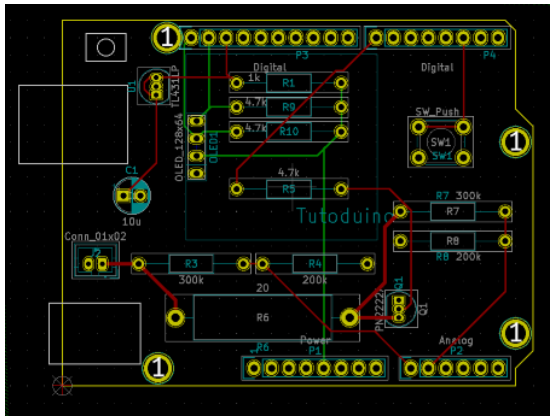
Microcontrôleurs

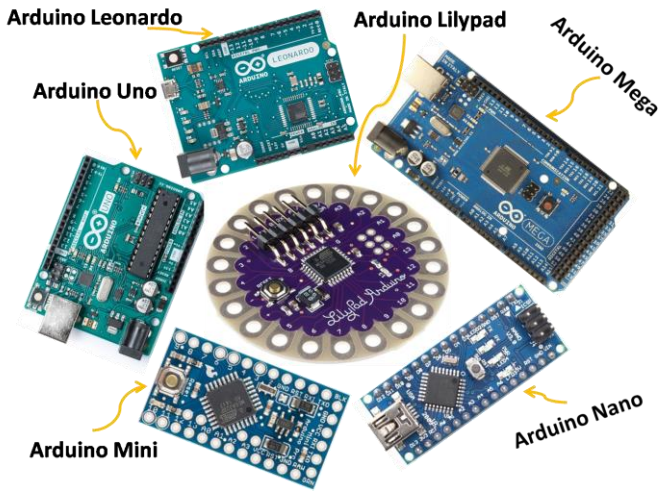
Apprentissage des bases des microcontrôleurs

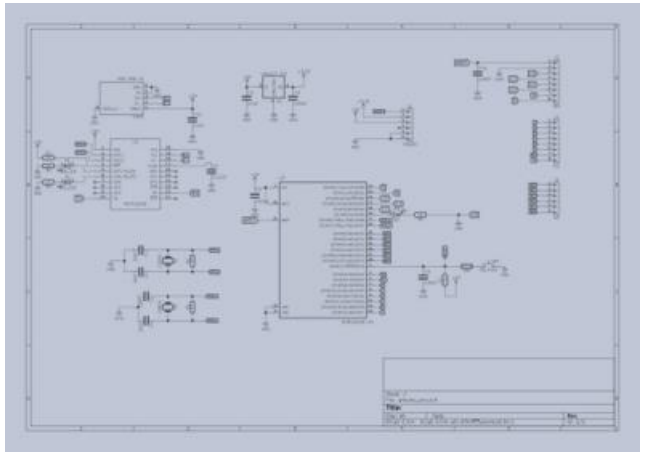
Mise en pratique sur Arduino

Qu’est ce qu’Arduino ?

