Programmation embarquée

Numérique et Sciences de l’Informatique

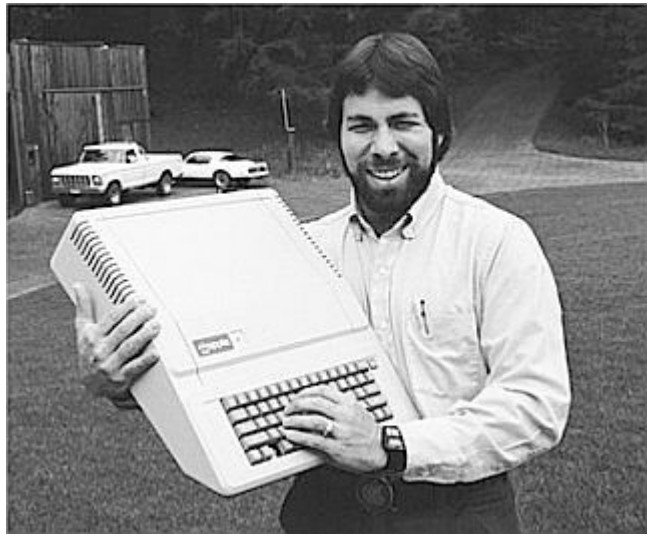
Programme (1/2)

Contenus	Capacités attendues	Commentaires
Modèle d'architecture séquentielle (von Neumann)	Distinguer les rôles et les caractéristiques des différents constituants d'une machine. Dérouler l'exécution d'une séquence d'instructions simples du type langage machine.	La présentation se limite aux concepts généraux. On distingue les architectures monoprocesseur et les architectures multiprocesseur. Des activités débranchées sont proposées. Les circuits combinatoires réalisent des fonctions booléennes.
Transmission de données dans un réseau Protocoles de communication Architecture d'un réseau	Mettre en évidence l'intérêt du découpage des données en paquets et de leur encapsulation. Dérouler le fonctionnement d'un protocole simple de récupération de perte de paquets (bit alterné). Simuler ou mettre en œuvre un réseau.	Le protocole peut être expliqué et simulé en mode débranché. Le lien est fait avec ce qui a été vu en classe de seconde sur le protocole TCP/IP. Le rôle des différents constituants du réseau local de l'établissement est présenté.

Programme (2/2)

Systèmes d'exploitation	Identifier les fonctions d'un système d'exploitation. Utiliser les commandes de base en ligne de commande. Gérer les droits et permissions d'accès aux fichiers.	Les différences entre systèmes d'exploitation libres et propriétaires sont évoquées. Les élèves utilisent un système d'exploitation libre. Il ne s'agit pas d'une étude théorique des systèmes d'exploitation.
Périphériques d'entrée et de sortie Interface Homme-Machine (IHM)	Identifier le rôle des capteurs et actionneurs. Réaliser par programmation une IHM répondant à un cahier des charges donné.	Les activités peuvent être développées sur des objets connectés, des systèmes embarqués ou robots.

