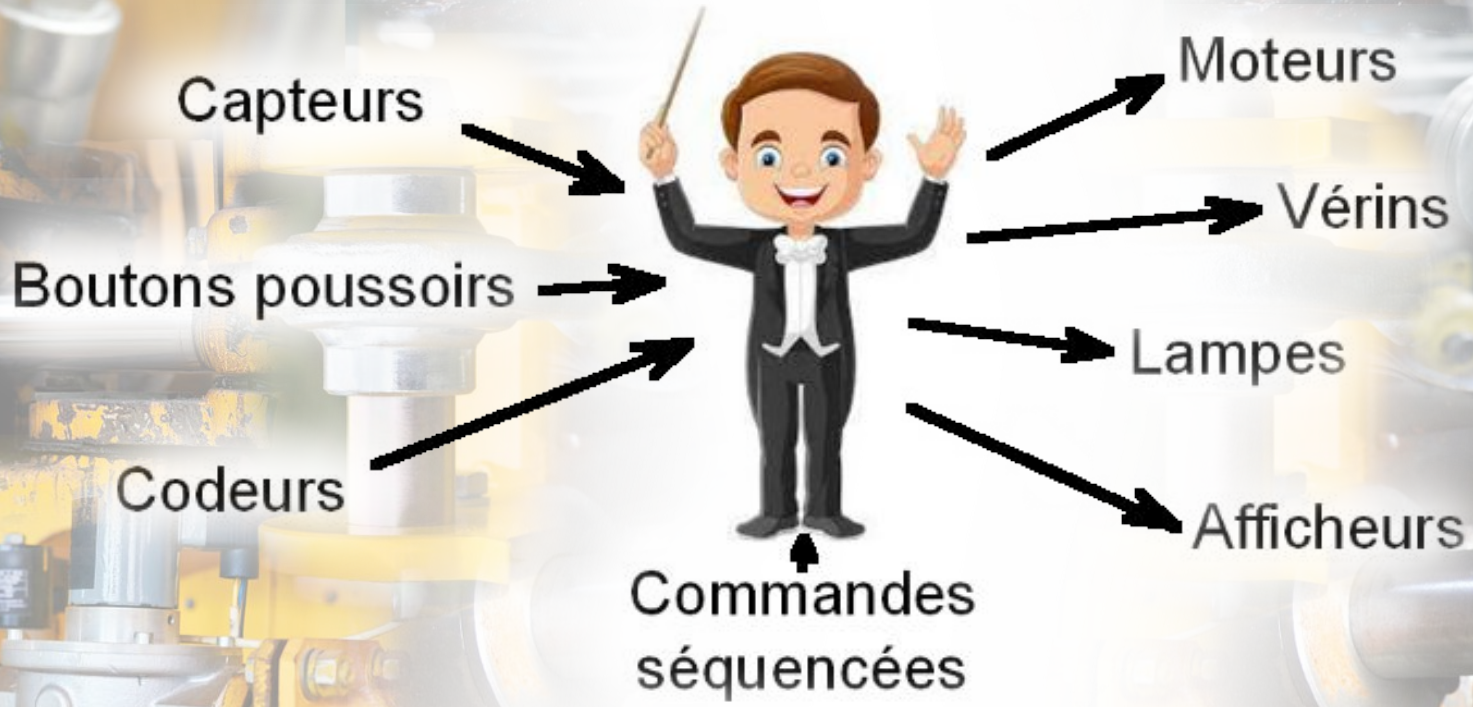
IDROLIK
AUTOMATISATION HYDRAULIQUE

Arduino - premier contact



IDROLIK
AUTOMATISATION HYDRAULIQUE

Sommaire

IDROLIK
AUTOMATISATION HYDRAULIQUE

Information préliminaire

- Cette présentation est accessible via le site idrolik.com
  - rubrique documentation
- Elle sera mise à jour sur le site au fur et à mesure qu'elle évoluera
- De cette manière, vous aurez accès aux exemples de programmes pour vérifier que ça marche

Introduction

- Aujourd'hui on a tous envie d'automatiser certaines tâches
  - Domotique, alimentation du chat en absence, cibles pour le tir au pistolet, poêle à granulés, machine à café ...
- Deux principales classes de circuits électroniques sont conçus pour faire tourner des programmes informatiques
  - Les microprocesseurs et les microcontrôleurs
- Le microprocesseur est taillé pour réaliser des traitements lourds
  - En termes de quantité de calculs
  - En termes de quantité de mémoire nécessaire
  - En termes de volumes de données manipulées (disques, cartes graphiques, réseaux, ...)
  - En termes de qualité d'interface homme-machine (affichage, souris, pad ...)
- Le discret microcontrôleur est taillé pour une action efficace au plus près du matériel

Vocabulaire – bit & octet

- **Un bit est l'information binaire minimale**
  - Est égal à 0 ou à 1
  - En électronique
    - 0 correspond souvent à une tension électrique nulle (ou proche de zéro volt)
    - 1 correspond souvent à une tension de 5 volts (ou proche de 5 volts)
- **Un octet est un ensemble ordonné de 8 bits**
  - C'est la quantité de base pour la mémorisation et le traitement d'informations numériques
  - On parle ...
    de kilo octets (kilo = 10^3),
    de méga octets (mega = 10^6),
    de giga octets (giga = 10^9)
    de tera octets (tera = 10^{12})
- **Dans un octet, on parle de …**
  - Bit de poids faible (bit de droite)
  - Bit de poids fort (bit de gauche)
  - Un octet représente un nombre écrit en base 2 (deux signes 0 et 1)
    Il peut prendre n'importe quelle valeur entière entre 0 et 255
    $01101010_{base\ 2} = 2^1 + 2^3 + 2^5 + 2^6 = 106$

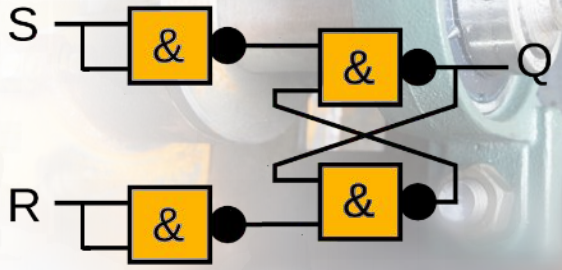
Vocabulaire – mémoire

- Imaginez que vous posiez 8 verres alignés sur une table, tantôt couchés, tantôt debout
  - En faisant ça, vous venez de créer une mémoire d'un octet
- Vous partez et vous revenez un an après
  - Si personne n'a modifié l'état de votre mémoire, vous retrouvez votre octet comme vous l'avez laissé
  - Dit autrement, votre octet est resté en mémoire
- En électronique, on parle de :
  - ROM : Read Only Memory (pour mémoriser un octet en ROM, on peut 8 fils à 5 volts et dont certains sont coupés pour faire des zéros)
  - RAM : Random Access Memory (pour mémoriser un octet en mémoire vive, on peut imaginer commander 8 bascules électroniques dites « RS »)
- Pour un microcontrôleur, on parle de kilo octets

Bascule RS :
S et R à zéro
S passe à 1, Q passe à 1
S repasse à zéro, Q reste à 1
R passe à 1, Q passe à zéro
R repasse à zéro, Q reste à zéro

Cellule « & » :
2 entrées à 1 ► sortie à 1
Sinon, sortie à zéro

« ● » :
Inversion de la sortie



Vocabulaire – liaison série

- **Principe de la liaison série**
  - Un circuit électronique émetteur impose le niveau de tension (0 ou 1) sur un fil à chaque top d'horloge
  - On parle de fréquence d'horloge
  - Le circuit récepteur lit à la même fréquence le niveau de tension sur le fil
  - Les circuits émetteur et récepteur peuvent avoir chacun leur horloge (oscillateur à quartz) qu'ils règlent à la même fréquence
  - Ou la fréquence d'écriture des bits sur la ligne peut être partagée par l'émetteur sur un fil du bus
- **Pour une liaison radio, c'est pareil …**
  - Les niveaux de tension sur un fil deviennent des trains d'ondes radio émis en série dans l'air

- **Lorsqu'on veut envoyer des informations complexes, on constitue des trames**
  - Une trame est souvent constituée par un grand nombre d'octets
  - Chaque trame commence par une en-tête qui indique (entre autres) l'adresse du destinataire, et celle de l'émetteur (ce qui permet au destinataire de répondre)
  - Chaque bit de chaque octet de la trame est « émis en série »

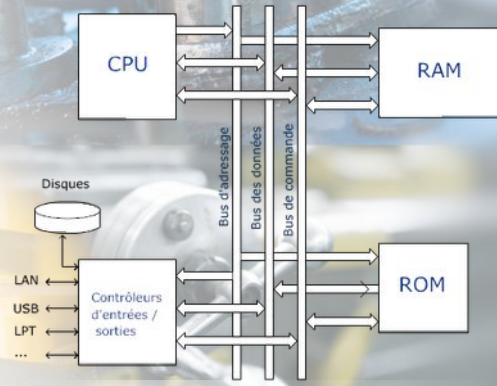
Vocabulaire – le bus

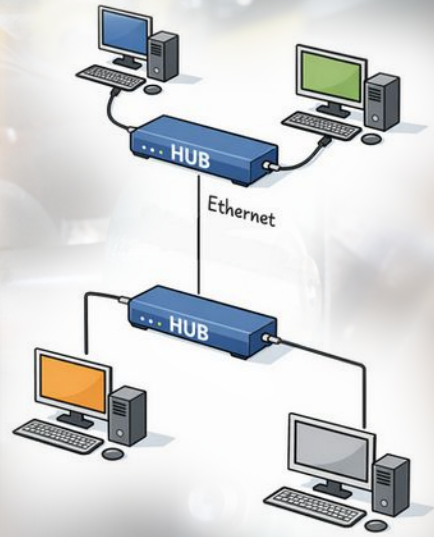
- Imaginons des prisonniers, chacun dans sa cellule ...
  - Ils ne peuvent pas se voir, ils ne peuvent pas s'entendre
  - Les cellules sont reliées entre elles par un réseau de radiateurs connectés par des tuyauteries métalliques
  - Chacun peut taper sur son radiateur pour émettre des messages par exemple avec le code morse

  - Tous les prisonniers entendent lorsque l'un d'entre eux frappe sur son radiateur
  - Si un prisonnier veut communiquer avec un autre, il fait précéder son message du numéro de la cellule du destinataire et de son propre numéro de cellule.
    Seul le destinataire prend le message pour lui, les autres l'ignorent (si on veut de la discrétion, le corps des messages peut être chiffré)
- Analogie : ici, le réseau de radiateurs constitue un bus de données, toutes les informations transitent par ce canal
  - Arbitrage à la mode Ethernet :
    - Règle 1 : si quelqu'un est en train de parler, les autres se taisent
    - Règle 2 : si deux personnes prennent la parole en même temps, le message est incompréhensible. Tout le monde se tait et initialise un temps d'attente aléatoire avant de reprendre la parole ... et la règle 1 s'applique
  - Arbitrage à la mode USB : le maître de cérémonie distribue la parole, les autres ne parlent que lorsqu'ils y sont invités