Atmega328p







Cartes de développement

IDE environnement de développement

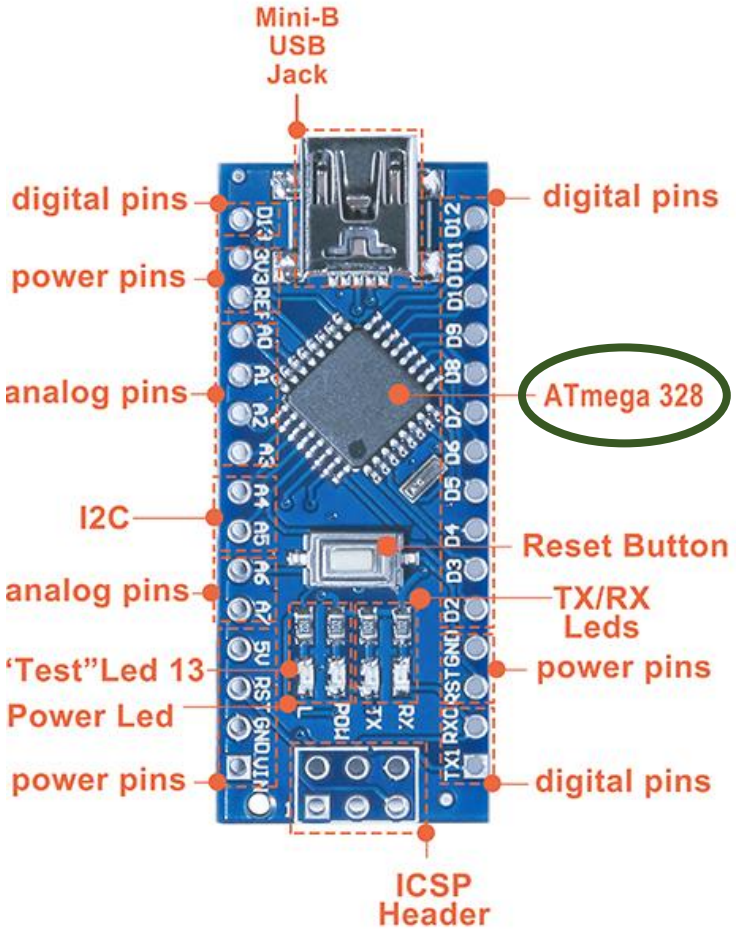
La carte Arduino

Pins de puissance
Pins digitaux
Pins analogiques

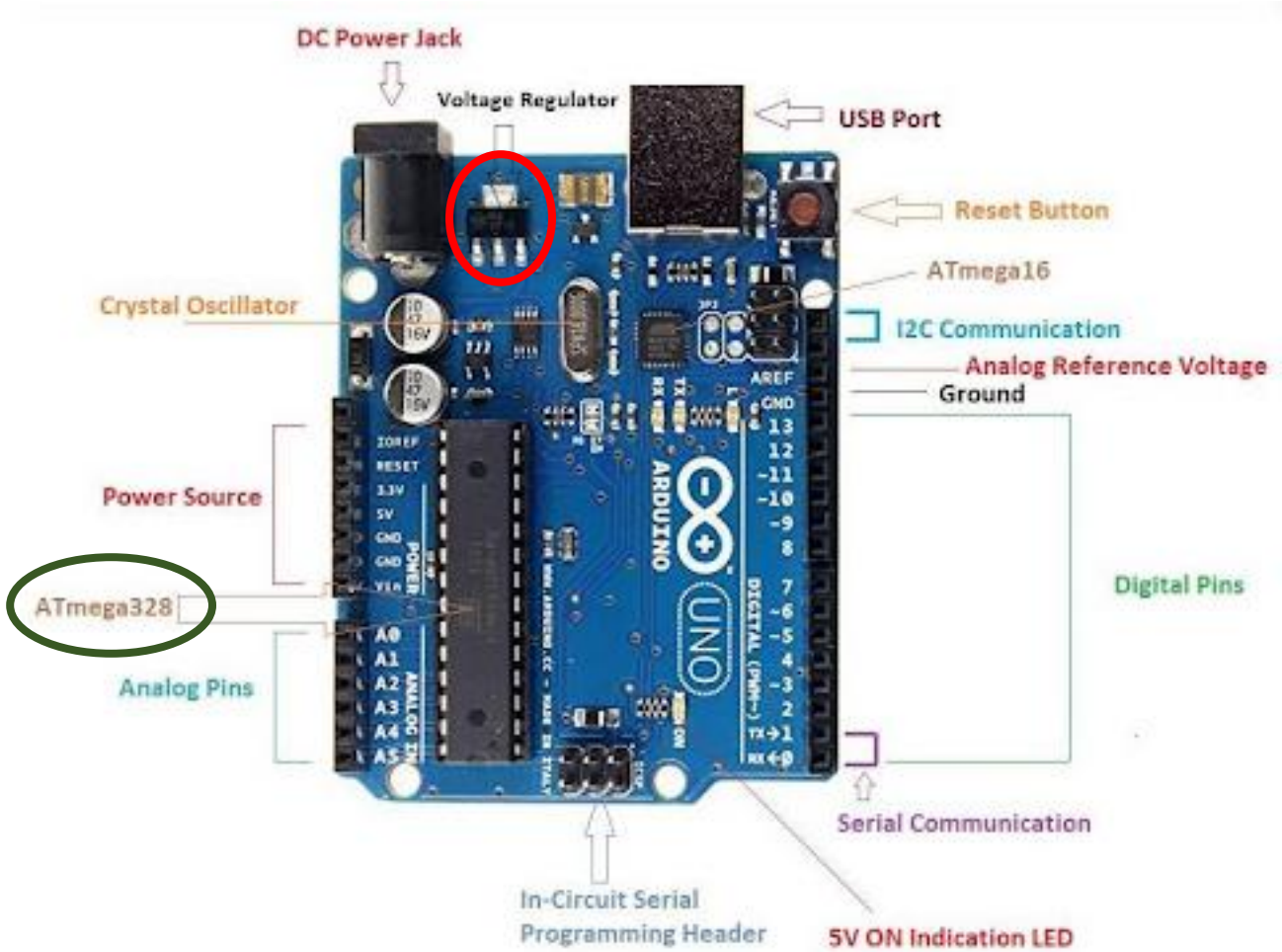
Protocole de communication :
I2C
SPI
Série (TX/RX) …

Type	Microcontrôleur	Avantages
Uno/nano R3	Atmega328P	Très bon support (documentation, librairies, forum, hacks, ...)
Mega	Atmega2560	Plus de GPIO Plus d'entrées analogique et PWM
Leonardo	Atmega32u4	Intègre la communication USB
MKR	ARM Cortex M0	Connectivité (WiFi, LoRa, ...)
Due	ARM Cortex M3	Puissance de calcul + USB