Architecture



INSTRUCTION SET SUMMARY

MNEMONIC	OPERATION	IMMEDIATE OP	#	ABSOLUTE OP	#	ZERO PAGE OP	#	ACCUM OP	#	IMPLIED OP	#	(IND,X) OP	#	(IND),Y OP	#	Z PAGE,X OP	#	ABS,X OP	#	ABS,Y OP	#	RELATIVE OP	#	INDIRECT OP	#	Z PAGE,Y OP	#	PROCESSOR STATUS CODES N V - B D I Z C	MNEMONIC
ADC	A + M + C → A	69	2	6D	4	65	3					61	6	71	5	75	4	7D	4	79	4							N ↕ - - - - ↕ ↕	ADC
AND	A ∧ M → A	29	2	2D	4	25	3					21	6	31	5	35	4	3D	4	39	4							N ↕ - - - - Z -	AND
ASL	C ← [7][6][5][4][3][2][1][0] ← 0			0E	6	06	5	0A	2							16	6	1E	7									N ↕ - - - - Z ↕	ASL
BCC	BRANCH ON C = 0																					90	2					- - - - - - - -	BCC
BCS	BRANCH ON C = 1																					B0	2					- - - - - - - -	BCS
BEQ	BRANCH ON Z = 1																					F0	2					- - - - - - - -	BEQ
BIT	A ∧ M			2C	4	24	3																					M7 M6 - - - - Z -	BIT
BMI	BRANCH ON N = 1																					30	2					- - - - - - - -	BMI
BNE	BRANCH ON Z = 0																					D0	2					- - - - - - - -	BNE
BPL	BRANCH ON N = 0																					10	2					- - - - - - - -	BPL
BRK	BREAK									00	7																	- - - 1 - - - -	BRK
BVC	BRANCH ON V = 0																					50	2					- - - - - - - -	BVC
BVS	BRANCH ON V = 1																					70	2					- - - - - - - -	BVS
CLC	0 → C									18	2																	- - - - - - - 0	CLC
CLD	0 → D									D8	2																	- - - - 0 - - -	CLD
CLI	0 → I									58	2																	- - - - 0 - - -	CLI
CLV	0 → V									B8	2																	- 0 - - - - - -	CLV
CMP	A - M	C9	2	CD	4	C5	3					C1	6	D1	5	D5	4	DD	4	D9	4							N ↕ - - - - Z C	CMP
CPX	X - M	E0	2	EC	4	E4	3																					N ↕ - - - - Z C	CPX
CPY	Y - M	C0	2	CC	4	C4	3																					N ↕ - - - - Z C	CPY
DEC	M - 1 → M			CE	6	C6	5									D6	6	DE	7									N ↕ - - - - Z -	DEC
DEX	X - 1 → X									CA	2																	N ↕ - - - - Z -	DEX
DEY	Y - 1 → Y									88	2																	N ↕ - - - - Z -	DEY
EOR	A ∨ M → A	49	2	4D	4	45	3					41	6	51	5	55	4	5D	4	59	4							N ↕ - - - - Z -	EOR
INC	M + 1 → M			EE	6	E6	5									F6	6	FE	7									N ↕ - - - - Z -	INC
INX	X + 1 → X									E8	2																	N ↕ - - - - Z -	INX
INY	Y + 1 → Y									C8	2																	N ↕ - - - - Z -	INY
JMP	JUMP TO NEW LOC			4C	3	3																		6C	5			- - - - - - - -	JMP
JSR	JUMP SUB			20	6	3																						- - - - - - - -	JSR
LDA	M → A	A9	2	AD	4	A5	3					A1	6	B1	5	B5	4	BD	4	B9	4							N ↕ - - - - Z -	LDA
LDX	M → X	A2	2	AE	4	A6	3													BE	4							N ↕ - - - - Z -	LDX
LDY	M → Y	A0	2	AC	4	A4	3									B4	4	BC	4									N ↕ - - - - Z -	LDY
LSR	[7][6][5][4][3][2][1][0] → C			4E	6	46	5	4A	2							56	6	5E	7									0 ↕ - - - - Z C	LSR
NOP	NO OPERATION									EA	2																	- - - - - - - -	NOP
ORA	A ∨ M → A	09	2	0D	4	05	3					01	6	11	5	15	4	1D	4	19	4							N ↕ - - - - Z -	ORA
PHA	A → M$_S$, S - 1 → S									48	3																	- - - - - - - -	PHA
PHP	P → M$_S$, S - 1 → S									08	3																	- - - - - - - -	PHP
PLA	S + 1 → S, M$_S$ → A									68	4																	N ↕ - - - - Z -	PLA
PLP	S + 1 → S, M$_S$ → P									28	4																	RESTORED	PLP
ROL	[7][6][5][4][3][2][1][0] ← C			2E	6	26	5	2A	2							36	6	3E	7									N ↕ - - - - Z C	ROL
ROR	C → [7][6][5][4][3][2][1][0]			6E	6	66	5	6A	2							76	6	7E	7									N ↕ - - - - Z C	ROR
RTI	RTRN INT									40	6																	RESTORED	RTI
RTS	RTRN SUB									60	6																	- - - - - - - -	RTS
SBC	A - M - C → A	E9	2	ED	4	E5	3					E1	6	F1	5	F5	4	FD	4	F9	4							N ↕ - - - - Z ↕	SBC
SEC	1 → C									38	2																	- - - - - - - 1	SEC
SED	1 → D									F8	2																	- - - - 1 - - -	SED
SEI	1 → I									78	2																	- - - - 1 - - -	SEI
STA	A → M			8D	4	85	3					81	6	91	6	95	4	9D	5	99	5							- - - - - - - -	STA
STX	X → M			8E	4	86	3																			96	4	- - - - - - - -	STX
STY	Y → M			8C	4	84	3									94	4											- - - - - - - -	STY
TAX	A → X									AA	2																	N ↕ - - - - Z -	TAX
TAY	A → Y									A8	2																	N ↕ - - - - Z -	TAY
TSX	S → X									BA	2																	N ↕ - - - - Z -	TSX
TXA	X → A									8A	2																	N ↕ - - - - Z -	TXA
TXS	X → S									9A	2																	- - - - - - - -	TXS
TYA	Y → A									98	2																	N ↕ - - - - Z -	TYA

