PWM (Pulse Width Modulation) pouvant actionner de nombreux composants (LED, transistor, etc.) mais elles ne peuvent pas fournir beaucoup de courant (40 mA pour une carte Arduino UNO). Pour piloter des circuits de plus forte puissance, il faut passer par des transistors ou des relais.

3 : Les entrées analogiques lui permettent de mesurer une tension variable (entre 0 et 5 V) qui peut provenir de capteurs ou d'interfaces diverses : potentiomètres, etc. il sont au nonre de 6.

4 : Connecteur In-Circuit Serial Programming ICSP pour le téléchargement du programme.

6: Microcontrôleur ATmega328, c'est unmicrocontrôleur ATMEL d

Entrée/Sortie série : broche 0 (RX) permet de recevoir (RX) et broche1 (TX) permet de transmettre desdonnées séries TTL. Ces broches sont raccordées à leurs homologues sur le chip Atmega8U2 spécialisé dans la conversion USB-to-TTL série.



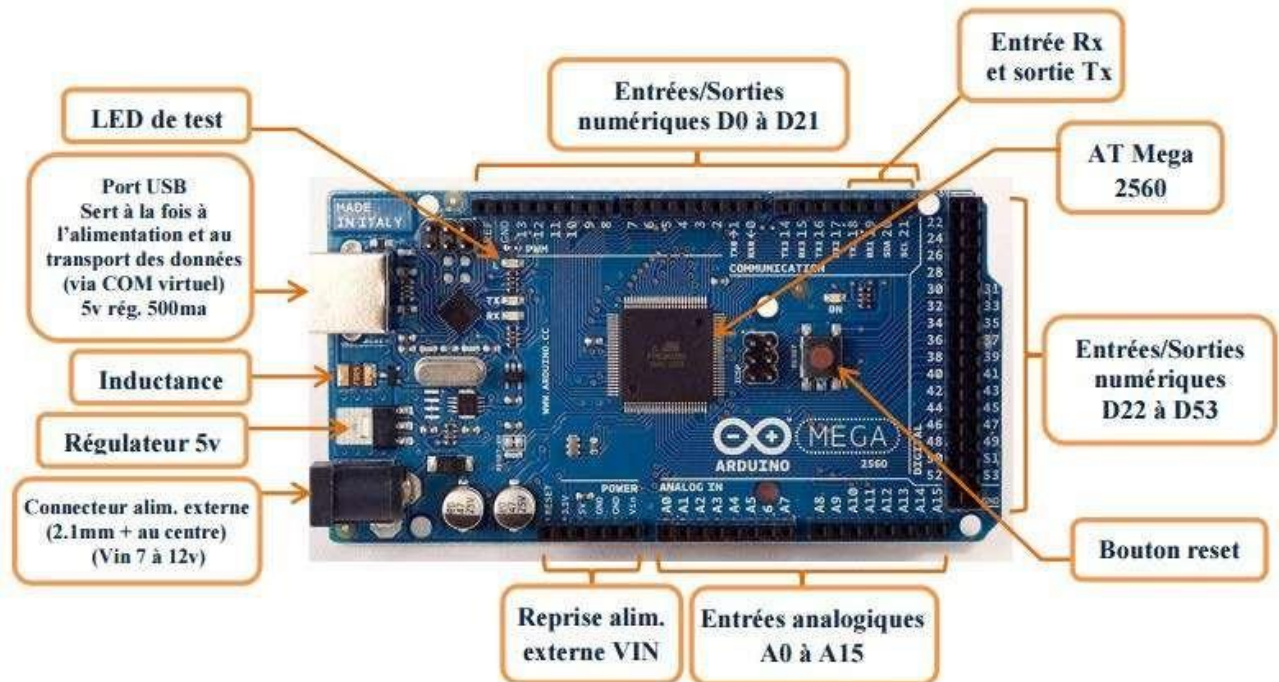
Une pile 9v et un connecteur

(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)
VCC	7	22	GND
GND	8	21	AREF
(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)
(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)

atmega328

2. Comparaison entre cartes Arduino UNO et Arduino MEGA

UNO		MEGA	
Microcontroller	ATmega328	Microcontroller	ATmega2560
Operating Voltage	5V	Operating Voltage	5V
Input Voltage (recommended)	7-12V	Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V	Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)	Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	6	Analog Input Pins	16
DC Current per I/O Pin	40 mA	DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA	DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader	Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	2 KB (ATmega328)	SRAM	8 KB
EEPROM	1 KB (ATmega328)	EEPROM	4 KB
Clock Speed	16 MHz	Clock Speed	16 MHz