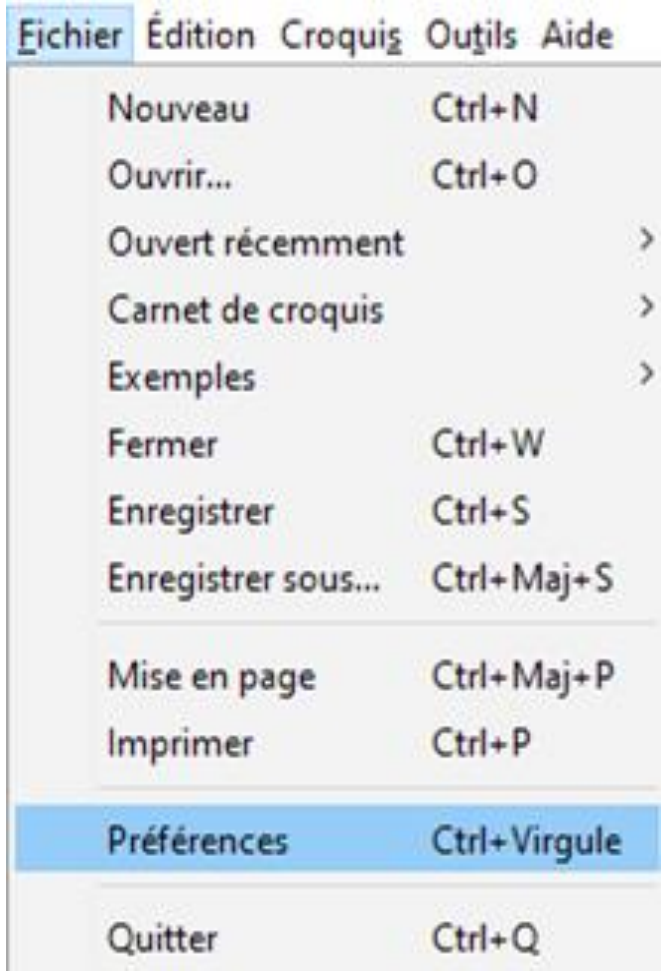
Alimentation de la carte

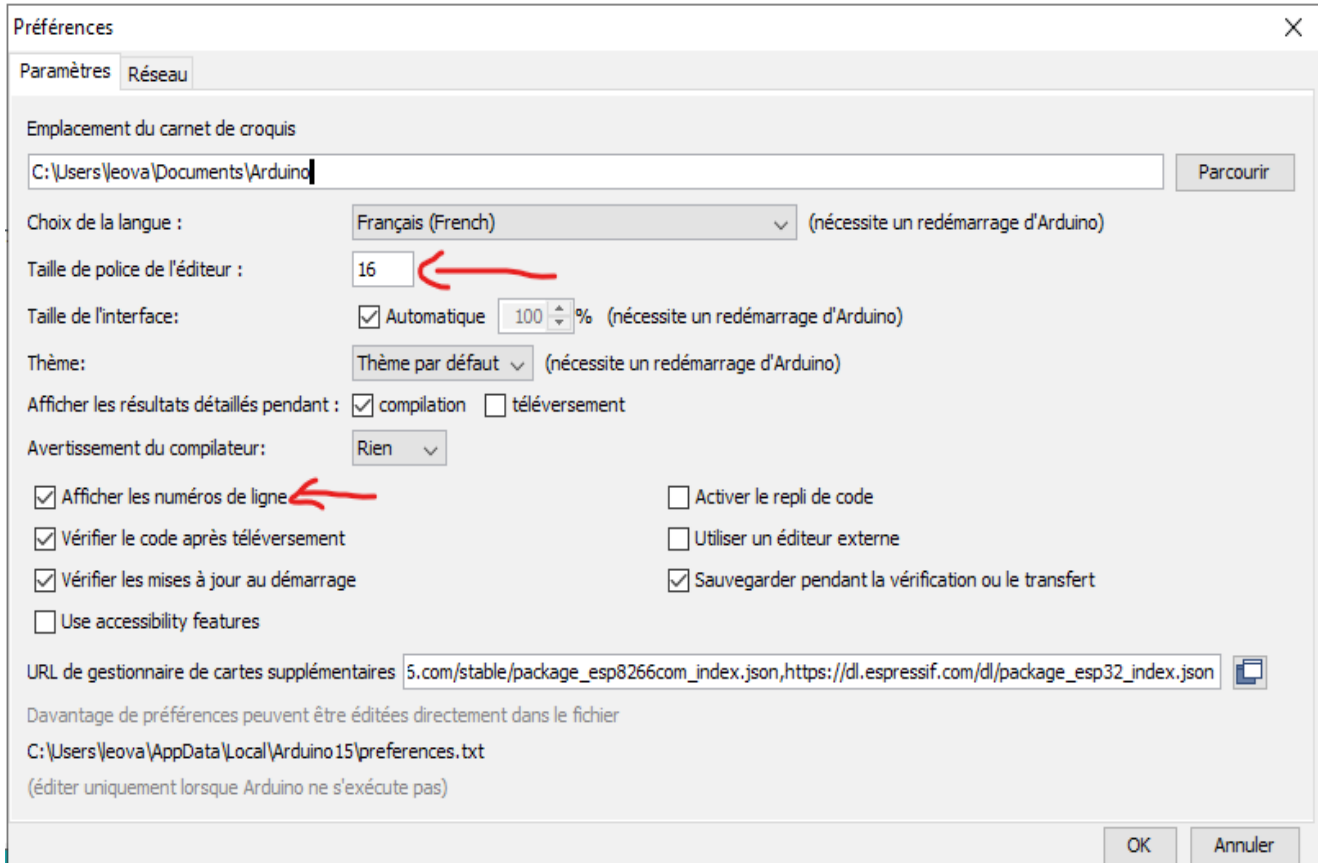


USB (5V)
V_{in} (7-12V)
Jack (7-12V)