1. ADD 1 IF PAGE BOUNDARY IS CROSSED
2. ADD 1 IF BRANCH OCCURS TO SAME PAGE; ADD 2 IF BRANCH OCCURS TO DIFFERENT PAGE
3. CARRY NOT = BORROW
4. IF IN DECIMAL MODE Z FLAG IS INVALID; ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

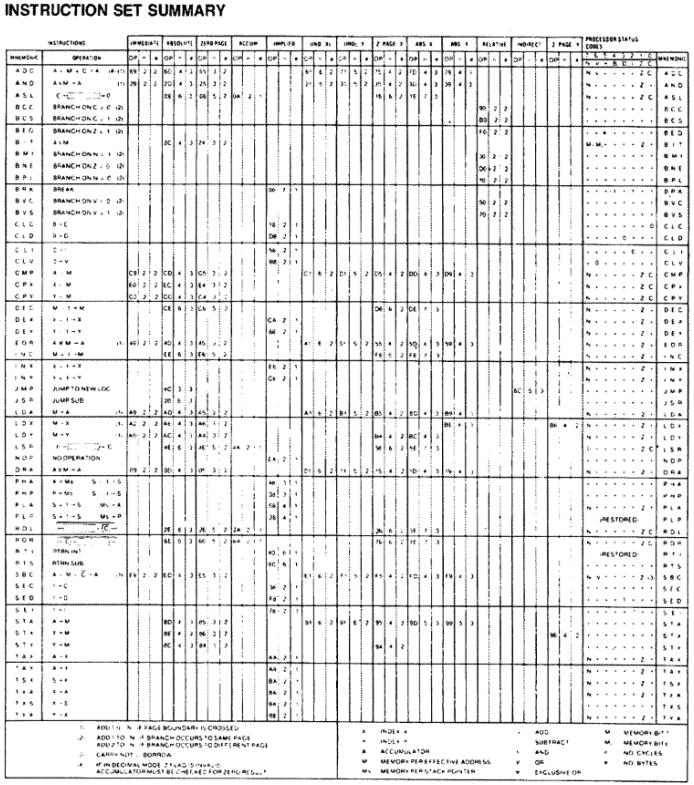
X = INDEX X
Y = INDEX Y
A = ACCUMULATOR
M = MEMORY PER EFFECTIVE ADDRESS
M$_S$ = MEMORY PER STACK POINTER

+ = ADD
- = SUBTRACT
∧ = AND
∨ = OR
∀ = EXCLUSIVE OR

M = MEMORY BIT
M$_n$ = MEMORY BIT n
- = NO CHANGE
* = NO BYTES

Architecture

- Exemple de l'**Apple II,** un des premiers ordinateurs personnels **:**
  - Processeur MOS 6502 8 bit à 1Mhz
  - RAM : 4ko
  - ROM : 8ko



INSTRUCTION SET SUMMARY

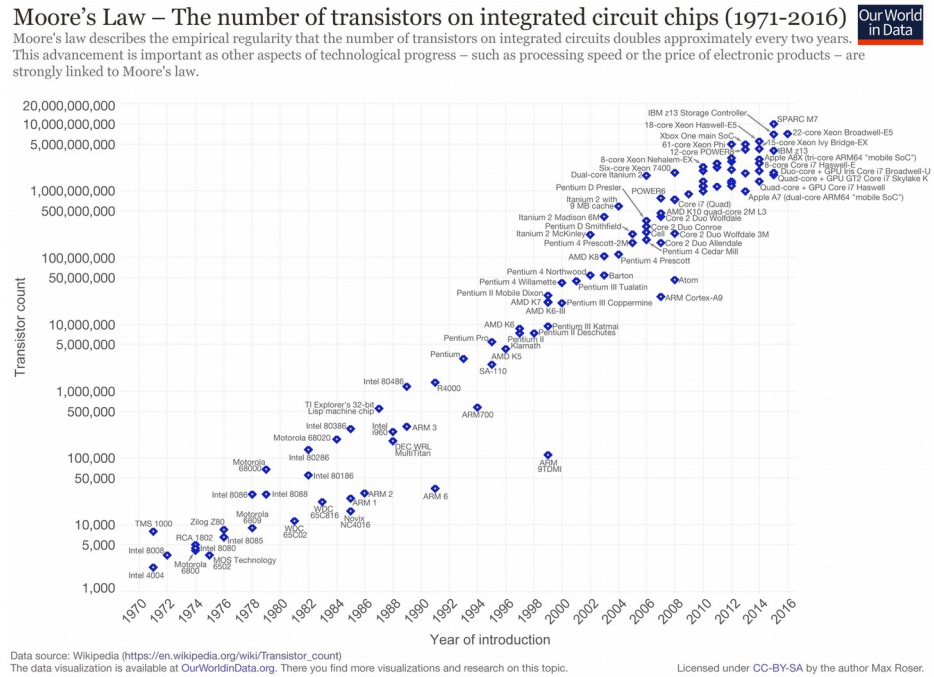
1979 vs 2019

Processeur	8bit 1Mhz	64bit 2.9Ghz
RAM	4Kio	16Gio
ROM	8Ko	1To
Coeurs	1	8
Affichage	6 couleurs 280 x 192	16 millions de couleurs 1920 x 1080
Réseau	-	Ethernet GigaBit