Carte Arduino MEGA
2560

3. Description de la structure d'un programme

Un programme utilisateur Arduino est une suite d'instructions élémentaires sous forme textuelle, ligne par ligne. La carte lit puis exécute les instructions les unes après les autres, dans l'ordre défini par les lignes de code, comme mors d'une programmation classique. Cette structure se décompose en trois parties :

- Description des constantes et variables globales du programme
- Fonction principale **void setup()**: cette partie ne sera exécutée qu'une seule fois, elle est destinée pour la configuration des modes entrées ou sorties des pin et d'autres éléments à configurer (vitesse port série...)
- Fonction boucle **void loop()**: contient le fonctionnement général du programme, la suite d'instruction de cette fonction s'exécutent en continue

Remarque : Il est possible d'ajouter des commentaires au programme. Pour cela on peut procéder de deuxmanière :

- avant une ligne de commentaire en ajoutant « // »
- en encadrant les lignes de commentaires entre « /* » et « */ »

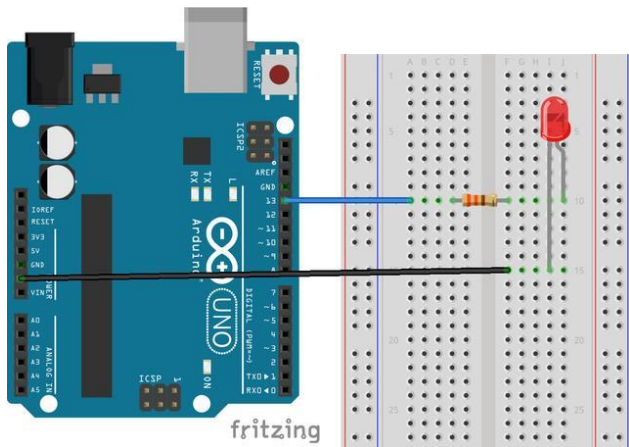
4. Exemple d'application Arduino Blink :

Allume une LED pendant une seconde, puis s'éteint pendant une seconde, à plusieurs reprises.

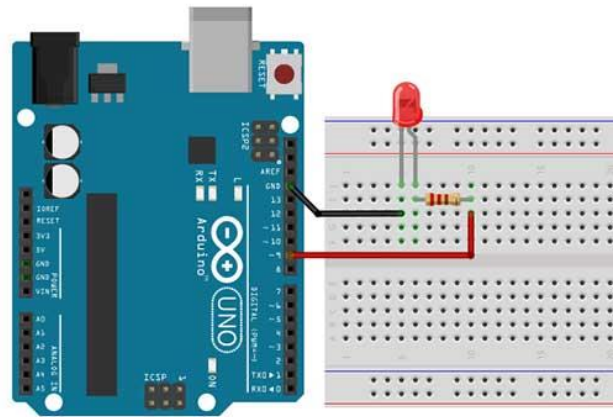
La plupart des Arduinos ont une LED intégrée que vous pouvez contrôler. Sur les UNO, MEGA et ZERO il est attaché à la broche numérique 13, sur MKR1000 sur la broche 6.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