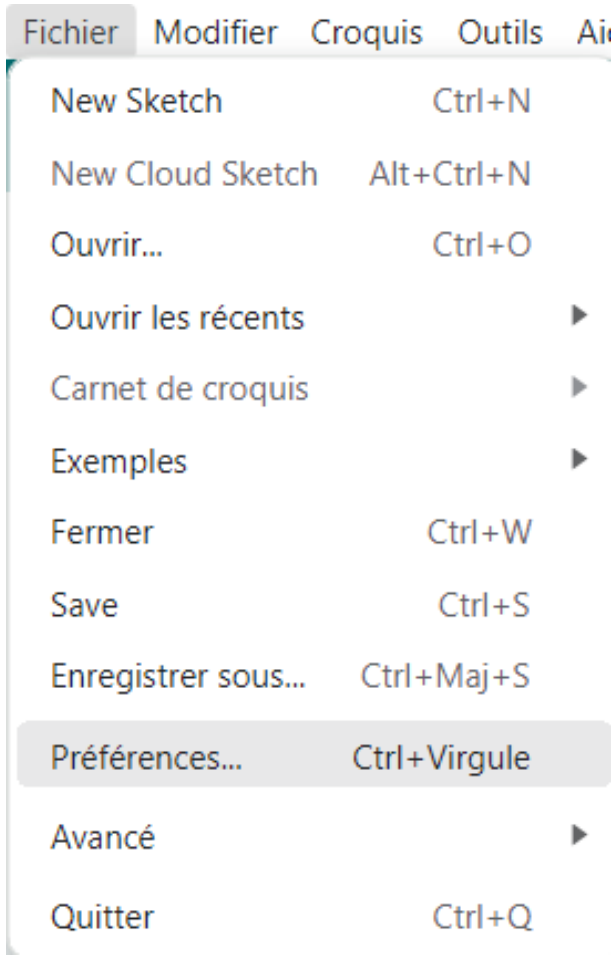


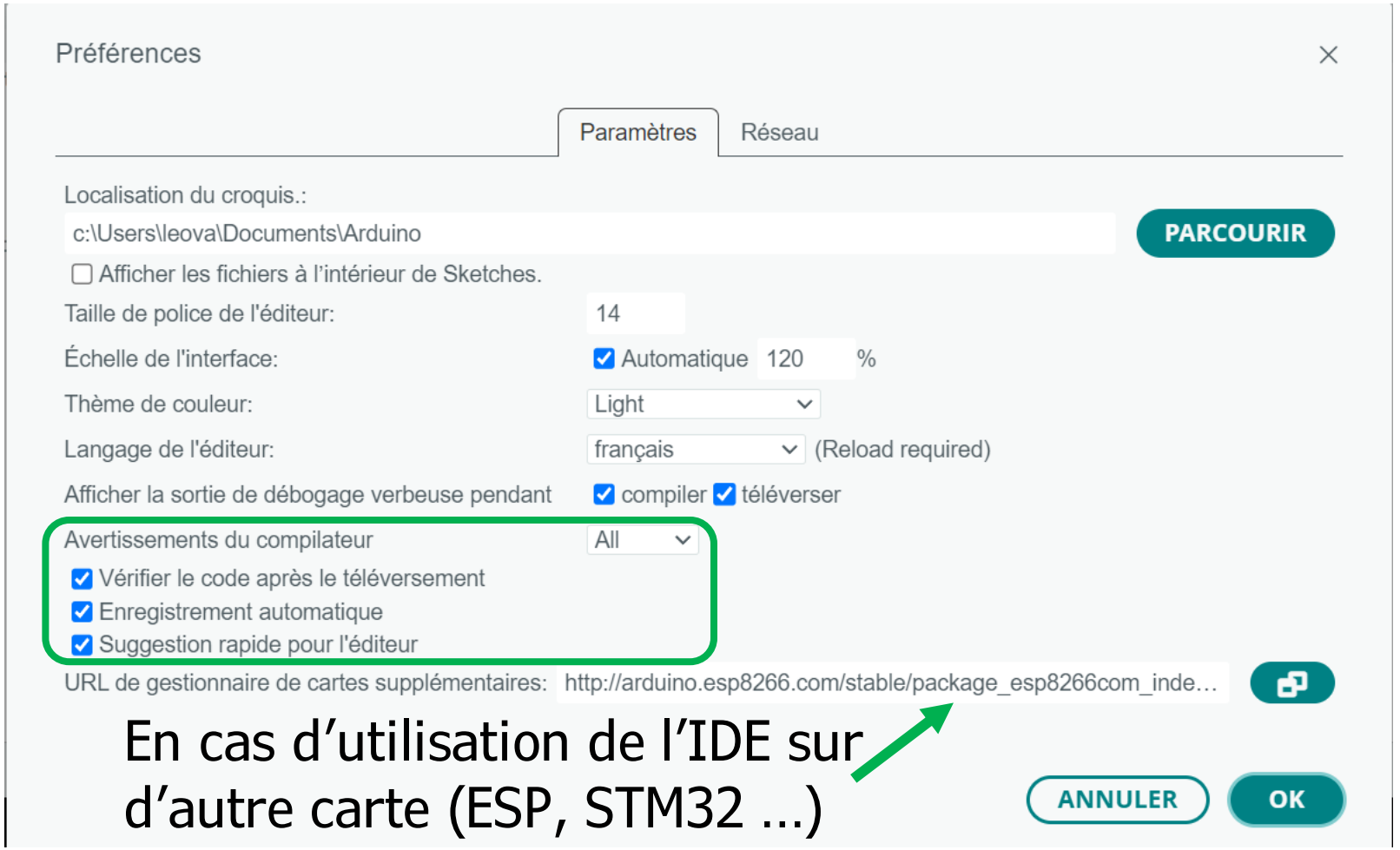
L'IDE l'ancienne version





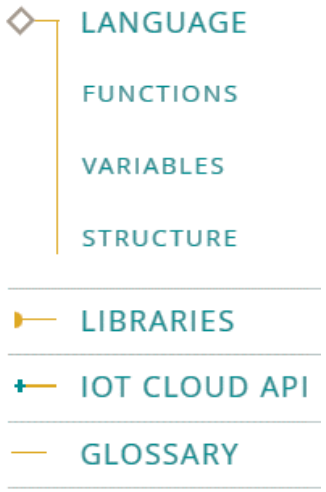
L'IDE la nouvelle version

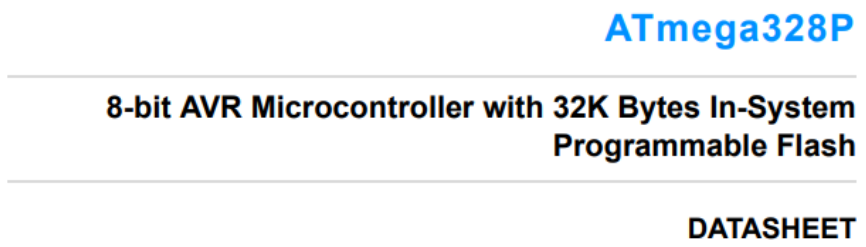




La documentation

Il y a 2 documents très important !





https://www.arduino.cc/reference/en/

Datasheet

Utilisation de l'IDE

Type de carte et port série utilisé

Compiler

Téléverser (compile + téléverse)

Traceur série

Moniteur série

```
1  void setup() {
2    // put your setup code here, to run once:
3
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8
9  }
```

Programmation microcontrôleur

.c .cpp
.ino

Ecriture du programme
C ou C++

Compilation du programme

.bin
.hex

Téléversement vers le µc

Atmega328p

Utilisation d'une
liaison Série (UART)

Les variables

bool / boolean	Sur 1 octet
byte / char	Sur 1 octet (8 bits) Entre 0 et 255
int	Sur 2 octets entre -32 768 et 32 767
long	Sur 4 octets entre -2 147 483 648 et 2 147 483 647
float	Sur 4 octets
double	Sur 8 octets Plus précis que le float
String	Pour les chaine de caractères
int8_t / uint8_t int16_t / uint16_t int32_t / uint32_t	Pour être plus claire dans la déclaration des variables

unsigned → Double la capacité (entier)

const → Variables constantes

volatile → Variables d'interruption

! Négation logique

~ pour la négation

<< ou >> pour le décalage de bit

| ou bit à bit

& et bit à bit

|| ou logique

&& et logique

https://www.arduino.cc/reference/en/

Les fonctions utiles

```
1  //Configure
2  pinMode(byte pin, etat); //etat={OUTPUT;INPUT;INPUT_PULLUP}
3  //Lire
4  analogRead(pin);
5  digitalRead(pin);
6  //Ecrire
7  digitalWrite(pin, etat); // etat={LOW,HIGH} peut être remplacer par 0 ou 1