Loi de Moore

« The complexity for minimum component costs has increased at a rate of roughly a factor of two per year »

→ Ne présage évidement pas de l'amélioration de l'expérience utilisateur



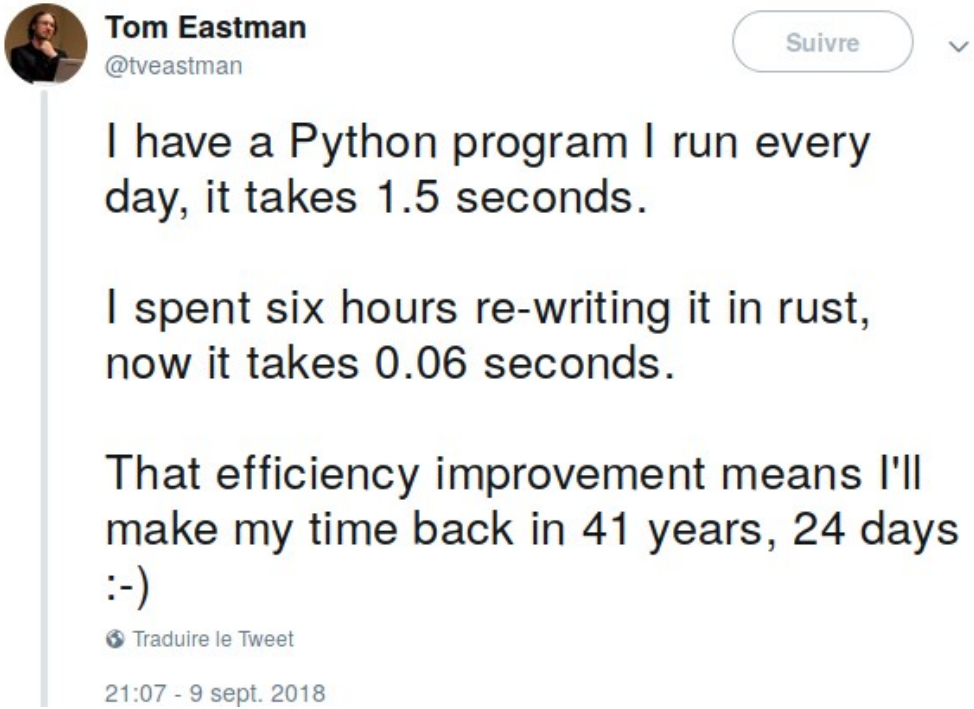
Loi de Wirth / May

Cette croissance à créé un fossé entre les développeurs et le matériel

« Software is getting slower more rapidly than hardware becomes faster. »

« Software efficiency halves every 18 months, compensating Moore's law. »

Le logiciel pas toujours à la hauteur ?



Tom Eastman
@tveastman

Suivre

I have a Python program I run every day, it takes 1.5 seconds.

I spent six hours re-writing it in rust, now it takes 0.06 seconds.

That efficiency improvement means I'll make my time back in 41 years, 24 days :-)

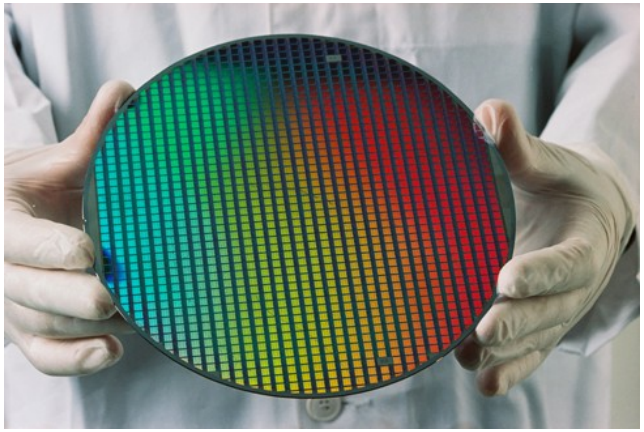
Traduire le Tweet

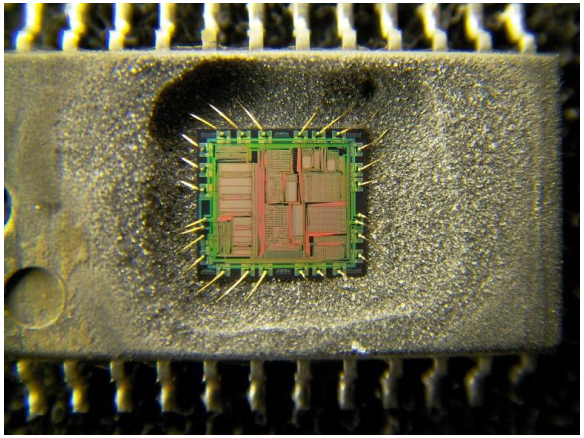
21:07 - 9 sept. 2018

Le logiciel pas toujours à la hauteur ?