Vocabulaire - Le bus

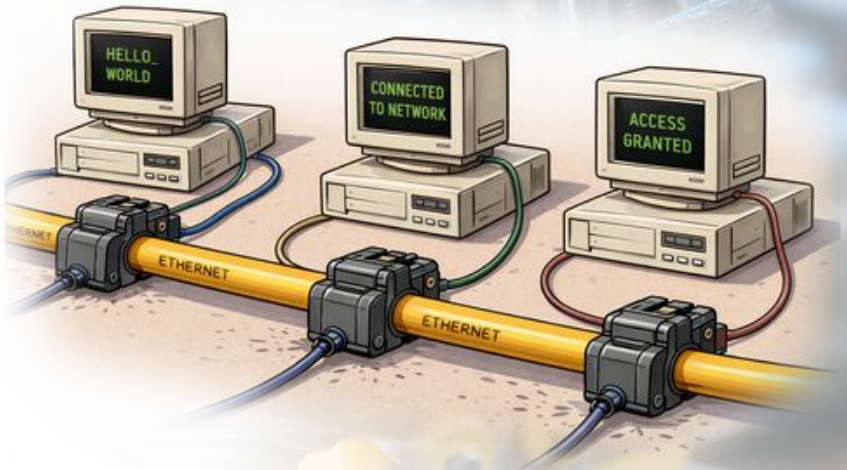
- Dans la suite, le mot « bus » est souvent utilisé
- En informatique, un bus ...
  - Est une liaison sur laquelle on peut connecter plusieurs circuits électroniques
    - Sur une carte électronique, le bus peut prendre la forme de plusieurs conducteurs électriques en parallèle sur lesquels sont directement connectés les interfaces des périphériques.
      - Si un circuit « parle » sur le bus (= impose les niveaux de tensions sur les conducteurs du bus), les autres circuits « se taisent » (= se contentent de « lire » les niveaux de tension)
      - Les périphériques ne prennent en compte que ce qui leur est adressé
  - Elle peut être constituée des liaisons point-à-point reliées en étoile(s) par des nœuds actifs (hubs ou switches)
    - Un format spécifique de connecteurs matériels est souvent prévu pour éviter qu'un circuit non compatible puisse être connecté et brouille les communications sur le bus.
  - Est un « lieu » où tous les dispositifs électroniques connectés peuvent être destinataires de messages
  - Est associé à un protocole de communication dédié
    - Le protocole définit et organise les échanges, avec un système d'adressage qui permet d'indiquer qui parle et qui est destinataire
- Par opposition au bus, on parle de liaisons point à point.
  Avec une liaison point-à-point, pas besoin de dire à qui s'adresse le message

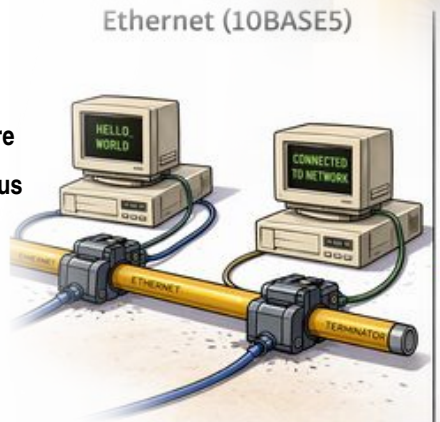




Exemple 1 – ETHERNET

- **A l'origine Ethernet était un protocole qui utilisait un câble coaxial unique**
  - Toutes les machines étaient connectées dessus avec des prises vampires
  - Adresses Ethernet uniques sur 48 bits (6 octets)
    Une adresse « broadcast » indiquait que toutes les machines étaient destinataires (DHCP – Dynamic Host Configuration Protocol))
  - Protocole basé sur des trames avec en-tête indiquant (entre autres) les adresses source et destination
  - Les machines émettaient leurs trames sur le coaxial
  - Lorsque deux machines émettaient en même temps, détection de la collision par l'ensemble des machines
    - Les machines stoppaient leur émission et initialisaient une temporisation aléatoire
    - La première machine qui émettait occupait le bus, les autres attendaient que le bus se libère
  - Des trames envoyées en « broadcast » permettaient
- **Le câble coaxial constituait un bus**
  - Par certains cotés, le WIFI (qui a son protocole spécifique) ressemble à Ethernet coaxial



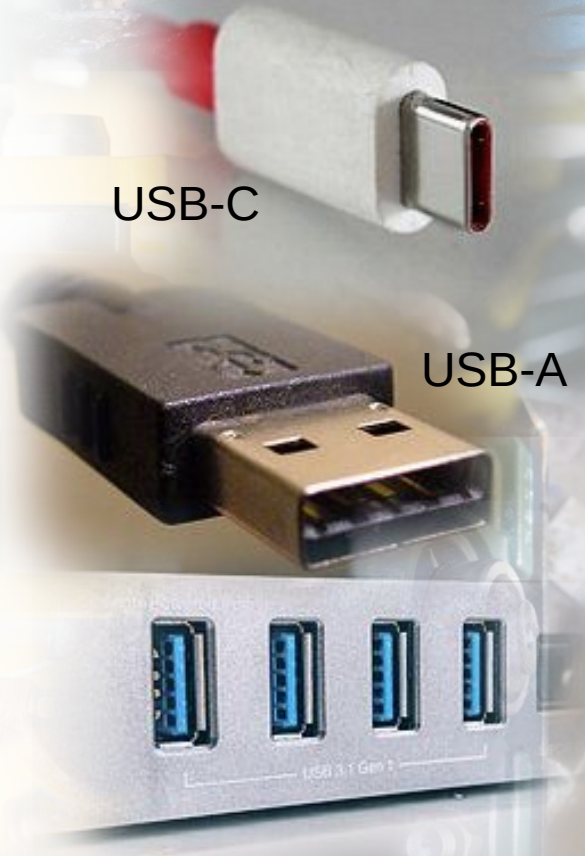


Exemple 1 - ETHERNET

- Aujourd'hui, Ethernet s'est modernisé
  - 4 paires de fils « full duplex » (transmission simultanée dans les deux sens sur chaque paire)
- Propagation des trames via des « switches » qui ...
  - transmettent vers les autres ports les trames reçues sur un port
  - apprennent de quel port arrivent les trames de chaque adresse Ethernet pour faire suivre les trames vers les ports pertinents
- Envoi toujours possible de trames « broadcast » (= à destination de tous les dispositifs présents sur le réseau local)
  - Recherche serveur(s) DHCP (Dynamic Host Configuration Protocol)
  - Recherche d'imprimante(s), de scanner(s) ...

Exemple 2 - USB

- Clavier, souris, disque dur, clé USB (Universal Serial Bus) ...
- 4 conducteurs (hors alim du périphérique)
  - 2 conducteurs D+ et D- pour transmettre les données en mode série (un sens à la fois)
  - 2 conducteur VCC et GND (tension alimentation et masse)
- Adresse USB sur 7 bits (127 périphériques maxi sur un bus USB)
- Protocole basé sur des liaisons série suivant le principe maître-esclave (1 maître, plusieurs esclaves)
  - Un esclave ne parle jamais spontanément, même pour se présenter. Sa présence est détectée à la connexion par une résistance pull-up sur une ligne de donnée
- Les hubs USB (si présents) sont interrogés périodiquement par le maître pour connaître les éventuels nouveaux périphériques connectés derrière



IDROLIK
AUTOMATISATION HYDRAULIQUE

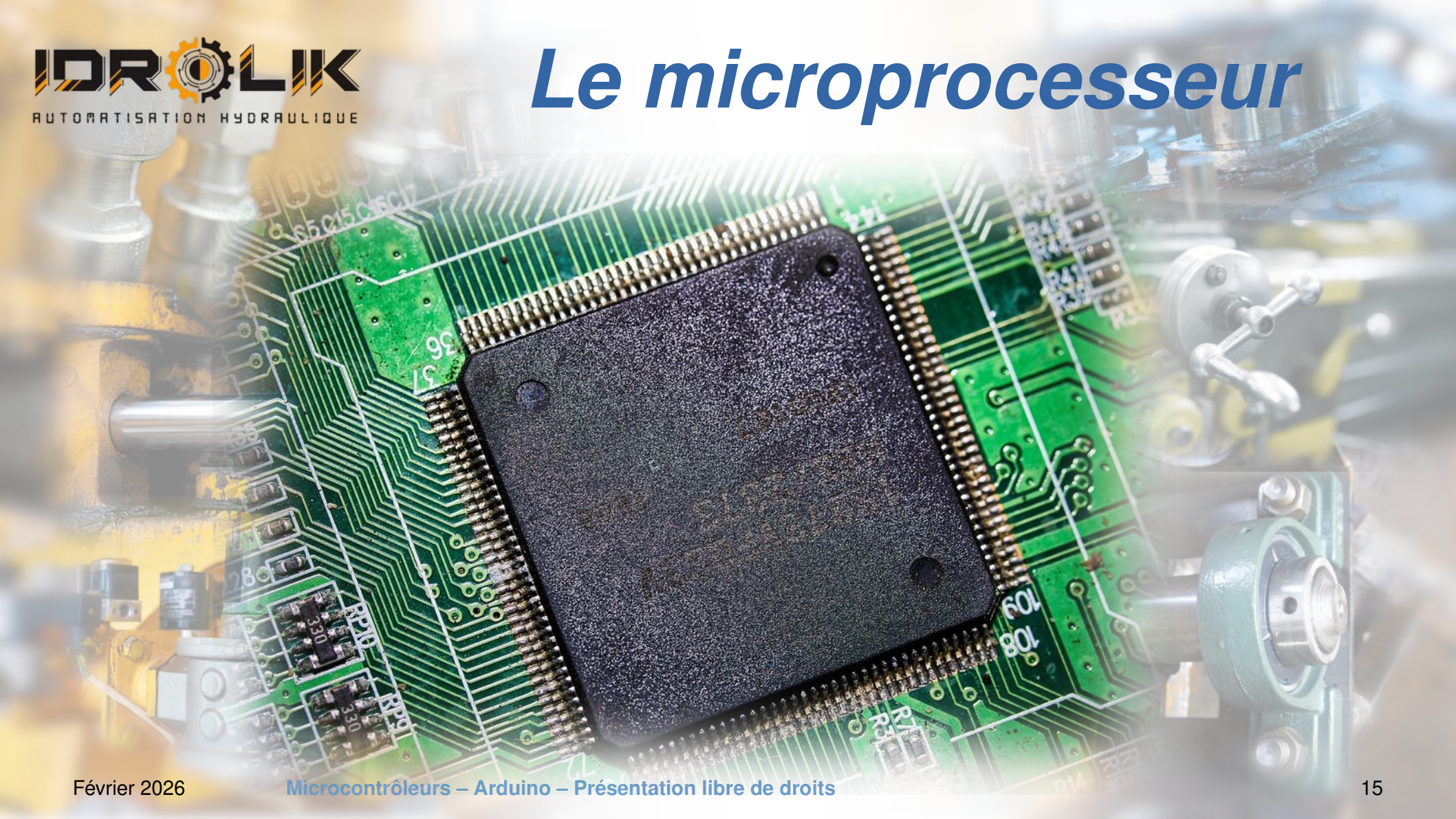
1 - Microprocesseur versus microcontrôleur