8  analogWrite(pin, arg); // arg entre 0 et 255
9  //Temps
10 millis(); // renvoie le temps écoulé depuis le début du programme (uint32_t temps)
11 micros(); // idem mais en microsecondes
12 delay(uint32_t temps);
13 delayMicroseconds(uint32_t temps);
```

https://www.arduino.cc/reference/en/

Les fonctions

```
1 void fonction(void){
2   //instructions
3 }
```

```
1 int fonction(void){
2   //instructions
3
4   return 0; // retourne la valeur dans le type défini
5 }
```

```
1 int fonction(int a, int b){
2   int res=0;
3   //instructions faisant intervenir a et b
4   return res;
5 }
```

Structure d'un programme Arduino

```
1  // inclusion des libraries
2  // déclaration des variables globales
3  // déclaration des fonctions
4
5  void setup() {
6    // s'execute au démarrage
7  }
8
9  void loop() {
10   // traitement continu
11 }
```

Moniteur série

```
Serial.begin(9600);
```

A initialiser dans le *setup*
avec une vitesse standard
9600 ou 115200 sont des vitesses standards

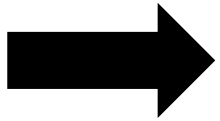
Envoyer dans le moniteur série

```
Serial.print("Bonjour ");
Serial.println("tout le monde");
```

Exemple de programme et de quelques *macro*

```
1   void setup() {
2     Serial.begin(9600);
3     while (!Serial) {
4       ;  // wait for serial port to connect.
5     }
6     Serial.println(__FILE__);
7     Serial.println(__DATE__);
8     Serial.println(__TIME__);
9     Serial.println(__LINE__);
10  }
```

Pratique

Moniteur série *UART Tx/Rx (Pin 1 et 0)*  Debug

Utiliser les fonctions présentes dans les Références Arduino et faire le programme suivant :

Incrémenter un compteur toutes les 500 millisecondes.
Afficher sa valeur dans le moniteur Série lors de son incrémentation.