- **Logiciel lent**
  - Windows 10 met **30 minutes** à se mettre à jour ...
  - ... dans le même temps, on aurait réécrit **5 fois le SSD entier**
- **Logiciel énorme**
  - Windows 95 pesait **30 Mo ...**
  - ... l'application clavier de Google (Android) consomme 150 Mo
- **Logiciel vite obselète**

Du silicium au logiciel





Les types de circuits intégrés

- Sur un circuit, on peut retrouver :
  - Les processeurs (CPU)
  - Les microcontrôleurs (MCU)
  - Les ASIC/ASSP
  - Les DSP
  - Les FPGA

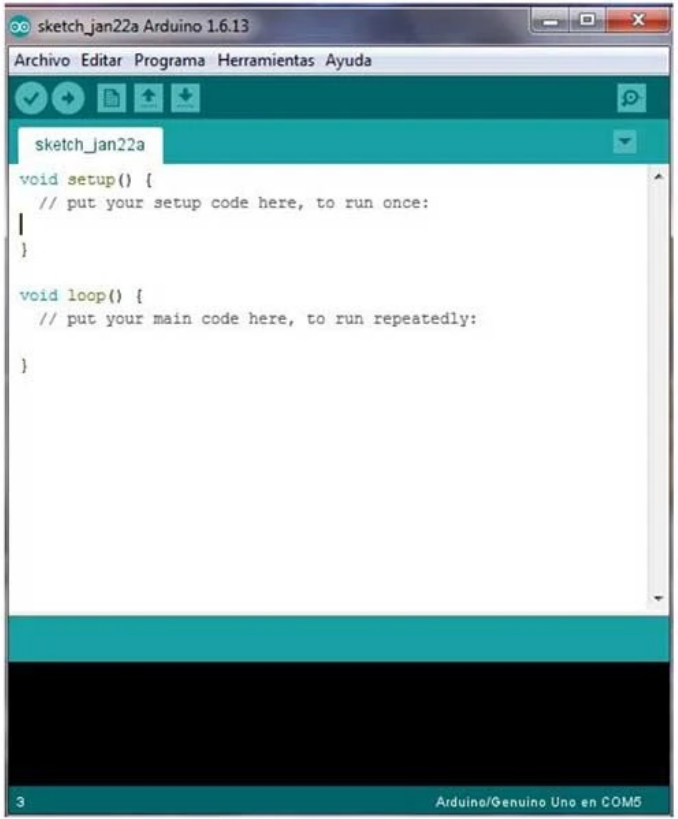
Cartes de développement

PC Portable
Windows
3Ghz
50W
~800€

Raspberry Pi
Linux
800Mhz
5W
~50€

Arduino Uno
Aucun OS
16Mhz
0.2W
~5€

L’outil Arduino



Pas de système d’exploitation

- C’est le cas par exemple des cartes Arduino
- D’après vous, quels sont les avantages ? Les inconvénients ?

Pas de système d’exploitation

- On s’épargne le surcoût mémoire d’un système (raison de l’absence sur Arduino)

  → **Pas de *préemption*, avantage pour mieux maîtriser ce qui se passe, mais très peu pratique pour développer**

Pas de système d’exploitation

- On s’épargne le surcoût mémoire d’un système (raison de l’absence sur Arduino)

  → **Pas de *préemption*, avantage pour mieux maîtriser ce qui se passe, mais très peu pratique pour développer**

Comment ça marche ?