Le microprocesseur

- Est plus complexe (nombre de transistors et structure interne) et plus performant que le microcontrôleur
  - Rapidité, capacité de calcul (coprocesseur arithmétique intégré), quantité de mémoire vive adressable
- Chauffe davantage, il a besoin d'un système de refroidissement (ventilateur et/ou circuit d'eau)
- Est l'élément central d'une carte mère, il s'interface matériellement :
  - Avec des circuits de mémoire via des bus d'adresse et de donnée
  - Avec des périphériques via des bus d'entrées-sorties
    - Carte graphique, disques durs, clavier, souris, Ethernet, Wifi...
- Fonctionne avec :
  - Un BIOS (Basic Input Output System) stocké dans le microprocesseur lui-même (firmware)
    - permet l'utilisation du disque dur afin de charger le système d'exploitation dans la mémoire du microprocesseur (bus SATA ...)
  - Un système d'exploitation qui souvent stocké sur un stockage de masse (disque dur...) et qui prend en charge
    - Le fonctionnement en multitâche
    - La gestion des périphériques (clavier, souris, imprimantes, écrans, scanners …)
    - Il fournit les interfaces systèmes pour les logiciels applicatifs (affichage, Internet, ...)

IDROLIK
AUTOMATISATION HYDRAULIQUE

Le microprocesseur

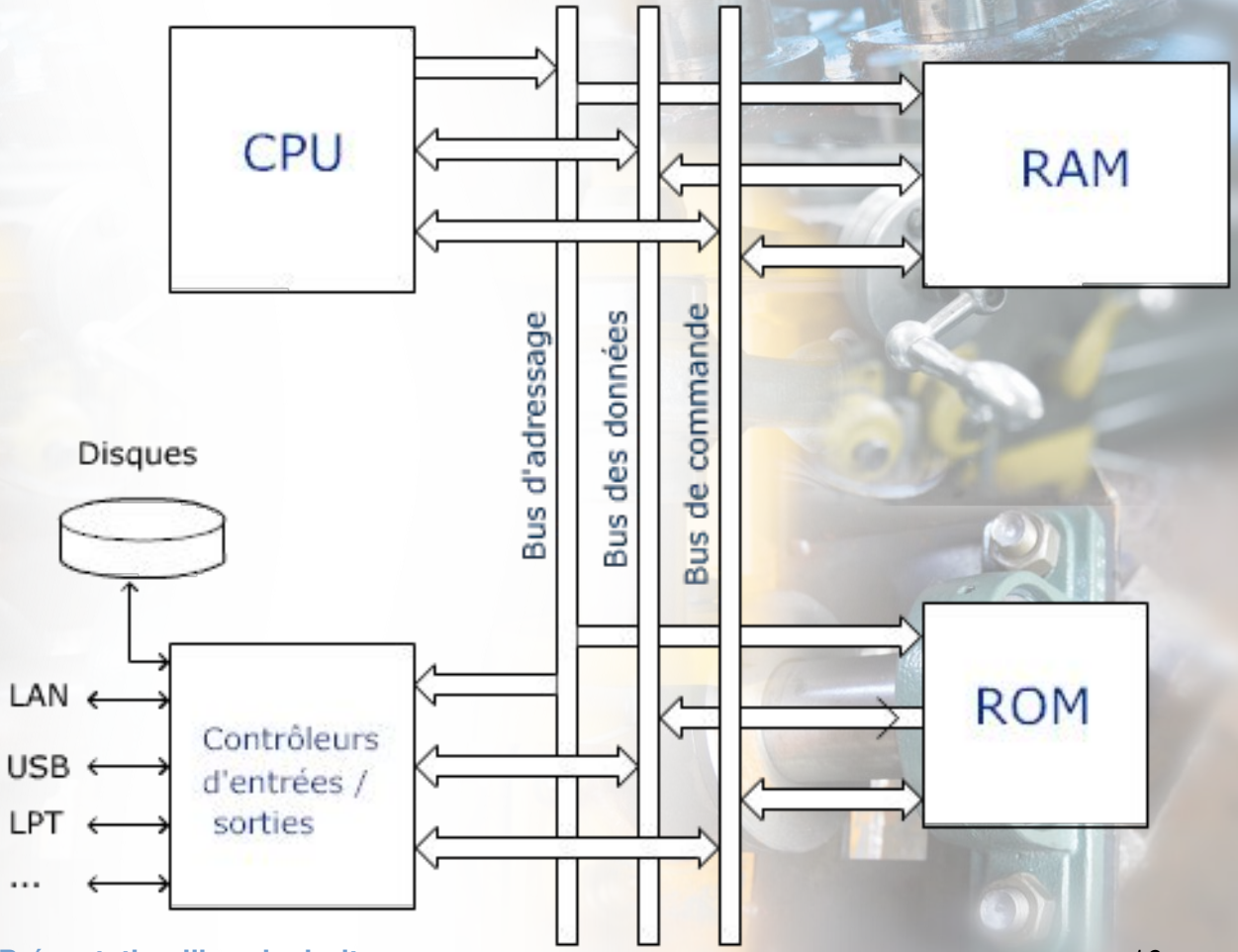


µprocesseur / carte mère

CPU : Central Processing Unit (**Microprocesseur**)
RAM : Random-Access Memory (Mémoire vive)
ROM : Read only memory
LAN : Local area network (typiquement : Ethernet)
USB : Universal Serial Bus
LPT : line printing terminal

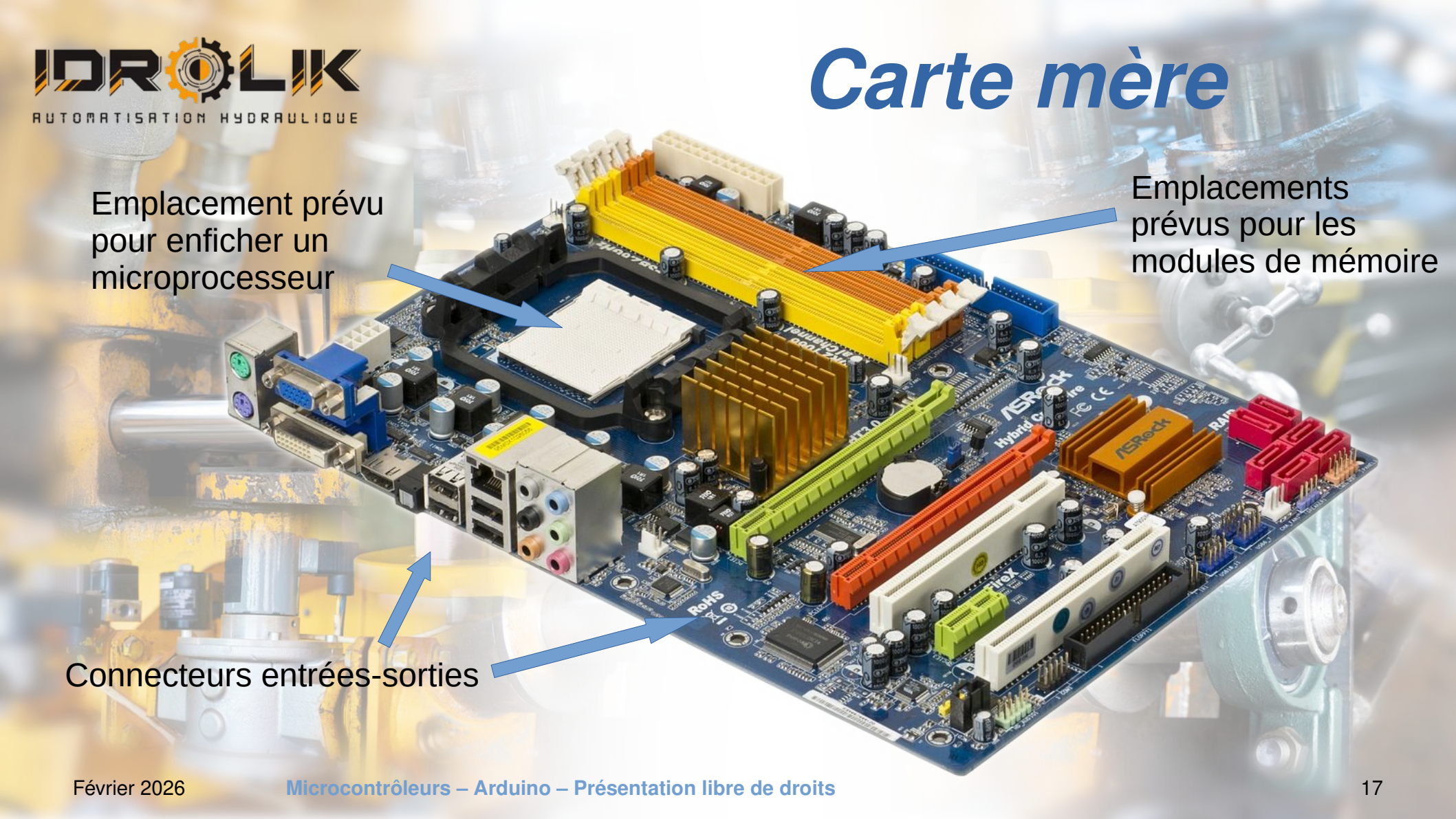
Bus d'adresse : 32 conducteurs si adressage sur 32 bits