Variantes / Bonus :

Utiliser une variable entière sur 8 bits, que remarquez-vous ?

Utiliser le traceur série

Traceur série

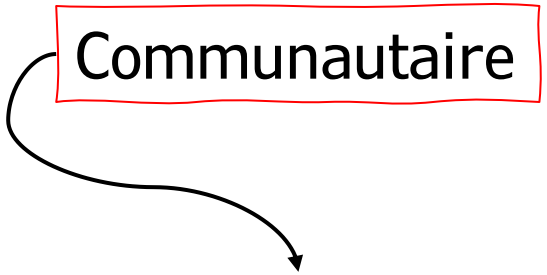
Format .csv :

valeur1,valeur2
valeur1,valeur2
…

```
1   int8_t compteurS = 0;
2   uint8_t compteurU = 0;
3
4   void setup() {
5     Serial.begin(9600);
6     while (!Serial) {
7       ;  // wait for serial port to connect.
8     }
9     while(millis() < 5000){
10      Serial.print(compteurS);
11      Serial.print(",");
12      Serial.println(compteurU);
13      compteurS++;
14      compteurU++;
15    }
16  }
```

Incrément que pendant les 5 premières secondes du programme

Bibliothèque Arduino



Multitude de bibliothèques et de forum d'aide

Plusieurs exemples d'utilisation !

https://www.arduino.cc/reference/en/libraries/

Bibliothèque - Servo

Servo

Device Control

Allows Arduino boards to control a variety of servo motors.

This library can control a great number of servos. It makes careful use of timers: the library can control 12 servos using only 1 timer. On the Arduino Due you can control up to 60 servos.

Exemples :

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = A0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  val = analogRead(potpin);            // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it for use with the servo (value between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
}
```

Bibliothèque - Stepper

Stepper

Device Control

Allows Arduino boards to control a variety of stepper motors.
This library allows you to control unipolar or bipolar stepper motors. To use it you will need a stepper motor, and the appropriate hardware to control it.

Go to repository

Compatibility

This library is compatible with **all** architectures so you should be able to use it on all the Arduino boards.

Bibliothèque – Stepper

Usage

This library allows you to control unipolar or bipolar stepper motors. To use it you will need a stepper motor, and the appropriate hardware to control it.

To use this library:

```
#include <Stepper.h>
```

Circuits

- Unipolar steppers.
- Bipolar steppers.

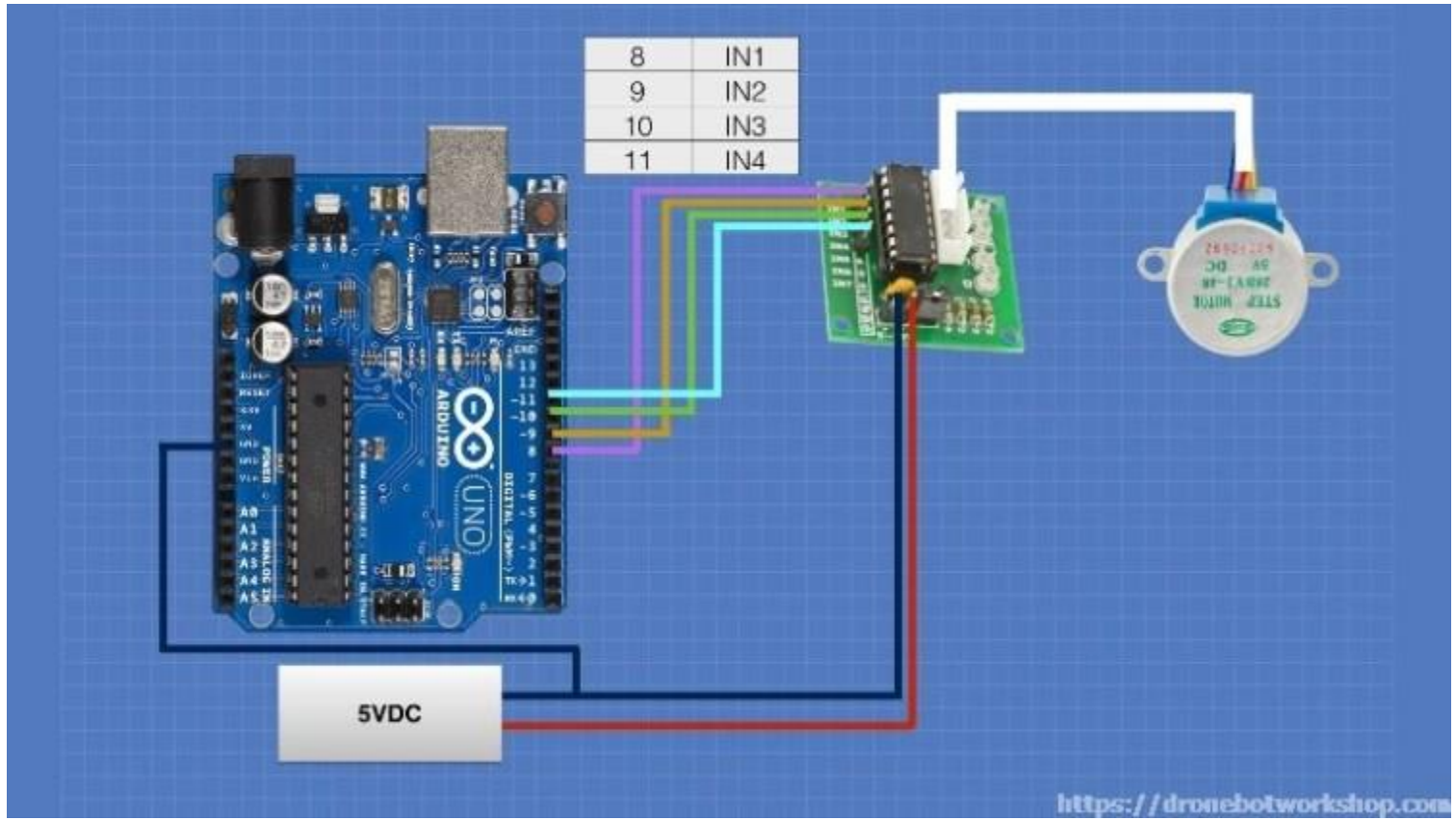
Examples

- Motor knob: Control a highly accurate stepper motor using a potentiometer.
- Stepper one revolution: Turn the shaft one revolution clockwise and one counterclockwise.
- Stepper one step at a time: Turn the shaft step by step to check the proper wiring of the motor.
- Stepper speed control: Control the stepping speed with a potentiometer.

Methods

- stepper()
- setSpeed()
- step()

Pratique - Stepper



En utilisant les exemples et la documentation :
Faire tourner le moteur pas à pas de 500 pas toutes les 2 secondes, dans un sens puis dans l'autre

Interface série

Utiliser les commandes de *Serial* pour envoyer le nombre de *step* à faire directement depuis le moniteur série

+ rien
+ ‘’ \n ’’
+ ‘’ \t ’’
+ ‘’ \t\n ’’

https://www.arduino.cc/reference/en/language/functions/communication/serial/

Potentiomètre

Pour lire une tension, quelque chose qui n'est pas numérisé (autre chose que des 0 et des 1), un ADC (Analog to Dital Converter) est utilisé.
Il y en a un de 10 bits disponibles sur les pin Analogiques :
valeur entre 0 et 1023 (10 bits)

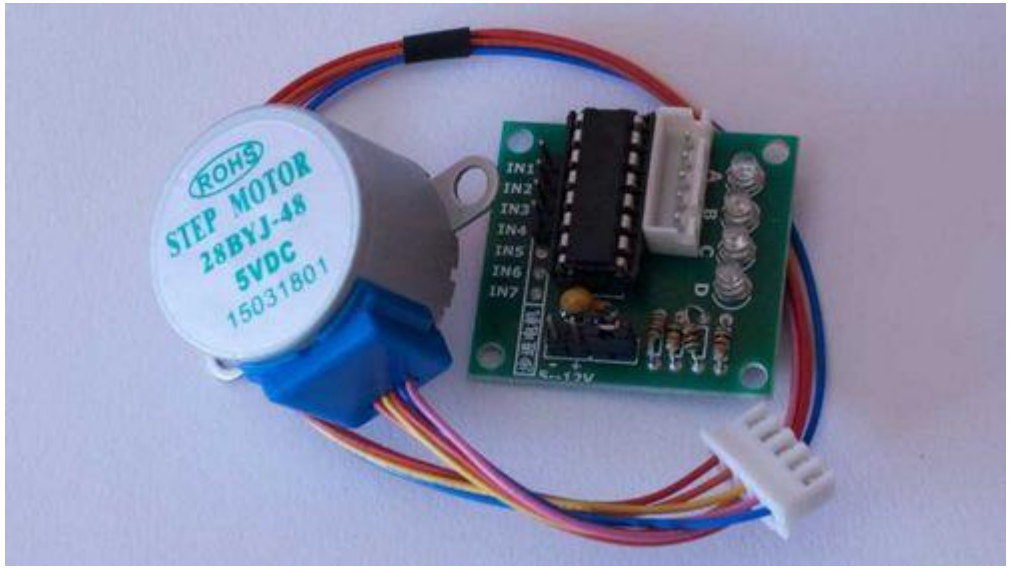
Lecture d'un potentiomètre :
Exemple > Basic > ReadAnalogVoltage

Affichage sur le traceur série

Potentiomètre + Stepper

On a une autre méthode d’interagir avec le microcontrôleur, le potentiomètre !

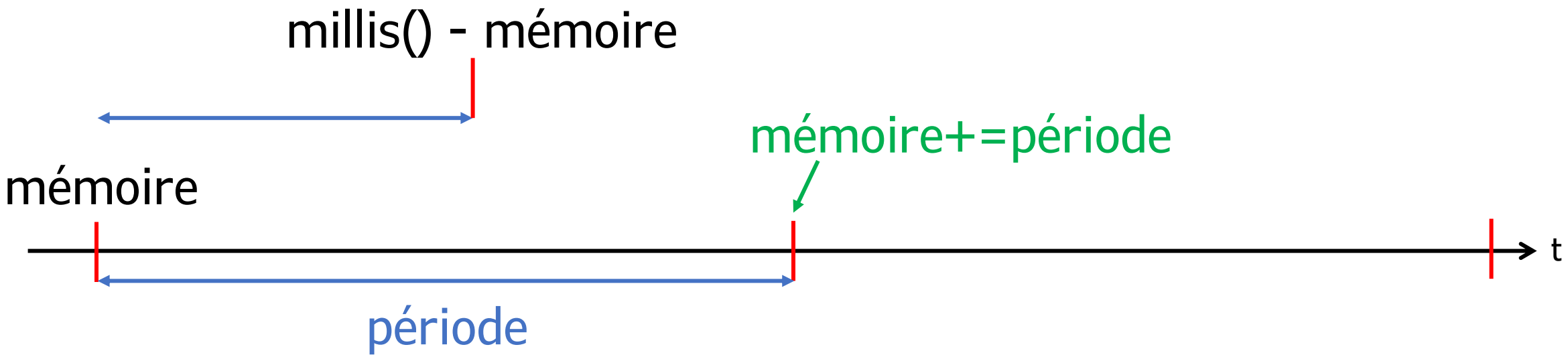
Si nous utilisions ce capteur pour donner des ordres au moteur pas à pas
Allez à la pratique !



Traitement pseudo multitâche

Gain en performance et en rapidité

millis(); → envoie le temps écoulé depuis le début du programme



Traitement pseudo multitâche

Version subissant un potentiel décalage