Intel core i7
x86 64 bit

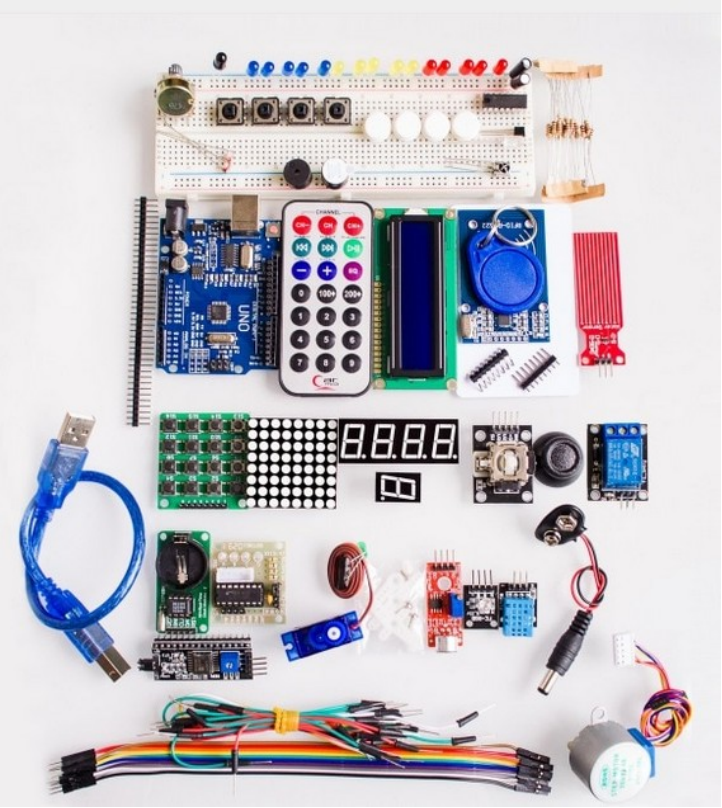
ATmega328P
AVR 8 bit

Comment ça marche ?

- L’ordinateur compile le programme pour la carte (différente architecture → on parle de cross-compilation)
- Le programme est chargé à bord à l’aide d’un *bootloader* (programme d’amorçage)
- **Contrainte :**
  - *l’interpréteur Python est trop lourd pour Arduino, nous programmons donc en C/C++ !*

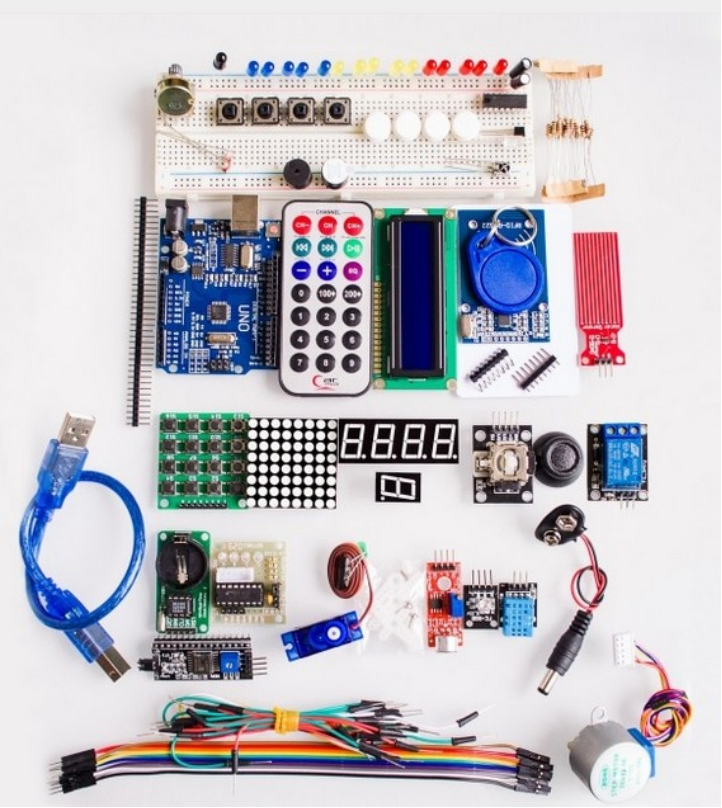
Capteurs et actuateurs

- D’un point de vue pratique, on trouve dans le commerce des kits et des capteurs/actuateurs compatibles Arduino



Capteurs et actuateurs

- **Capteurs**
  - Température, luminosité, distance, pression, accélération, vitesse de rotation, magnétique (boussole), poids, ...
- **Actuateur**
  - Moteurs, servomoteurs, moteurs à pas, moteurs linéaires, électro-aimants ...



Programmer dans Arduino IDE

- Il est possible d'utiliser l'interface série pour afficher des messages :

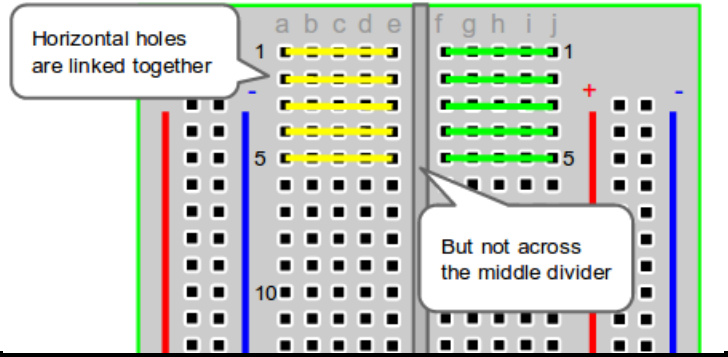
```
void setup()
{
  // Initialisation de la communication série à
  // 115200bauds
  Serial.begin(115200);
}

void loop()
{
  // Affichage d'un message
  Serial.println("Bonjour!");
  // On attend 1s
  delay(1000);
}
```