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



Blink built-in broche 13



Montage pour faire clignoter une LED connectée au PIN 9

```
const int LED_9 = 9;

// the setup function runs once when you press reset or power the board

void setup() {
  pinMode(LED_9, OUTPUT); // initialize digital pin LED_BUILTIN as an output.
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_9, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(LED_9, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

5. Les différents types de données : un type de donnée définit les valeurs que peut prendre une variable ou une constante. On déclare une variable de la façon suivante :

type_de_données nom_de_la_variable ;

Exemple : **int ledRouge**


Une constante est une variable dont la valeur est inchangeable lors de l'exécution d'un programme. On la déclare de la façon suivante :

const type_de_données nom_de_la_constant


Exemple : **const int led2=2 ;**

NOM DU TYPE	VALEUR MIN/MAX	TAILLE EN MEMOIRE
VALEURS BINAIRES		
boolean	0/1	1 octet
VALEURS NUMERIQUES ENTIERES SIGNEES		
int	-32 768 / +32 767	2 octets
long	-2 147 483 648 / +2 147 483 647	4 octets
VALEURS NUMERIQUES ENTIERES NON SIGNEES		
byte	0 / +255	1 octet
unsigned int	0 / +65535	2 octets
word	0 / +65535	2 octets
unsigned long	0 / +4 294 967 295	4 octets
VALEURS NUMERIQUES A VIRGULE		
float	-3.4028235E+38 / +3.4028235E+38	4 octets
double	-3.4028235E+38 / +3.4028235E+38	4 octets
CARACTERES		
char	-128 / +127 (ASCII)	1 octet

5. Compilation du programme

La compilation permet de vérifier si le code contient des erreurs de syntaxes. En cas d'anomalie de compilation, le compilateur renseigne sur le type d'erreur et la ligne où elle se trouve. Pour lancer la compilation, il faut appuyer sur le bouton . A ce moment-là, le bouton devient jaune et la zone de message affiche « Compiling » indiquant que la compilation est en cours. Si la compilation se déroule sans erreur, le message « Done compiling » apparaît, suivi de la taille du programme.