Bus de données : 16 conducteurs si les circuits manipulent des mots de 16 bits



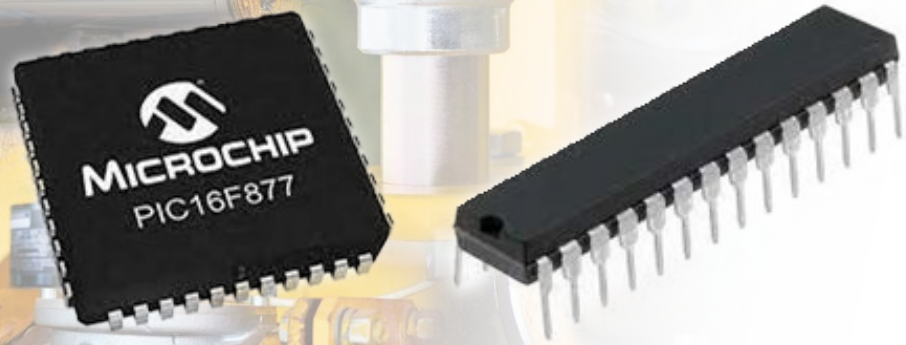
IDROLIK
AUTOMATISATION HYDRAULIQUE

Carte mère



Avec le microcontrôleur

- Pas de carte mère
- Mise en œuvre simplifiée

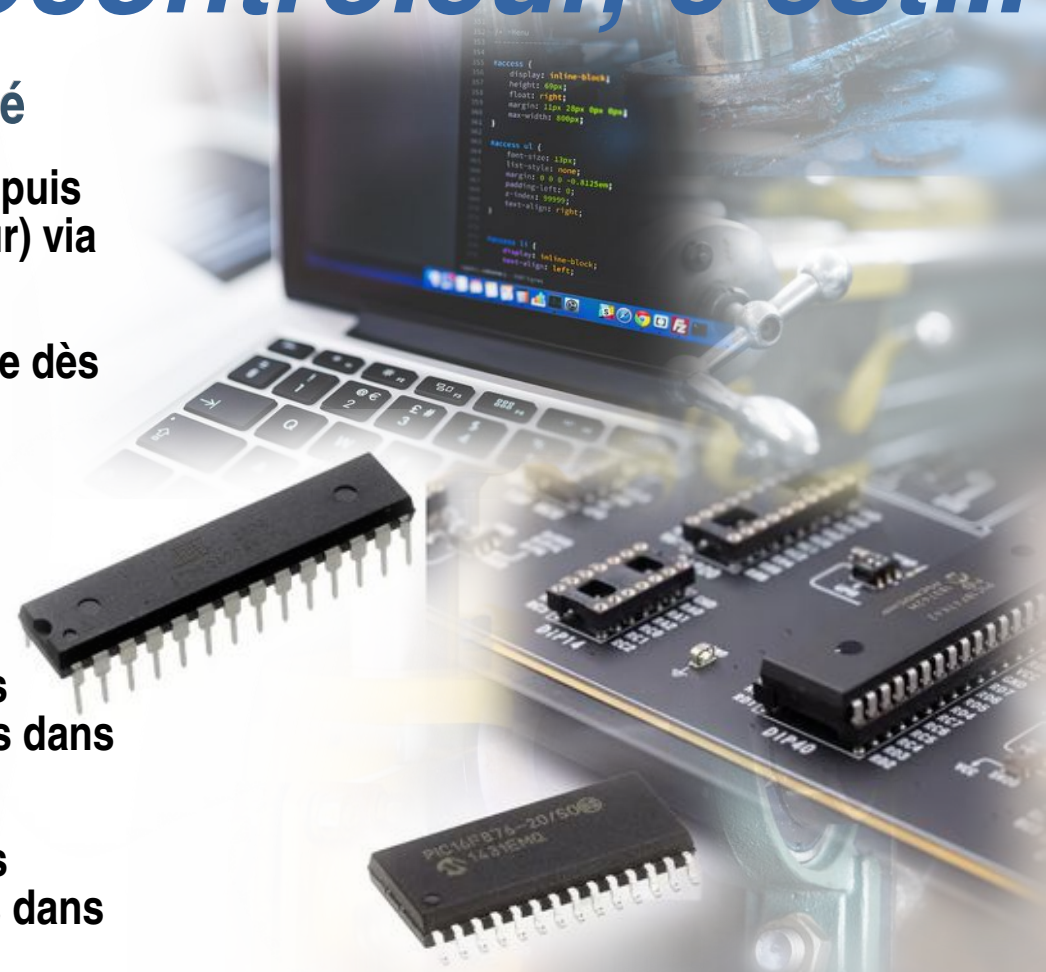


Le microcontrôleur

- Embarque tout ce qui est nécessaire à son fonctionnement dans un seul circuit
  - Processeur, mémoire, interfaces d'entrées-sorties
- Permet de séquencer automatiquement des opérations (programme informatique)
  - Avec des variables, des boucles, des tests, des fonctions…
- Permet de communiquer avec d'autres circuits électroniques
  - USB, liaison série, bus I2C, ...
- Permet de récupérer des informations sur l'environnement :
  - Boutons poussoirs, capteurs fin-de-course, capteurs de proximité (inductifs, capacitifs, ultrasons…), capteurs de température, de lumière, d'accélération, GPS…
- Permet de commander des organes de puissance (via un relais ou une électronique de puissance) :
  - Chauffage, lumière, sirène, moteur, vérin ...

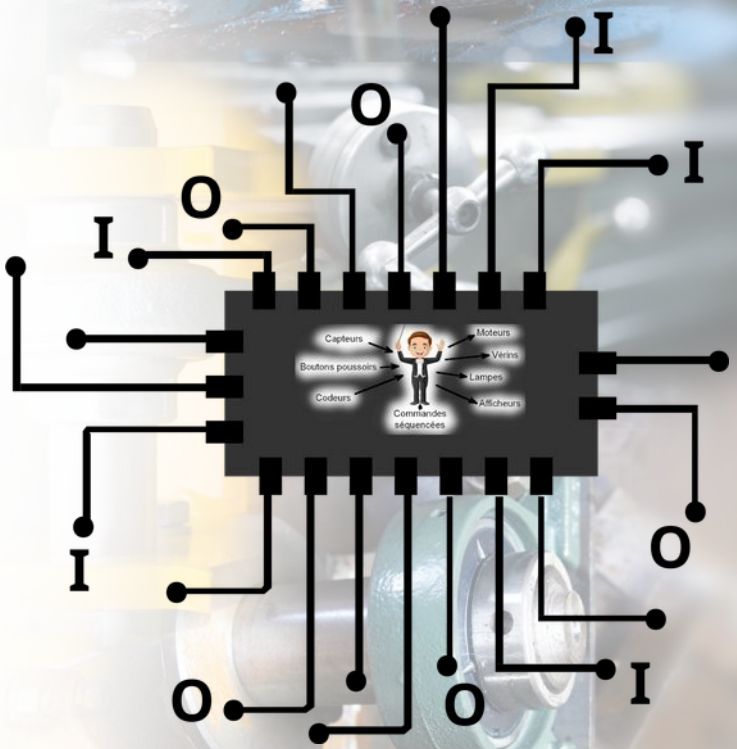
Un microcontrôleur, c'est...

- Un circuit électronique micro programmé
  - Le programme est préparé sur ordinateur puis téléversé (ordinateur vers microcontrôleur) via un câble USB
  - Le programme s'exécute ensuite en boucle dès la mise sous tension du microcontrôleur (fonctionnement en autonomie)
- Un circuit électronique qui présente des broches (contacts)
  - Certaines broches peuvent être désignées comme des entrées (à l'aide d'instructions dans le programme)
  - Certaines broches peuvent être désignées comme des sorties (à l'aide d'instructions dans le programme)



Un microcontrôleur, c'est...

- **Un circuit électronique dont les broches désignées comme des entrées peuvent être lues par le programme**
  - Une tension proche de 5 volts appliquée sur une broche d'entrée numérique est interprétée comme un 1
  - Une tension proche de 0 volt appliquée sur une broche d'entrée numérique est interprétée comme un 0
  - Certaines entrées peuvent être configurées pour lire une tension analogique entre 0 et 5V
  - Les entrées permettent de lire l'état de boutons poussoirs, de capteurs …
- **Un circuit électronique dont les tensions sur les broches désignées comme des sorties peuvent être mises à 0 ou à 1 par le programme**
  - Une sortie mise à 1 applique une tension de 5 volts sur la broche correspondante
  - une sortie mise à 0 applique une tension de 0 volt sur la broche correspondante
  - Les sorties permettent de commander des lampes, des moteurs …



En résumé

- **Avec le microcontrôleur :**
  - Pas de clavier/souris comme l'ordinateur, plutôt des boutons poussoirs, des potentiomètres, des boutons rotatifs, des claviers à code ...
  - Pas d'affichage graphique évolué (fenêtres avec gestion devant/derrière…), plutôt des LED, des afficheurs à une ou plusieurs lignes de caractères ...
  - Pas de programme applicatif lourd, plutôt des programmes courts et astucieux, le talent du programmeur s'exprime dans la concision et l'efficacité
  - Pas de multitâche massif géré par un système d'exploitation, plutôt une boucle de surveillance rapide et des traitements sur interruption ...
- **Ses forces :**
  - Le microcontrôleur est aisé à mettre en œuvre
  - Le microcontrôleur est adapté à la gestion d'automatismes au plus près du matériel
  - Le microcontrôleur peut communiquer en direct avec d'autres microcontrôleurs et avec le monde extérieur via des modules électroniques additionnels spécialisés

Exemple machine à café

- **Entrées microcontrôleur**
  - Bouton de sélection café long ou café court
  - Bouton poussoir de lancement d'un cycle
  - Capteur de niveau d'eau mini dans le réservoir d'eau
  - Sonde de température d'eau dans l'unité de chauffage
- **Sorties microcontrôleurs**
  - Résistance de chauffage de l'eau
  - Pompe pour pousser l'eau chaude sous pression à travers la dosette

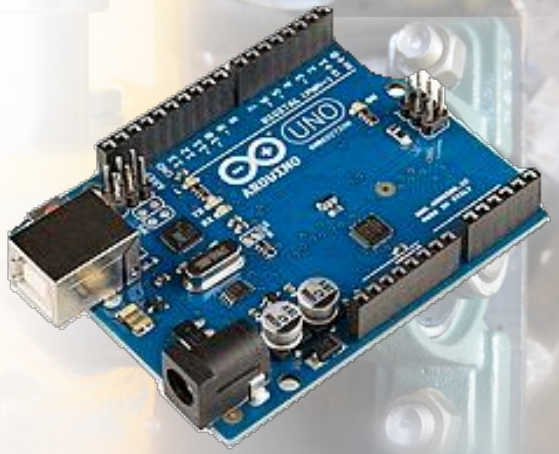
- **Boucle programme microcontrôleur**
  - Attendre l'appui sur le bouton « démarrage »
  - Activer (mettre à 1) la sortie « chauffage de l'eau »
  - Attendre que l'eau soit chaude (surveillance entrée sonde de température)
  - Activer (mettre à 1) la sortie « pompe »
  - Attendre X secondes suivant entrées « café long / café court », attente interrompue si bouton « démarrage » appuyé pendant le fonctionnement
  - Mettre à 0 la sortie « chauffage de l'eau »
  - Mettre à 0 la sortie « pompe »
  - Pendant le cycle, surveiller le capteur de niveau d'eau et mettre les sorties à 0 si manque d'eau détecté