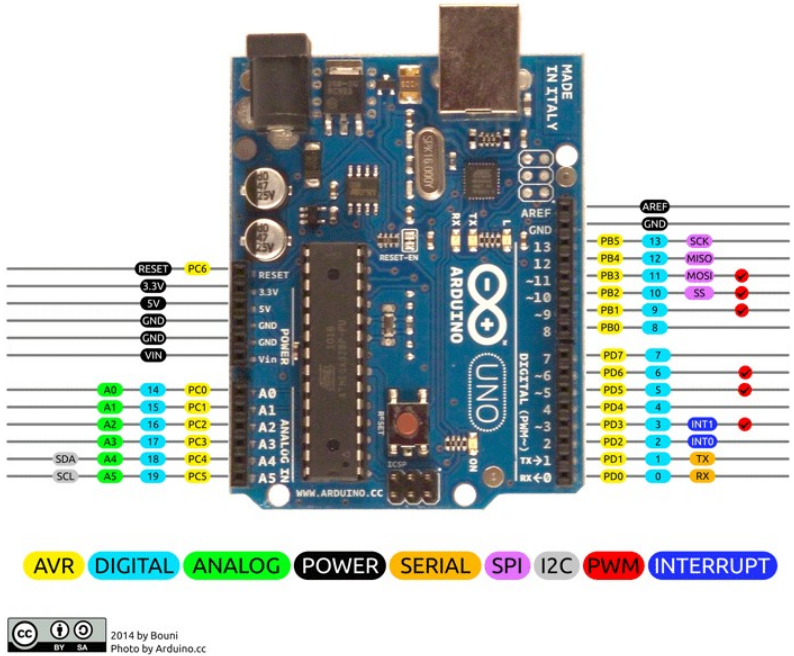
Programmer dans Arduino IDE

- Nous vous proposons un exemple de code commenté ici, qui permet :
- [exemple-complet.ino]

Broches et branchements

- Les cartes de développement sont équipées de broches (*pin* en anglais)
- On trouve aussi des platines « *breadboard* », qui permettent des branchements sans soudure :





Manipulation 1

Entrées/sorties analogiques et digitales

Entrées/sorties générales

- Toutes les broches peuvent être utilisées en entrée/sortie **digitale**
- On appelle cela **GPIO**
  - *General Purpose Input/Output*
- Le niveau « 0 » (bas) et « 1 » (haut) correspondent à un niveau voltage (par exemple 0V et 5V)

Entrées/sorties générales

- Dans Arduino, il faut paramétrer la broche (en sortie *OUTPUT* ou en entrée *INPUT*) avec la fonction *pinMode*, puis on peut contrôler son niveau logique à l'aide de *digitalWrite* :

```
// La broche 3 est en sortie
pinMode(3, OUTPUT);
// On écrit le niveau "haut" (5V)
digitalWrite(3, HIGH);
```