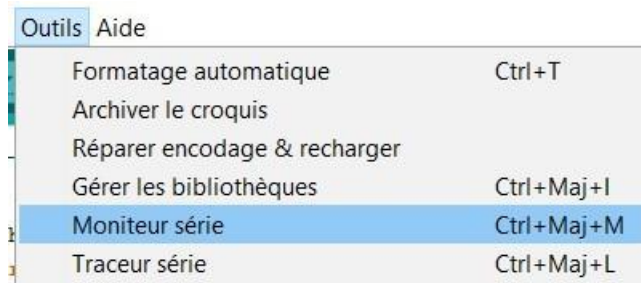
6. Téléverser le code de l'application dans la carte Arduino

Pour transférer le programme vers la carte Arduino appuyer sur le bouton , mais avant cela il faut sélectionner la bonne carte Arduino (Uno ou Mega2560...) depuis le menu **Outils-> Type de carte**. La carte doit évidemment être connectée à l'ordinateur via un câble USB. Ensuite il faut sélectionner le bon port série depuis le menu **Outils ->Port**. Sous Windows, sélectionner le port COM3 ou supérieur pour une liaison avec câble USB.

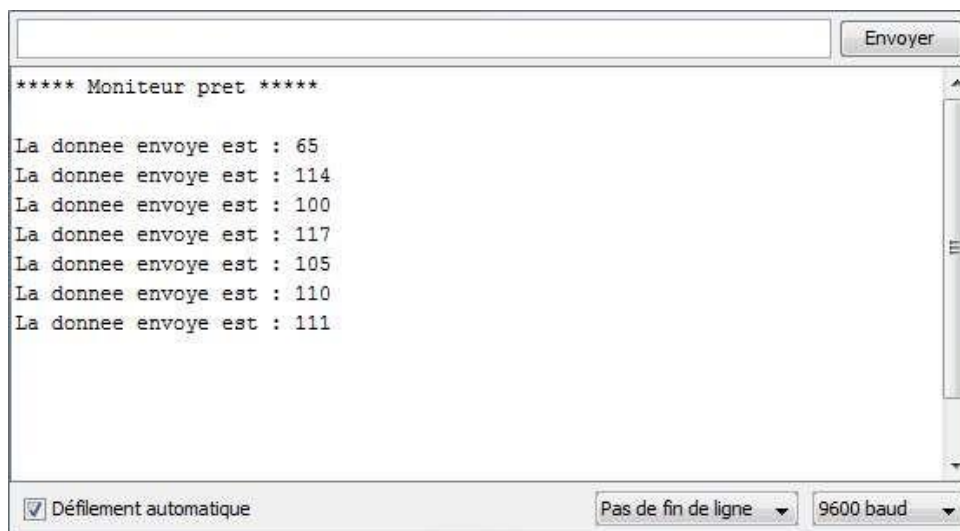
8. Le Moniteur Série

Il se nomme "moniteur série" car il utilise la communication dite "série" entre le "PC" et la carte Arduino qui sont connectés via leurs ports "USB". La carte Arduino n'est pas dotée d'écran, le moniteur série est considéré comme l'écran de cette carte. Il est utile pour faire un affichage des données à partir d'une carte Arduino, il est indispensable pour les tests de programmes.

Le lancement du moniteur série se fait à partir du menu Outils



Exemple de moniteur série qui affiche des données envoyées par la carte Arduino.



Le débit de communication entre les deux périphériques est mesuré en **bauds**. Ce dernier, est une unité de mesure de débit qui correspond au nombre de bits transmis par seconde. Le port série connecté à la carte Arduino utilise un débit de 9600 bauds.

Pour initialiser cette vitesse rajouter la ligne suivante dans la fonction Setup()

Serial.begin(9600);