IDROLIK
AUTOMATISATION HYDRAULIQUE

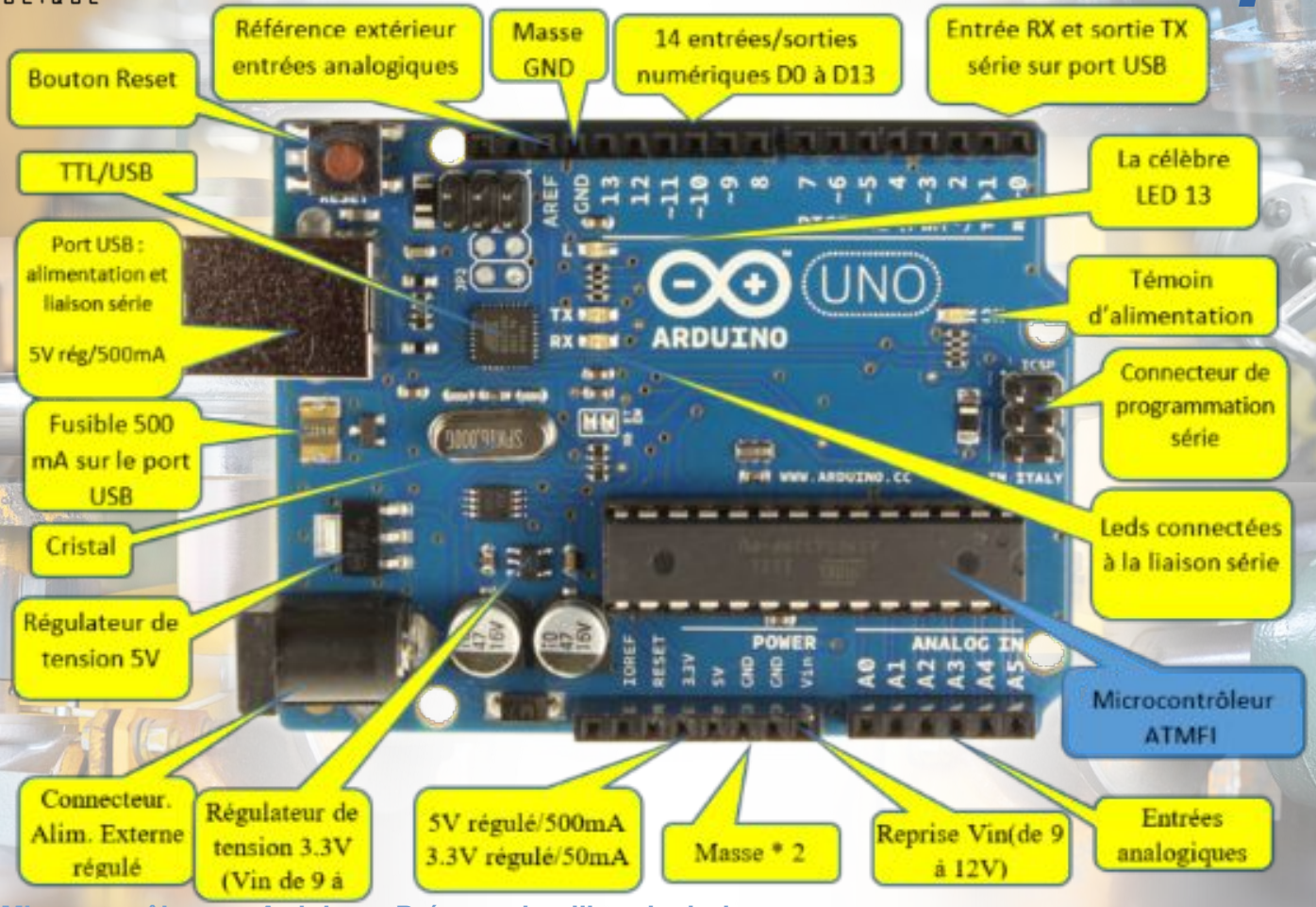
• 2 - Arduino, qu’est-ce que c’est ?

Arduino

- L'environnement Arduino intègre
  - Des modules électroniques basés sur des microcontrôleurs (Atmega ...)
  - Chaque module est équipé du petit nécessaire pour fonctionner (oscillateur, connecteurs ...)
  - Une interface logicielle de développement disponible sur Linux et Microsoft Windows
- Qu'est-ce qu'Arduino a de particulier?
  - Arduino est largement répandu
  - C'est à la fois une société et une vaste communauté open source
  - Le matériel n'est pas cher
  - L'ensemble est très bien documenté, avec une communauté importante
  - L'interface de programmation et de téléversement est open source, pratique et gratuite



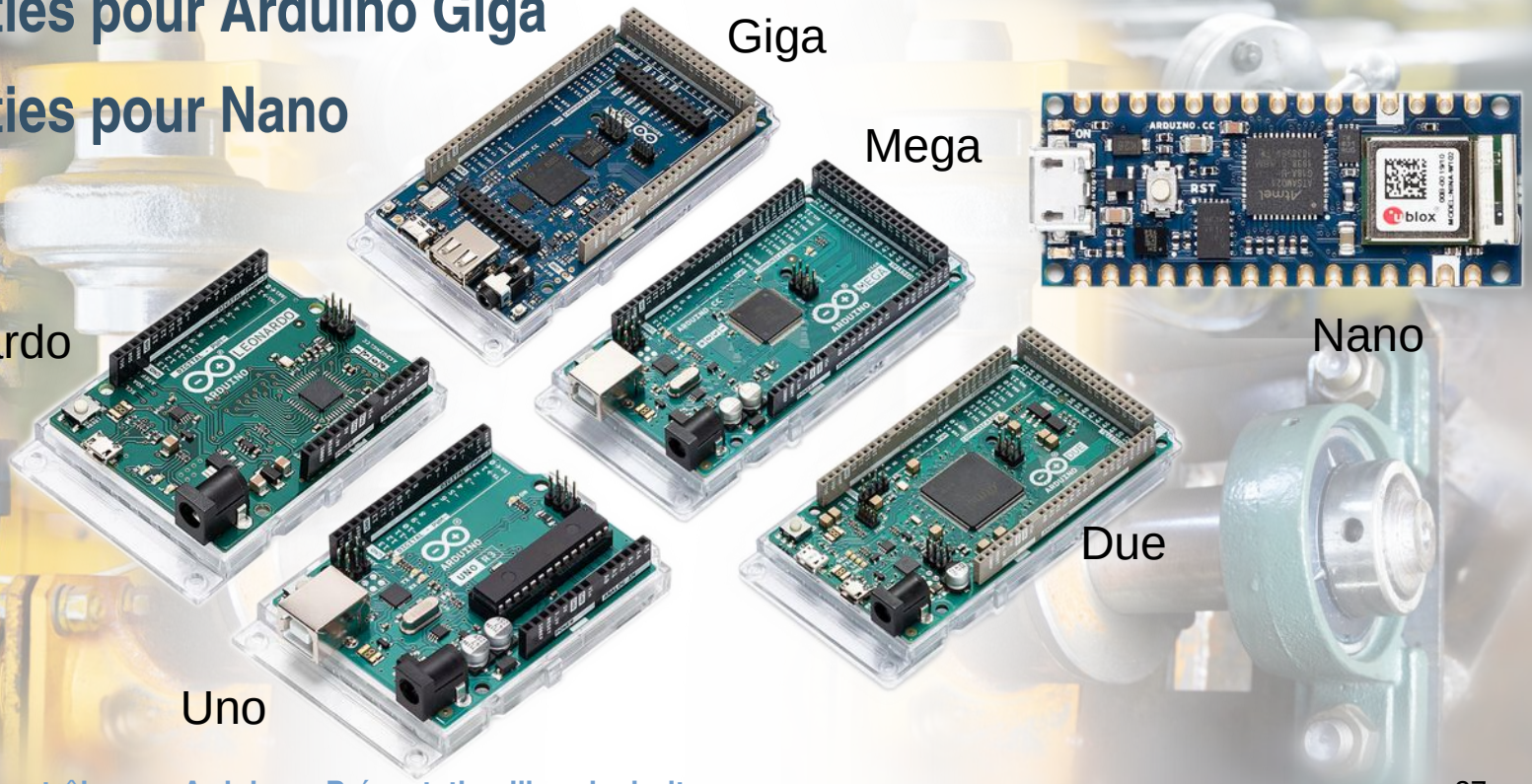
Module : on trouve quoi ?



IDROLIK
AUTOMATISATION HYDRAULIQUE

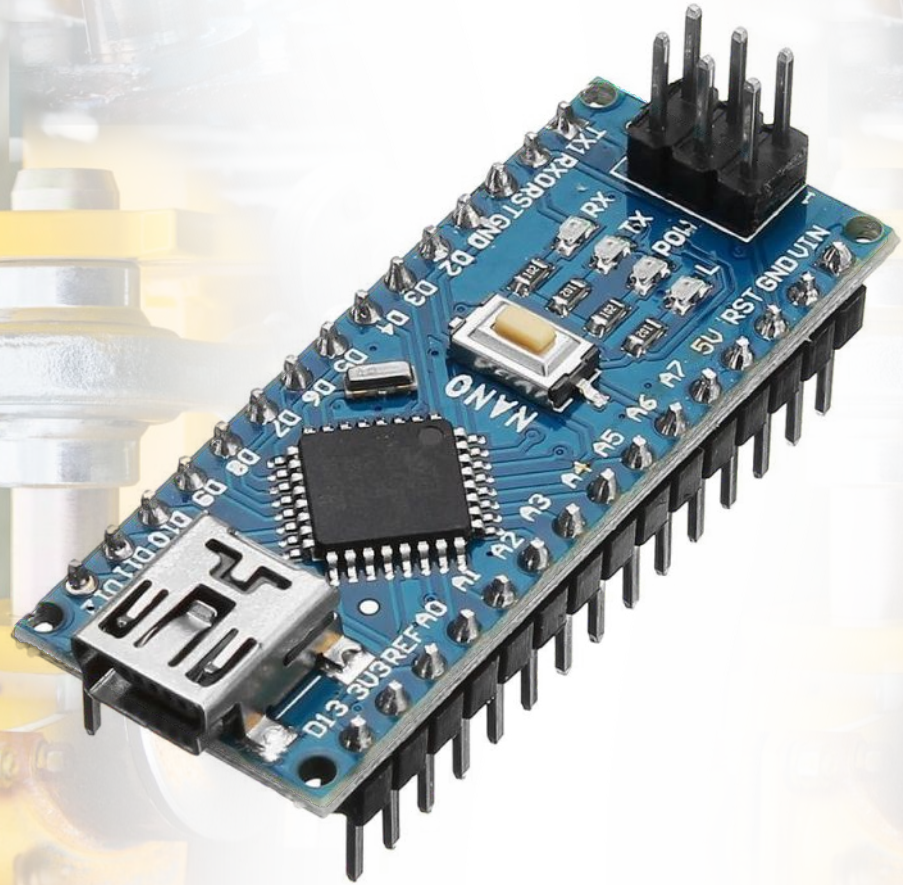
Arduino

- Propose plusieurs modules basés sur différents microcontrôleurs
- 76 entrées-sorties pour Arduino Giga
- 14 entrées-sorties pour Nano