```
1   uint32_t t_ref;
2   uint32_t periode = 1000;
3
4   void setup() {
5     Serial.begin(9600);
6     while (!Serial) {
7       ;  // wait for serial port to connect.
8     }
9     delay(2000); // Processus d'initialisation
10
11    t_ref = millis(); // capture du temps après le setup
12  }
13
14  void loop() {
15      if(millis() - t_ref >= periode){
16      Serial.println(millis()%periode); // vérification
17
18      t_ref = millis();
19    }
20    // d'autres tâches sont possibles ou des vérifications durant les temps "d'inactivités"
21  }
```

Traitement pseudo multitâche

```
1   uint32_t t_ref;
2   uint32_t periode = 1000;
3
4   void setup() {
5     Serial.begin(9600);
6     while (!Serial) {
7       ;  // wait for serial port to connect.
8     }
9     delay(2000); // Processus d'initialisation
10
11    t_ref = millis(); // capture du temps après le setup
12  }
13
14  void loop() {
15      if(millis() - t_ref >= periode){
16      Serial.println(millis()%periode); // vérification
17
18      t_ref += periode;
19    }
20    // d'autres tâches sont possibles ou des vérifications durant les temps "d'inactivités"
21  }
```

Bibliothèque – Ticker

Ticker

Timing

A library for creating Tickers which can call repeating functions. Replaces delay() with non-blocking functions.
The Arduino Ticker Library allows you to create easily Ticker callbacks, which can call a function in a predetermined interval. You can change the number of repeats of the callbacks, if repeats is 0 the ticker runs in endless mode. Works like a "thread", where a secondary function will run when necessary. The library use no interupts of the hardware timers and works with the micros() / millis() function. You are not (really) limited in the number of Tickers.

Author: Stefan Staub

Compatibility

This library is compatible with **all** architectures so you should be able to use it on all the Arduino boards.

Ticker

GitHub ? Qu'est-ce ?
https://github.com/sstaub/Ticker

Détail de l'exemple :

```
Ticker timer1(printMessage, 0, 1); // once, immediately
Ticker timer2(printCounter, 1000, 0, MILLIS); // internal resolution is milli seconds
Ticker timer3(printCountdown, 1000, 5); // 5 times, every second
Ticker timer4(blink, 500); // changing led every 500ms
Ticker timer5(printCountUS, 100, 0, MICROS_MICROS); // the interval time is 100us and the internal reso
```

Questions – Approfondissement

Présentation de quelques projets plus conséquent

La différence avec ce qu'on a pu faire réside dans la manière de programmer, fonctions, objet, bibliothèques utilisées …

Des questions ?

Supplément

Quelques capteurs et modules intéressant à connaitre :

- ESP01 / NRF24L01 module WiFi
- HC-05 / HC-06 module Bluetooth
- Capteur effet hall (détection aimant)
- Codeur incrémental (interface avec l'Humain)
- Potentiomètre (interface avec l'Humain)
- Relais (commutation commandée)
- Driver moteur DC (L298N, L293D); stepper (A4988 …); brushless (ESC …)
- LED IR (comme les télécommandes)
- Module carte SD
- Capteur température / humidité …
- …