Dans le corps du code Arduino, il est possible d'utiliser la méthode :

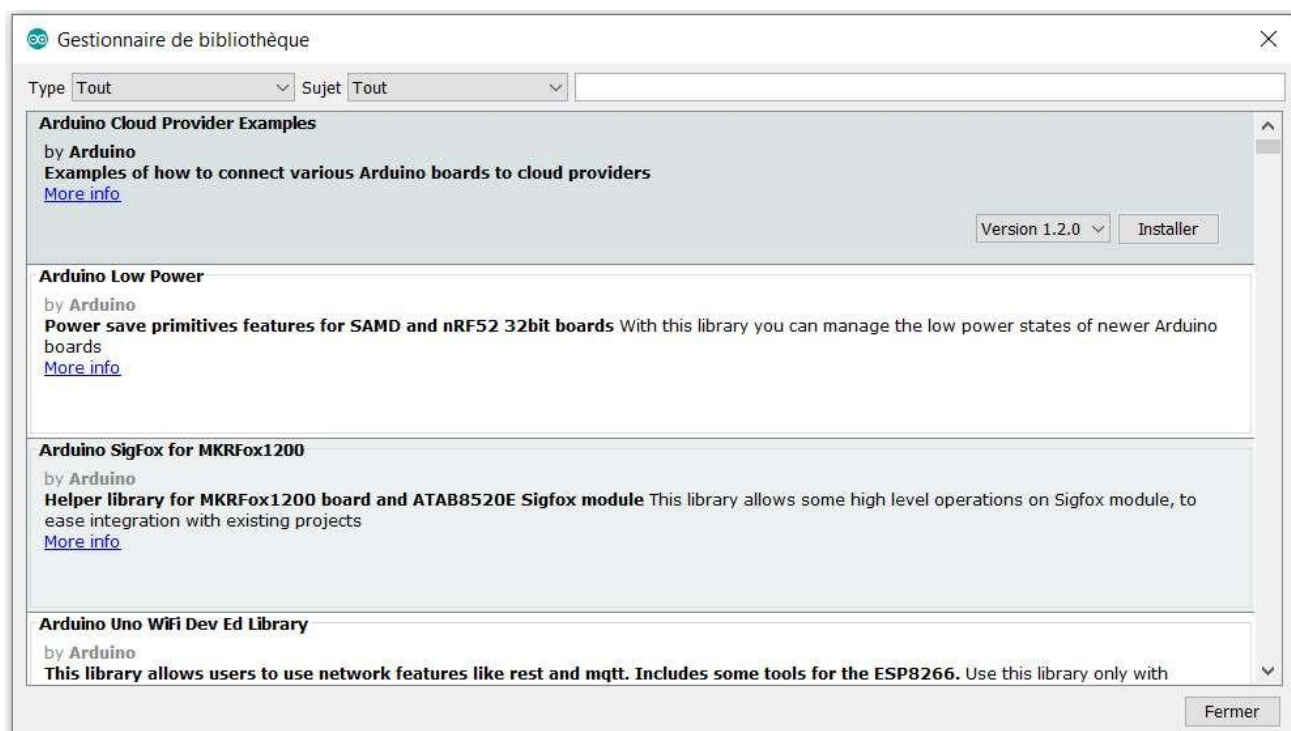
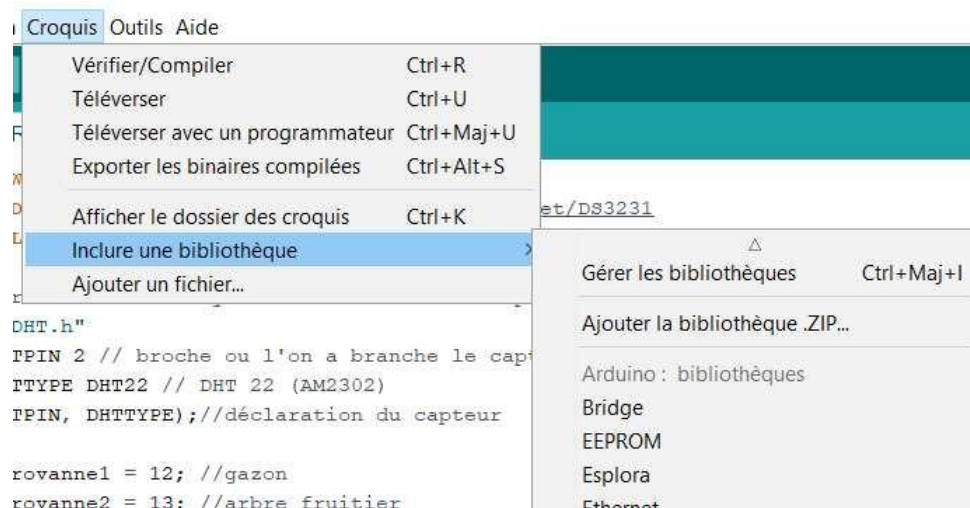
Serial.print(variable ou bien texte)

Serial.println(variable ou bien texte) pour écrire sur le port série et afficher sur le moniteur série s'il est activé.

Exercice d'application : modifier l'application de l'exemple précédent pour pouvoir afficher sur le moniteur série l'état du Led : allumé ou éteint.

7. Le gestionnaire des bibliothèques :

Il permet d'installer de nouvelles bibliothèques pour un matériel spécifique, ces bibliothèques soient elles sont téléchargeables sous format fichier .zip ou à installer à partir le gestionnaire de bibliothèque. Le gestionnaire de bibliothèque se trouve se le menu **Croquis**.

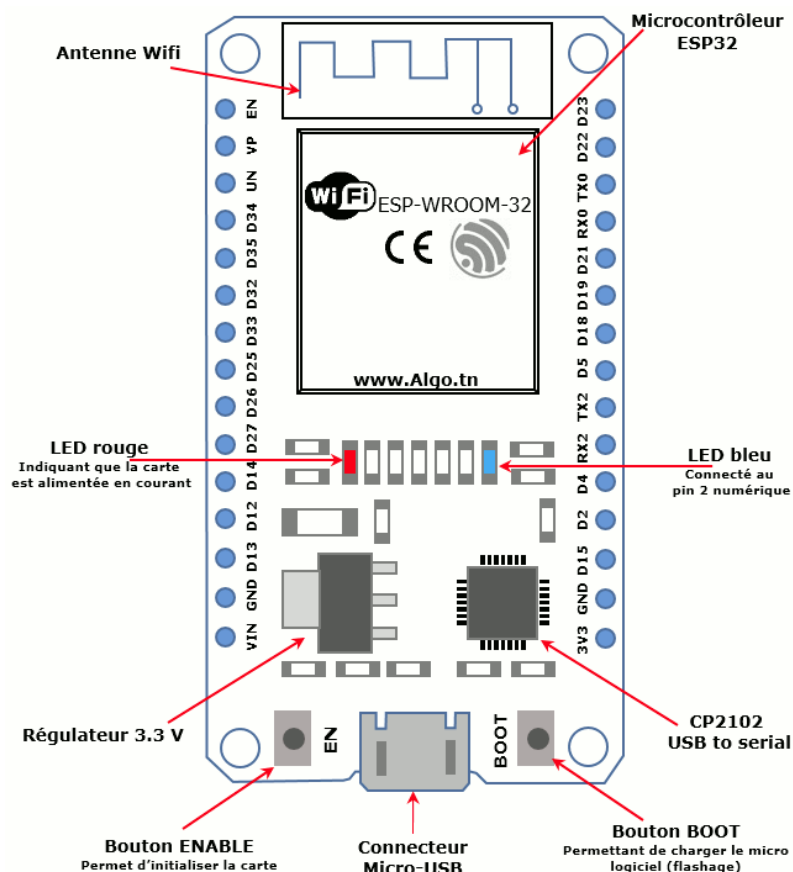
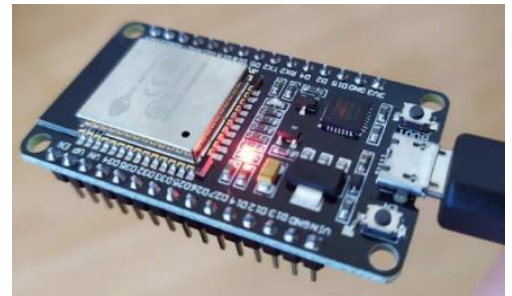


B/ la carte ESP32

L'ESP32, développé par la société Espressif Systems. C'est un système sur puce (SoC) populaire utilisé pour le développement de projets liés à l'Internet des objets (IoT) et d'autres applications embarquées. Ces applications peuvent inclure le contrôle de périphériques IoT, la collecte de données, la surveillance à distance, les projets domotiques, et bien d'autres encore. Les communautés de développeurs sont actives, ce qui facilite l'accès à de nombreuses ressources et bibliothèques pour étendre les fonctionnalités de l'ESP32 dans divers projets.

Voici quelques caractéristiques clés de l'ESP32 :

- **Microcontrôleur Dual-Core** : L'ESP32 est équipé de **deux cœurs processeurs** (dual-core), généralement basés sur l'architecture Xtensa, ce qui permet **d'exécuter plusieurs tâches en parallèle**. La fréquence est de **240 MHz**.