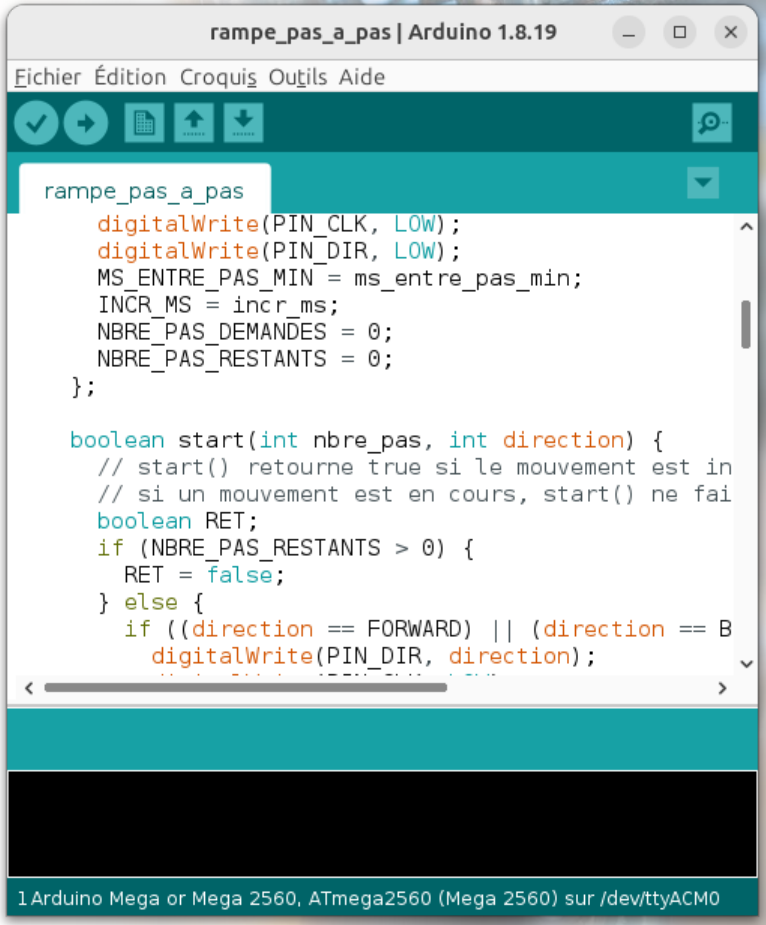
IDROLIK
AUTOMATISATION HYDRAULIQUE

Le petit de la série



IDE Arduino

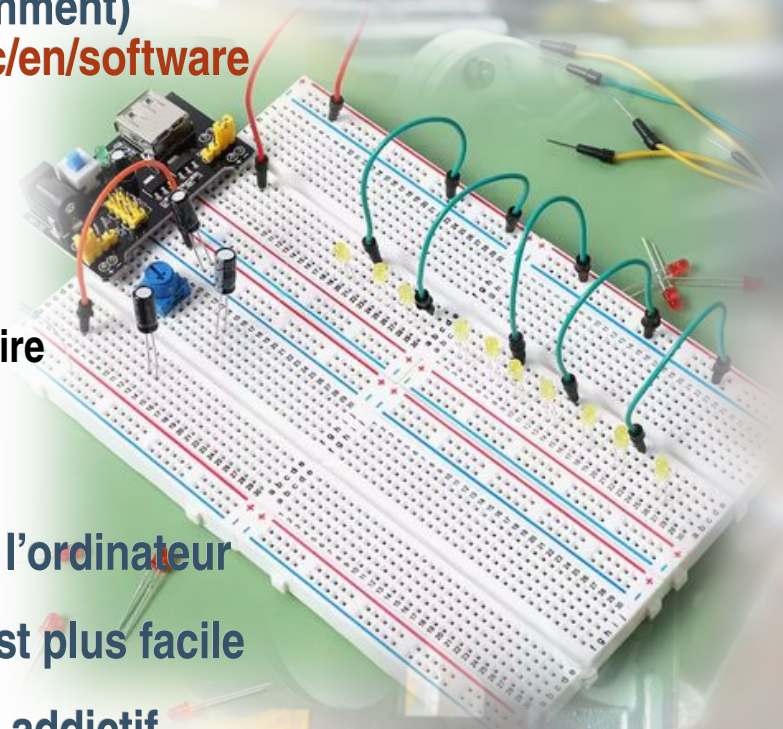
- L'IDE (Integrated Development Environment) Arduino …
  - Est un logiciel open source à installer sur un ordinateur
  - Est une interface qui permet
    - De rédiger les programmes
    - De générer le code machine et de le téléverser sur le module Arduino (via un câble USB)
  - Est basé sur le compilateur open source GCC (langage C/C++)
    - Compilation = « traduction » du texte écrit en respectant la syntaxe C/C++ en codes machine exécutables par le microcontrôleur cible
    - Il est nécessaire de préciser la carte Arduino avec laquelle on travaille avant compilation et téléchargement (génération du bon code machine compréhensible par le microcontrôleur cible)



IDROLIK
AUTOMATISATION HYDRAULIQUE

Démarrage avec Arduino

- Il faut un ordinateur (portable ou fixe)
- Il faut télécharger l'IDE (Integrated Development Environment) Arduino depuis le site Arduino : https://www.arduino.cc/en/software
  - Une fois l'IDE Arduino installé sur l'ordinateur, la liaison Internet n'est plus nécessaire
- Il faut acheter une carte Arduino, par exemple Nano
  - Si vous êtes sympa, vous l'achetez chez Arduino pour faire vivre la société et contribuer aux évolutions futures
  - Vous pouvez aussi acheter un clone chinois
- Il faut acheter un câble USB pour connecter l'Arduino à l'ordinateur
- Il faut acheter une plaque d'essai et des fils d'essai, c'est plus facile
- Il faut l'envie d'essayer. Attention, Arduino peut devenir addictif

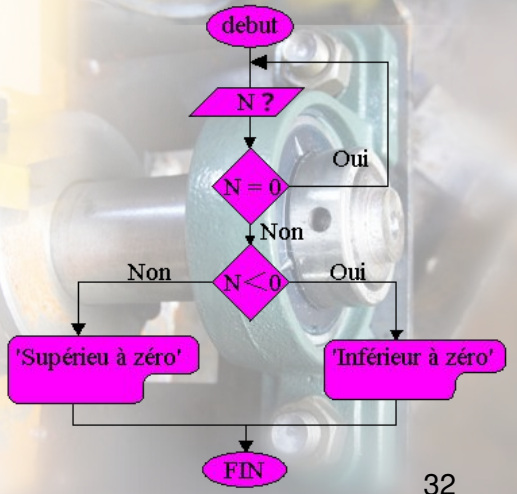


IDROLIK
AUTOMATISATION HYDRAULIQUE

• 3 - L’interface de programmation

Écrire un programme

- C'est être clair sur ce qu'on veut automatiser avec le microcontrôleur
  - Un organigramme et/ou un descriptif rédigé avec des phrases sont deux bon moyens de se fixer un cahier des charges (= contrat fonctionnel)
  - « Ce qui se conçoit bien s'énonce clairement et les mots pour le dire s'énoncent clairement »
    Ça va sans dire, mais ça va mieux en le disant, j'insiste
  - Question structurante liée au micro-contrôleur : quelles informations va-t-on lire (entrées) et quelles commandes va-t-on générer (sorties) ?
- Une bonne pratique : écriture et test de programmes élémentaires correspondant chacun à un couple brique matérielle + brique logicielle
  - Choix et tests unitaires des capteurs que l'on pense utiliser
  - Choix et tests unitaires des actionneurs pressentis avec leur électronique de puissance (une sortie du microcontrôleur délivre quelques milliampères)
  - Ça veut dire qu'il faut identifier des matériels candidats et étudier leur datasheets. Datasheet pas claire = A FUIR.
    Trop de fonctions intégrées = A FUIR
    Capteurs compliqués à utiliser = A FUIR



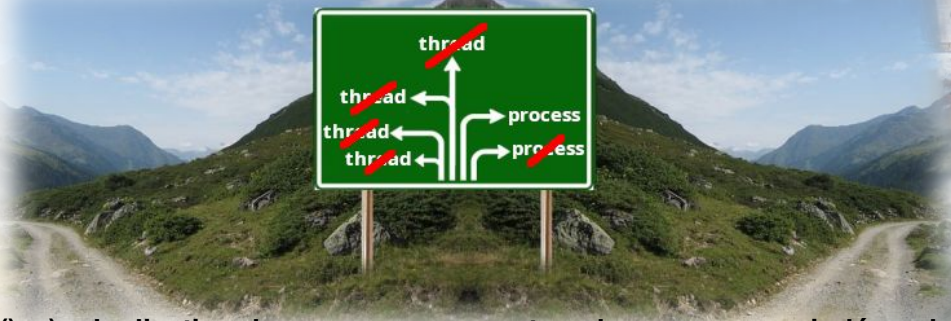
Écrire un programme

- **Une bonne pratique : penser dès le début à la maintenabilité**
  - On est très content de s'y retrouver des années après
    - Lorsqu'on veut améliorer et/ou étendre le programme
    - Lorsqu'un dispositif électronique qui n'est plus fabriqué tombe en panne et que l'on veut en utiliser un autre
  - Démarche et logique doivent rester claires et lisibles (= pas de bidouillage)
    - On commente abondamment : ce qui semble naturel lorsqu'on a la tête dans le projet ne l'est plus forcément des années après
    - On évite les pirouettes qui font gagner trois lignes de code : la lisibilité devrait rester la priorité
    - On regroupe ce qui va ensemble (données et fonctions) dans des objets (= classes) aussi indépendants les uns des autres que possible
  - L'interface avec chaque élément matériel devrait se limiter à un seul objet logiciel (= une classe)
    - Le remplacement d'un capteur ou d'un actionneur ne doit pas remettre en cause l'ensemble du programme (impact aussi localisé que possible)
    - Éviter les matériels multifonction : un matériel pour une fonction, c'est bien

Écrire un programme

- Apprendre à se servir de l'IDE
  - Écrire du texte dans la fenêtre
  - Lancer la compilation et le téléversement vers le microcontrôleur (presse-bouton)
- Écrire un texte qui respecte la syntaxe du C++ :
  - Un programme est une suite d'instructions en C++ qui produit le fonctionnement attendu
- Trouver et corriger les erreurs de compilation (= instruction C++ → codes machine)
  - Le compilateur détecte des variables non initialisées, des incohérences de types, des dépassements de capacité ...etc… dont l'origine n'est pas toujours facile à identifier
- Tester le fonctionnement et revenir sur le programme autant que de besoin (essai – erreur)
  - Le plan de test doit être aussi exhaustif que possible
  - Logiciel instable = irritant

Spécificités C++ Arduino



- **Avec Arduino, on ne peut pas :**
  - Dupliquer un processus (fonction « fork() ») : duplication du processus courant en deux processus indépendants vis-à-vis du système d'exploitation, chacun avec son pointeur d'exécution, sa pile, son espace mémoire, son contexte.
    Par exemple, une variable qui a reçu une valeur avant le fork() continue à vivre sa vie dans chaque processus (= avec une valeur possiblement différente dans chaque processus)
  - Faire appel au multi threading = traitements d'un même processus qui tournent indépendamment, ce qui permet d'exploiter le caractère multi cœurs du microprocesseur tout en partageant le même espace mémoire (= les mêmes variables)
- **Pourquoi ?**
  Parce que ces deux fonctions font appel au système d'exploitation de l'ordinateur
  - Sur un ordinateur, quand on ne peut plus augmenter la fréquence d'horloge d'un processeur (limitation liée à l'électronique), on essaie de paralléliser les traitements sur plusieurs processeurs et/ou sur plusieurs cœurs
- **Certaines fonctions proches du hardware sont spécifiques à l'Arduino**
  - Exemples : « digitalWrite() » pour mettre 0 ou 1 sur une patte initialisée en sortie numérique, « analogRead() » pour lire la tension sur une patte initialisée comme une entrée analogique...