- **Connectivité sans fil** : Il prend en charge le **Wi-Fi** pour la connectivité à Internet (**Wifi 802.11 b/g/n 2,4 GHz**) et le **Bluetooth** pour les **communications sans fil avec d'autres dispositifs** (Classique / BLE).
- **Mémoire intégrée** : Il intègre une quantité significative de mémoire, comprenant généralement de la **mémoire Flash (4 Méga)** pour le stockage du programme et de la mémoire **RAM** pour l'exécution des programmes (**SRAM: 512 kB**).
- **Faible consommation d'énergie** : L'ESP32 est conçu pour être économe en énergie, ce qui le rend **adapté aux applications alimentées par batterie**.
- **Support pour le développement Arduino** : Il est **compatible avec l'environnement de développement Arduino**, ce qui facilite le processus de programmation pour de nombreux développeurs.



ESP32 dispose des ports **GPIO** (**G**eneral **P**urpose **I**nput/**O**utput), très utilisés dans le monde des microcontrôleurs, en particulier dans le domaine de l'électronique embarquée, Elles sont placées sur un circuit électronique afin de communiquer avec des composants électroniques et circuits externes. Tels que les sensors pour collecter des données, ou encore un relais pour commander des dispositifs électriques.

E/S disponibles :

- 15 E/S numérique (digitale)

Parmi ces 15 pin, 10 sont compatibles PWM (Pulse Width Modulation ou bien « Modulation de Largeur d'Impulsion ». C'est une technique couramment utilisée pour synthétiser des signaux pseudo analogiques à l'aide de circuits numériques. Par exemple ça peut servir par exemple pour contrôler la luminosité d'une LED, changer la couleur d'une LED RGB ou encore piloter la vitesse d'un moteur.

- 15 x entrées analogiques (ADC)
- 2 x sorties analogiques (DAC) (Digital To Analogic Converter)

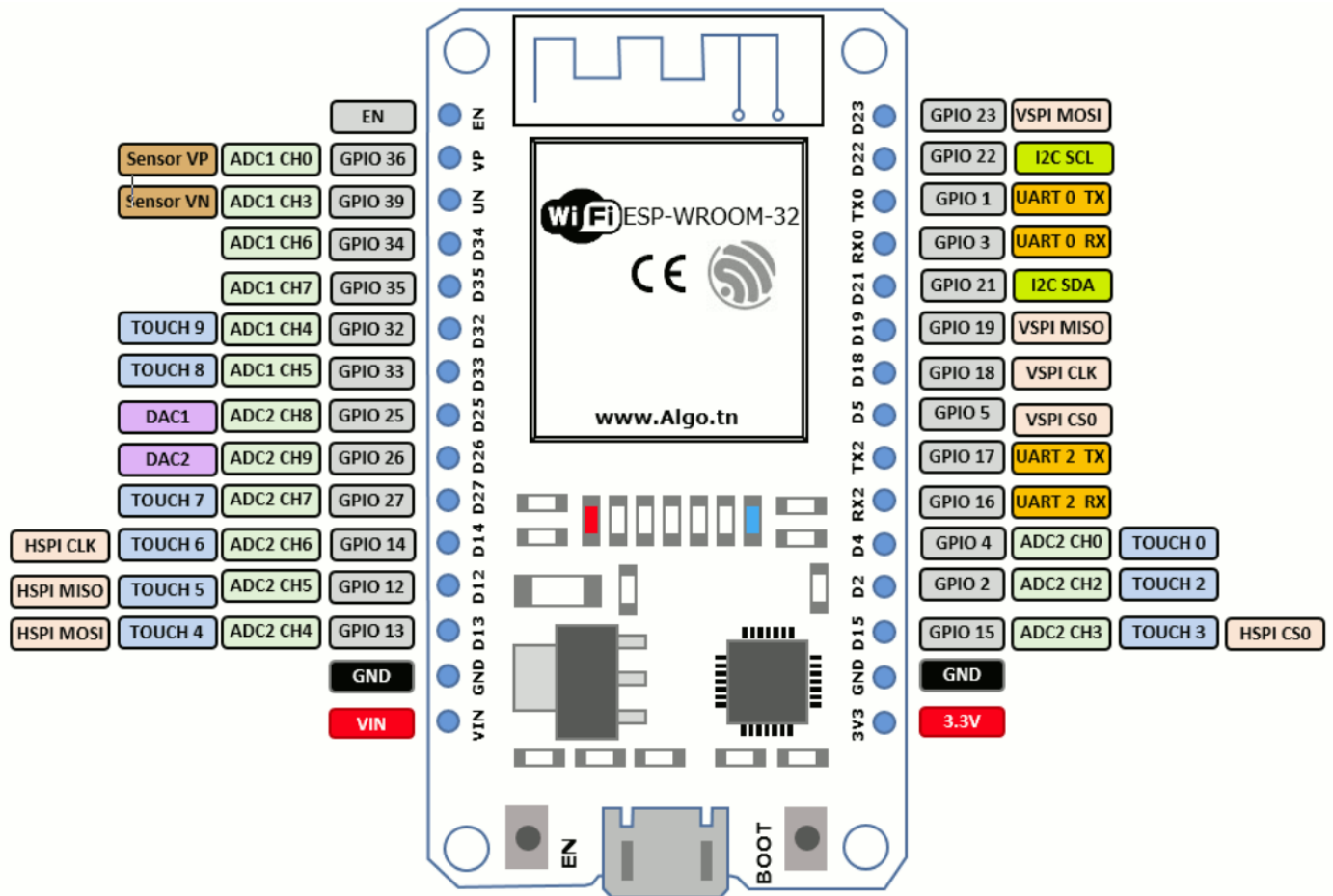