Spécificités C++ Arduino

- Deux fonctions sont exécutées par le microcontrôleur
- La fonction « setup() »
  - Exécutée une fois au démarrage du microcontrôleur
- La fonction « loop() »
  - Exécutée en boucle tant que le microcontrôleur est sous tension
- Une fois le programme implanté ...
  - Le microcontrôleur n'a plus besoin du câble USB (sauf si on souhaite s'en servir comme alimentation)
  - A chaque mise sous tension du microcontrôleur, « setup() » est exécuté, puis « loop() » est exécuté en boucle

```
class LED { // declaration d'une classe
  public :
    boolean etat_LED; // LED allumee ou eteinte
    LED(boolean etat_initial) { // Constructeur
      etat_LED = etat_initial;
    }
    void change () {
      if (etat_LED == true) {
        etat_LED = false;
      }
      else {
        etat_LED = true;
      }
    }
};

LED ma_LED = new LED(false); // instanciation
```

```
void setup() {
  /* setup() = fonction appelee une fois par au
     demarrage */
  pinMode(3, OUTPUT); // la patte 3 est une sortie
}
```

```
void loop() {
  /* loop() = fonction appelee en boucle par le
     microcontroleur */
  delay(1000); // attente 1000 ms = 1 s
  ma_LED.change();
  digitalWrite(3, ma_LED.etat_LED);
}
```

IDROLIK
AUTOMATISATION HYDRAULIQUE

C++

- **Commentaires :**
  - Dès qu'on trouve « // », le reste de la ligne est ignoré par le compilateur
  - Tout ce qui est encadré par « /* » et « */ » est ignoré par le compilateur
- **Un « ; » indique la fin d'une instruction**
  - Plusieurs instructions peuvent se suivre sur une même ligne, le séparateur est le « ; » mais ...
  - Si on veut rendre le programme illisible, on met plusieurs instructions à la suite sur la même ligne
- **Les blocs sont encadrés par « { } »**
- **Une bonne pratique : « outils » + « formatage automatique »**
  - En user et en abuser

```
class LED {// declaration d'une classe
  public :
    boolean etat_LED;// LED allumee ou eteinte
    LED(boolean etat_initial) {// Constructeur
      etat_LED = etat_initial;
    }
    void change () {
      if (etat_LED == true) {
        etat_LED = false;
      }
      else {
        etat_LED = true;
      }
    }
};

LED ma_LED = new LED(false);// instanciation

void setup() {
  /* setup() = fonction appelee une fois par au
     demarrage */
  pinMode(3, OUTPUT);// la patte 3 est une sortie
}

void loop() {
  /* loop() = fonction appelee en boucle par le
     microcontroleur */
  delay(1000);// attente 1000 ms = 1 s
  ma_LED.change();
  digitalWrite(3, ma_LED.etat_LED);
}
```

Variables en C++

- En C++, on déclare les variables qu'on va utiliser
  - Une variable permet de stocker un type d'information
  - Les variables peuvent être de plusieurs types
    - Booléen (vrai ou faux), Entier, Réel (simple ou double précision), Caractère
      int compteur; compteur = 12 ;
      float largeur ; largeur = 32,7 ;
      char carac ; carac = 'c' ; carac = 99 ;
    - Classes = objets composites
  - Un nom de variable est une suite de caractères (a, titi3, perimetre33, h_2_v …)
    - Pas de blancs (mettre des « _ » à la place si on veut un séparateur), pas de caractères accentués, pas de chiffre en première position, pas d'opérateurs… Globalement, pas de fantaisie, ça évite les ennuis
- Bonne pratique : déclarer toutes les variables en début de programme
  - Dans un fichier à part si besoin (directive de préprocesseur « #include nom_fichier »)

```
class LED { // declaration d'une classe
  public :
    boolean etat_LED; // LED allumee ou eteinte
    LED(boolean etat_initial) { // Constructeur
      etat_LED = etat_initial;
    }
    void change () {
      if (etat_LED == true) {
        etat_LED = false;
      }
      else {
        etat_LED = true;
      }
    }
};

LED ma_LED = new LED(false); // instanciation
```

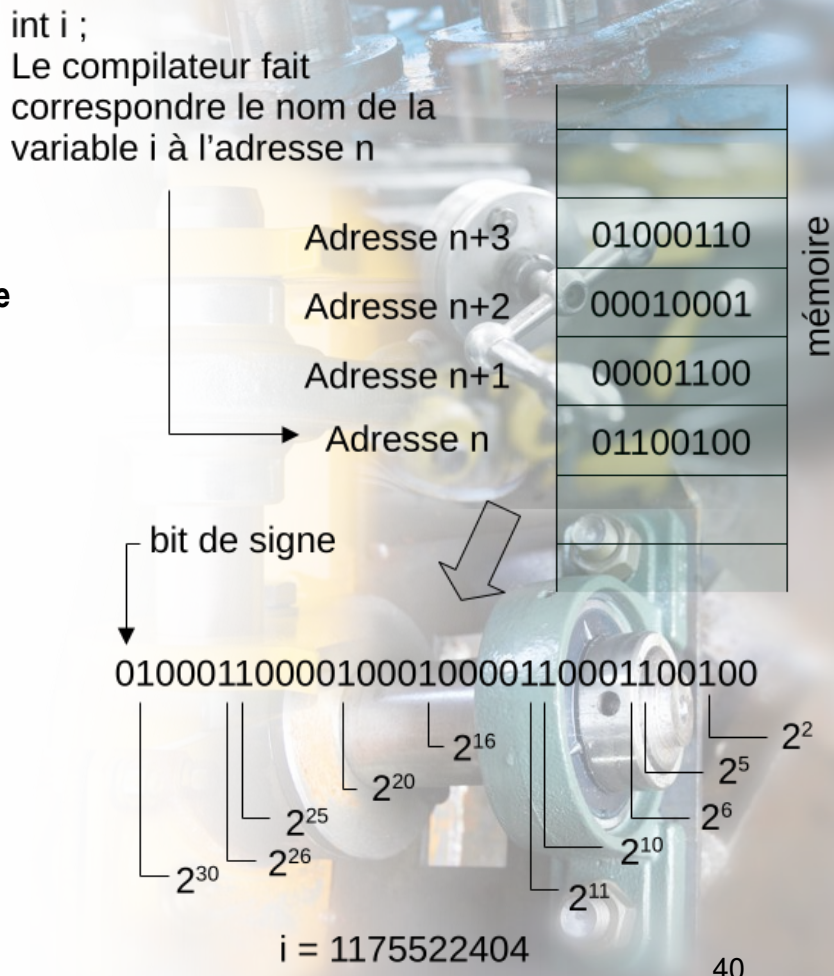
```
void setup() {
  /* setup() = fonction appelee une fois par au
     demarrage */
  pinMode(3, OUTPUT); // la patte 3 est une sortie
}

void loop() {
  /* loop() = fonction appelee en boucle par le
     microcontroleur */
  delay(1000); // attente 1000 ms = 1 s
  ma_LED.change();
  digitalWrite(3, ma_LED.etat_LED);
}
```

IDROLIK
AUTOMATISATION HYDRAULIQUE

- **Petit tour sur les variables qu'on va utiliser**

Entier

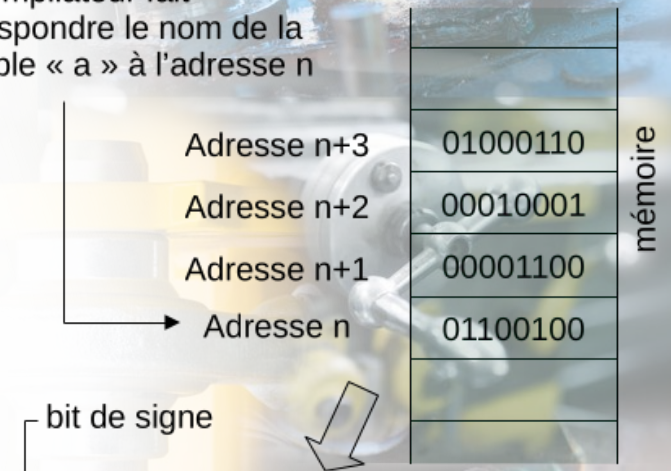
- **Déclaration d'une variable entière : « int i ; » ou « long i »**
  - La déclaration « long i » ; correspond à un entier long
  - Dans le code machine généré par le compilateur, une zone est réservée en mémoire (4 octets dans l'exemple ci-contre, 1 octet = 8 bits)
  - A chaque fois qu'on affecte une valeur à i, on modifie cette zone mémoire
- **Un entier codé sur 4 octets peut représenter un entier entre $-(2^{31} - 1)$ et $+(2^{31} - 1)$**
  - Dans la réalité, la valeur négative est codée en complément à deux (inversion bit à bit +1, dépassement ignoré).
    Ou comment transformer une soustraction en addition
- **En C++ on peut définir un pointeur vers un entier avec la déclaration « int *p ; »**
  - int i, *p ; // declaration d'un entier i et d'un pointeur p vers un entier
    p = &i ; // l'expression « &i » est l'adresse de i
    *p = 12 ; // l'adresse pointee par p reçoit 12
    // i vaut maintenant 12
  - Note : un tableau de pointeurs peut être très efficace pour trier des données volumineuses. L'intérêt des pointeurs est limité pour les entiers