On trouve d'autres types d'Interfaces dans les ports GPIO : Les ports GPIO offrent une grande flexibilité pour connecter divers périphériques. On peut assigner plusieurs fonctions au même pin, grâce au circuit de multiplexage de la carte ESP32. On peut choisir le rôle d'un pin (**UART, I2C, SPI**) par programmation.

- Des ports UART (**U**niversal **A**synchronous **R**eceiver **T**ransmitter) : L'UART est une liaison série permettant l'envoi et la réception de données. C'est un circuit électronique qui permet la communication série asynchrone entre des appareils. Il est couramment utilisé dans les systèmes embarqués, notamment pour échanger des données entre des microcontrôleurs. Elle a une architecture assez simple, qui se résume :
 - une alimentation (VCC et GND),
 - un pin d'émission (TX)
 - un pin de réception (RX).
- Des ports I2C : Le bus I2C (Inter Integrated Circuit Bus) est un bus série développé par Philips, principalement pour les applications de domotique. Il permet de relier facilement un microprocesseur à différents circuits, facilitant ainsi la communication asynchrone entre ces dispositifs. Le bus I2C permet de faire communiquer grâce à seulement 4 fils :
 - SDA (Serial Data Line) : ligne de données bidirectionnelle
 - SCL (Serial Clock Line) : ligne d'horloge de synchronisation bidirectionnelle.
 - Vcc (Voltage Common Collector)
 - GND (Masse)

Exemple : bus I2C intégré à un Afficheur LCD pour faciliter l'affichage à partir d'une carte Arduino/Esp32

- Ports SPI : (Serial Peripheral Interface) est un **bus de données série synchrone** qui opère en mode full-duplex. Les circuits communiquent selon un schéma maître-esclave, où le maître contrôle la communication. Par exemple les BUS SPI sont utilisé dans les modules de communication radio NRF

Voici une description exhaustive des pin ESP32 :



- EN Enable :(broche du régulateur 3.3 V) On peut utiliser cette broche connectée à un bouton poussoir et au GND pour redémarrer la carte ESP32.
- 15 ADC (Analog-to-Digital Converter) convertisseur analogique-numérique) pour lire les entrées analogiques
- 4 interfaces SPI: SPI0 et SPI1 (réservés) , HSPI et VSPI(peuvent être utilisés).
- GPIO 34,35,36 et 39 à utiliser comme entrée seulement
- 9 capteurs tactiles capacitifs internes (TOUCH) (GPIO 2,4,15,12,13,14,27,32 et 33).

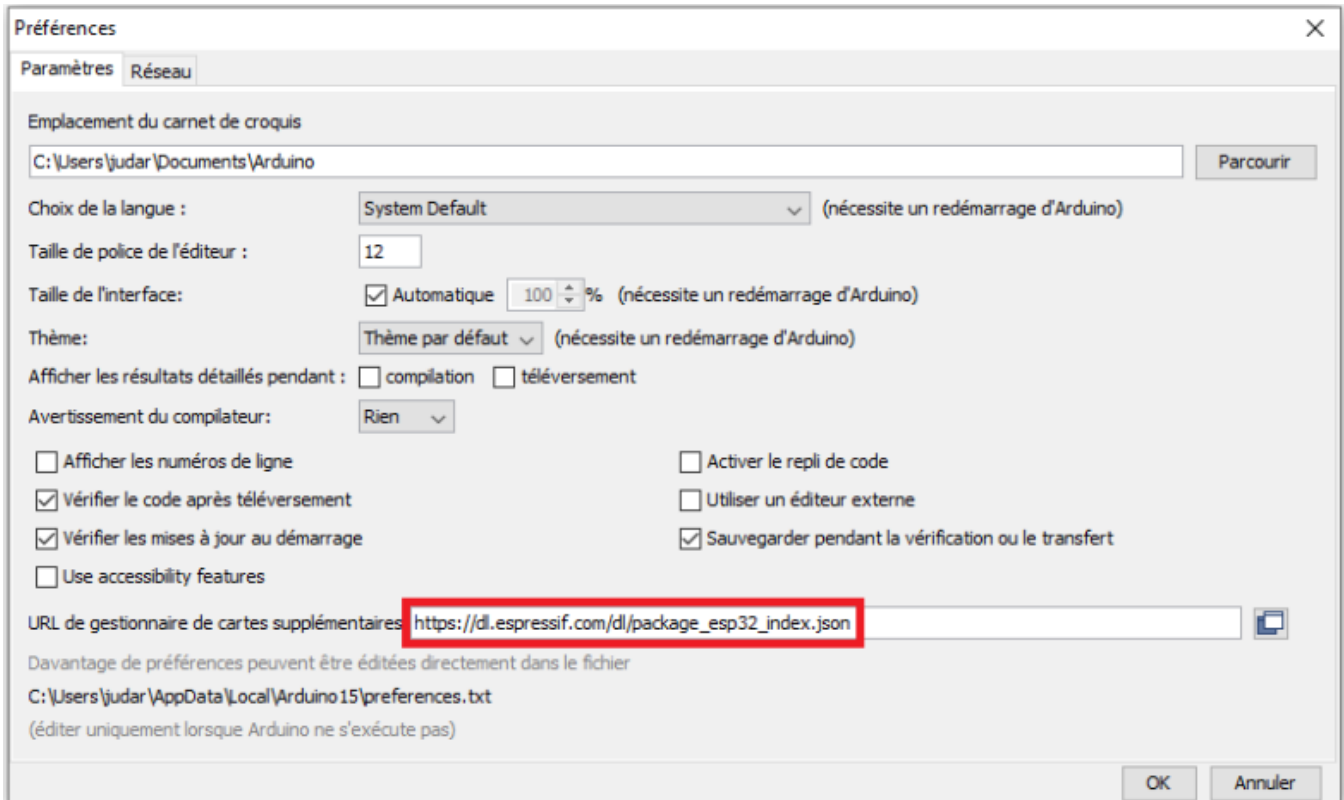
Remarque : ces ports peuvent détecter des variations dans tout ce qui contient une charge électrique, comme la peau humaine. Ils peuvent ainsi détecter les variations induites lors du contact du GPIO avec le doigt. Ces broches peuvent remplacer les boutons mécaniques.

Installation explicite de l'ESP32 dans IDE Arduino

1./ Ajouter la carte ESP32 dans la base des cartes Arduino de l'IDE Dans l'IDE Arduino :

- Ouvrir l'onglet Fichier > Préférences
- Entrer l'adresse suivante dans le champ URL de gestionnaire de cartes supplémentaires :

https://dl.espressif.com/dl/package_esp32_index.json



2/ Installer la carte ESP32 dans l'IDE Arduino

Dans l'IDE Arduino :

1. Ouvrir l'onglet *Outil > Type de carte > Gestionnaire de carte*
2. Rechercher le paquet *ESP32*
3. Installer le paquet : sélectionnez la dernière version disponible et cliquer sur *Installer*



3/ Sélectionner explicitement la carte ESP32

Il ne reste plus qu'à définir la carte ESP32 pour que le programme soit correctement envoyé sur la carte. Dans l'IDE Arduino :

- Ouvrir l'onglet *Outil > Type de carte*
- Choisissez alors la carte ESP32 qui correspond à votre version d'ESP32. Le plus souvent ce sera la carte **ESP32 Dev Module**