Réel

- Déclaration d'une variable réelle : « float a ; » ou « double a; »
- Un réel est codé sur 4 octets avec un signe, un exposant et une mantisse
  - Signe du réel : 1 bit (0 = positif)
  - Exposant : 8 bits, valeur positive à laquelle on retire systématiqement 127 → exposant positif ou négatif
  - Mantisse : 23 bits (bits derrière « 1, » implicite)
- Exemple : 5,75 en décimal s'écrit 101.11 en binaire
  - En base 10, chaque décimale divise la précédente par 10
    0,1 = 1/10
    0,01 = 1/100
  - En binaire, chaque « décimale$_{base\ 2}$ » divise la précédente par 2
    $0,1_{base\ 2}$ = 1/2 = 0,5
    $0,01_{base\ 2}$ = 1/4 = 0,25

« float a ; »
Le compilateur fait correspondre le nom de la variable « a » à l'adresse n



01000110000100010000110001100100

Exposant
$= 2^7 + 2^3 + 2^2$
= 140
On enlève 127
Exposant = 13

Mantisse

a = $1.00100010000110001100100 \times 2^{1101}$
= $10010001000011.00011001_{base\ 2}$
= 9285,09765625

IDROLIK
AUTOMATISATION HYDRAULIQUE

Caractère

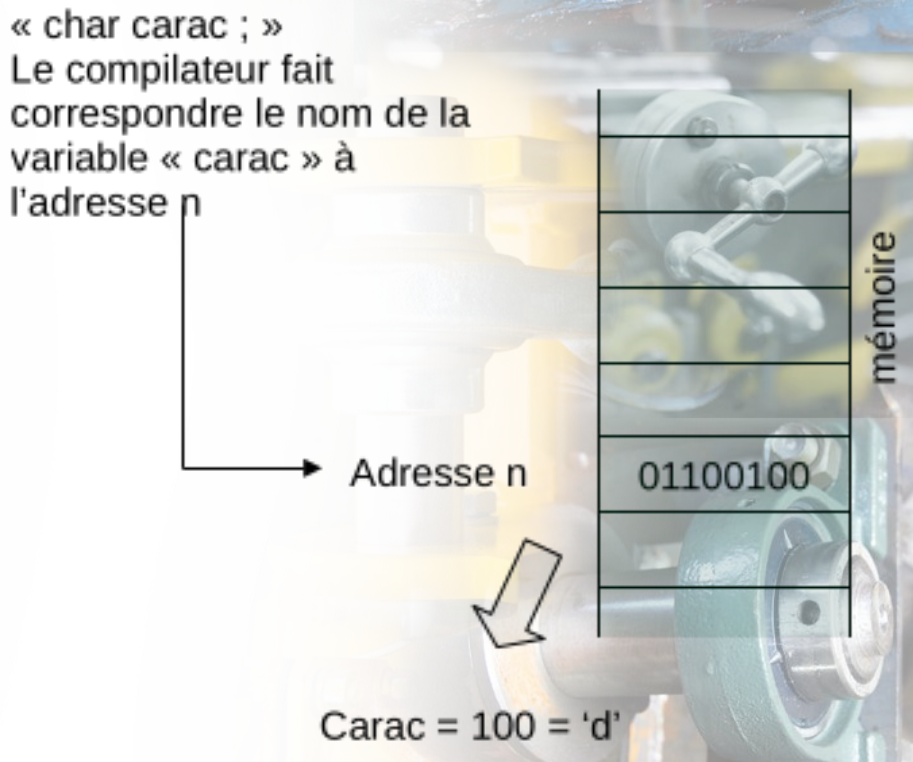
- Chaque caractère est associé à un code sur 7 bits
  - Le 8eme bit peut être utilisé comme bit de parité (= bit de contrôle)
- ASCII :
  - American Standard Code for Information Interchange
  - De 0 à 31 inclus : codes de contrôle
  - Exemple : les lignes des fichiers textes se terminent par un retour-chariot (13) et un line feed (10). (références à nos ancêtres les machines à écrire)
- Bonne pratique :
  - Rester sur les caractères du code ASCII
  - Éviter les caractères accentués (codes > 127) et les jeux de caractères spécifiques à chaque pays

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Caractère

- Un caractère consomme un octet en mémoire
  - On verra plus tard qu'on peut stocker une chaîne de caractères sous la forme d'un tableau de caractères
- Note : un booleen (vrai ou faux) est stocké comme un caractère
  - 0 = faux
  - Différent de 0 = vrai



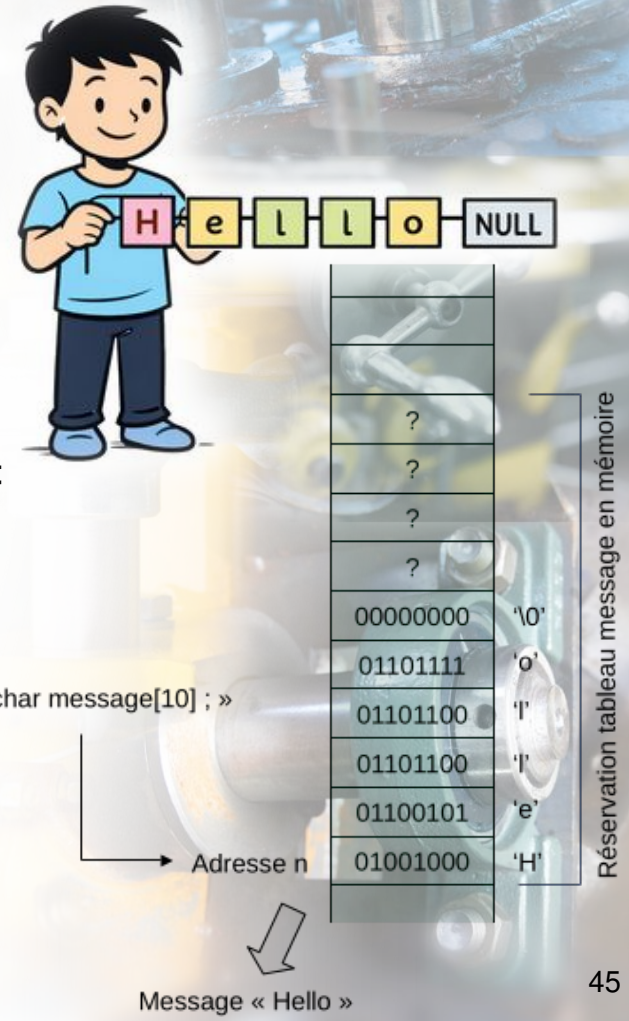
Tableaux

- Chaque type de variable peut donner lieu à un tableau
  - Utilisé pour manipuler un ensemble de données de même nature (éléments numérotés)
  - Pour chaque tableau, on indique le nombre d'éléments
  - Un tableau commence à 0
  - Chaque élément d'un tableau est utilisé comme une variable
- Tableau d'éléments de type « class »
  - Plus simple si le constructeur de la classe n'attend pas d'arguments
  - Commode pour manipuler les informations d'une collection d'objets similaires (capteurs, moteurs pas-à-pas ...)

```
int i, Toto[10];

for (i = 0; i < 9; i++) {
      Toto[Indice] = 0;
}



class LED { // declaration d'une classe
  public :
    boolean etat_LED; // LED allumee ou eteinte
    LED() { // Constructeur
      etat_LED = false;
    }
    void change ()
    {
      if (etat_LED == true) { // == comparaison
        etat_LED = false; // = affectation
      }
      else {
        etat_LED = true;
      }
    }
};

LED ma_LED[5];

for (i = 0; i < 5; i++) {
    ma_LED[i].change();
  }
```

Chaîne de caractères

- Souvent, on veut afficher des messages, sous la forme de chaînes de caractères (par exemple l'heure, ou une température, ou encore un message d'erreur)
- Il existe une classe « String » qui permet de manipuler des chaînes de caractères
  - Cette classe s'occupe de réserver dynamiquement de l'espace mémoire
- Perso, je préfère la méthode classique qui traite les chaînes de caractères comme des tableaux de caractères
  Pourquoi ? Parce que j'aime bien maîtriser la quantité de mémoire utilisée
  - La fin de la chaîne de caractères est identifiée par un caractère NULL (= 0) qui suit le dernier caractère de la chaîne
  - Bien sûr, le caractère NULL doit faire partie de la réservation en mémoire
  - Exemple : la chaîne de caractères « Hello » nécessite la réservation en mémoire d'un tableau de caractères d'au moins 6 éléments
  - char message[10] ; // reservation pour une chaine de 9 caracteres max
    message[0] = 'H' ; // elements du tableau numerotes a partir de 0
    message[1] = 'e' ;
    …
    message[5] = 'o' ;
    message[6] = '\0' ; // caractere NULL pour marquer la fin de la chaine



Classe

- **Une classe permet de regrouper des variables et des méthodes**
  - Les variables peuvent être des entiers, des réels, des caractères, des tableaux, des instanciations d'autres classes …
  - Les méthodes sont des fonctions
    - Une fonction exécute un ensemble d'instructions
    - Une fonction peut attendre des arguments (de différents types) … ou pas.
    - Une fonction peut retourner une valeur … ou pas (void).
    - Une fonction se déclare sous la forme : « int ma_fonction(int i, float a, …) { instructions… } »
- **Une classe peut être définie à partir d'une autre (héritage)**

Classe

- Pour une classe (ici « LED »), une zone mémoire est réservée pour les éléments constants et partagée par toutes les instances
  - En général, les éléments constants sont du code, on peut aussi définir des constantes (comme π)
  - Ici, un espace mémoire pour le code de la fonction « LED () »
    - Fonction de même nom que la classe = constructeur (exécuté automatiquement à la création de chaque instanciation)
  - Ici, un autre espace mémoire pour le code de la fonction « change() »
- Pour chaque instance d'une classe (ici « ma_LED »), un espace mémoire indépendant est réservé pour les variables de chaque instanciation de « LED »
  - Ici : « etat_LED »

```
class LED { // declaration d'une classe
  public :
    boolean etat_LED; // LED allumee ou eteinte
    LED(boolean etat_initial) { // Constructeur
      etat_LED = etat_initial;
    }
    void change () {
      if (etat_LED == true) {
        etat_LED = false;
      }
      else {
        etat_LED = true;
      }
    }
};

LED ma_LED = new LED(false); // instanciation

void setup() {
  /* setup() = fonction appelee une fois par au
     demarrage */
  pinMode(13, OUTPUT); // la patte 13 est une sortie
}

void loop() {
  /* loop() = fonction appelee en boucle par le
     microcontroleur */
  delay(1000); // attente 1000 ms = 1 s
  ma_LED.change();
  digitalWrite(13, ma_LED.etat_LED);
}
```

IDROLIK
AUTOMATISATION HYDRAULIQUE

- **Petit tour sur les opérations de base**

Test

- If (condition) { instructions … }
-

```
int a = 5;
int b = 8;
if (a < b):
    string meilleur_telephone = "bleu";
} else {
    string meilleur_telephone = "jaune";
}
```