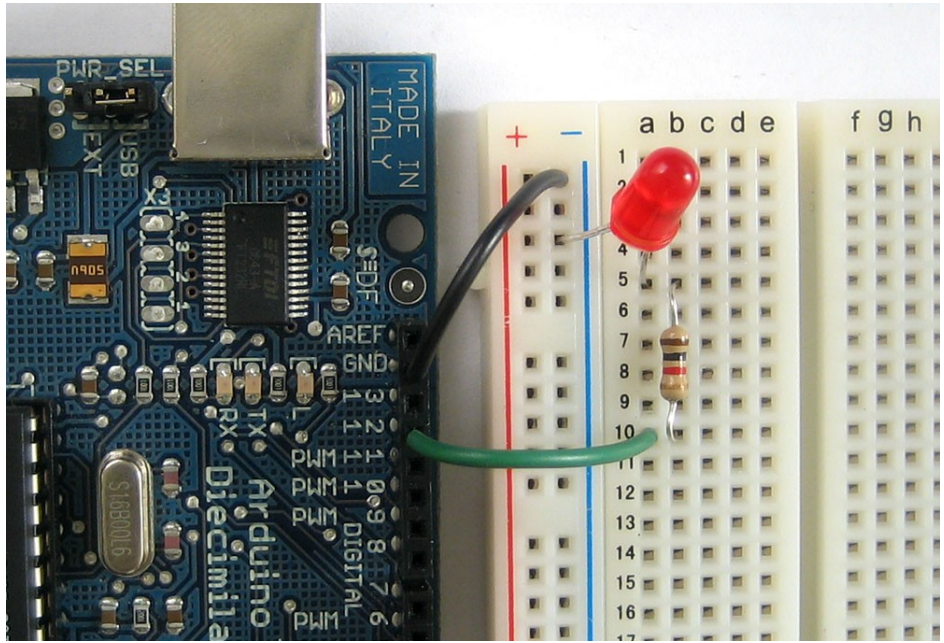
Entrées/sorties générales

- De la même manière, on peut mettre une broche en entrée et la lire à l'aide de *digitalRead* :

```
// La broche 3 est en entrée
pinMode(3, INPUT);
// Si on lit 5V, on affiche un message
if (digitalRead(3) == HIGH) {
  Serial.println("Niveau haut!")
}
```

Branchement d’une LED

- Voici à quoi ressemble le branchement d’une LED et de sa résistance:

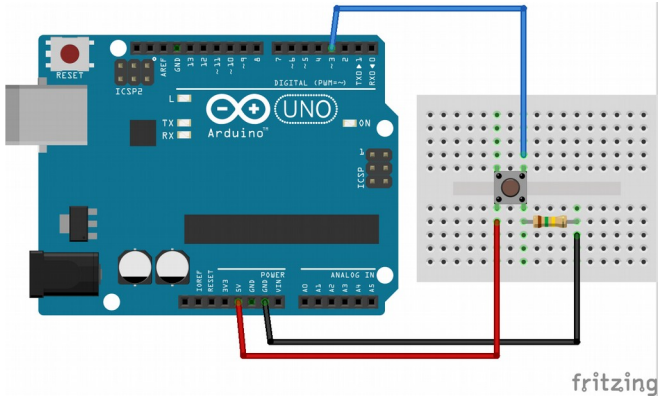


Branchement d’un bouton

- Pour lire un bouton, on peut utiliser un branchement similaire à l’image ci-contre.
- La résistance ici permet de s’assurer que l’on lit un « 0 » lorsque le bouton n’est pas appuyé (résistance de tirage)

Note : une autre solution est d’activer la résistance de tirage interne :

```
pinMode(10, INPUT_PULLUP);
```



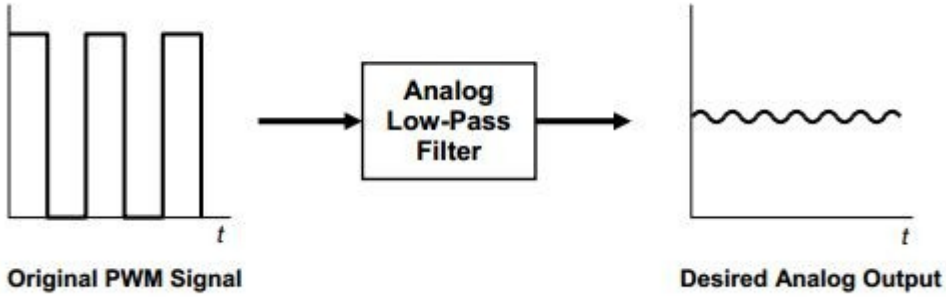
Entrées/sorties analogiques

- Les entrées analogiques permettent de mesurer une valeur non-binaire, par exemple entre 0V et 5V. On appelle cela l'échantillonnage par un convertisseur analogique/numérique
- Sur Arduino, cette valeur est sur 10 bits (de *0* à *1023*) :

```
// La broche "A1" est configurée en entrée
pinMode(A1, INPUT);

if (analogRead(A1) > 512) {
  Serial.println("Le voltage est plus haut que 2.5V!");
}
```

Entrées/sorties analogiques

- Dans l'autre sens, il n'est pas possible de produire une valeur analogique à proprement parler, car la carte n'est pas équipée de convertisseur numérique/analogique
- Cependant, il est possible de produire un signal carré, dont on règle le *rapport cyclique.* On nomme ce principe **PWM** (pulse width modulation):



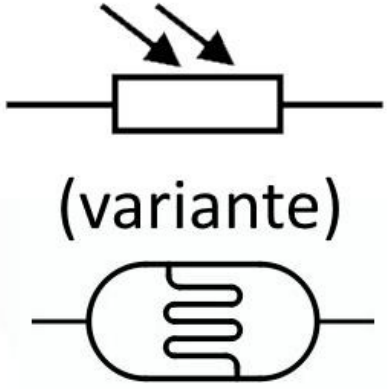
Entrées/sorties analogiques

- Arduino permet de faire cela à l'aide de *analogWrite* :

```
// La broche D9 est configurée en sortie
pinMode(9, OUTPUT);

// On règle le rapport cyclique à 200 sur 255
analogWrite(9, 200);
```

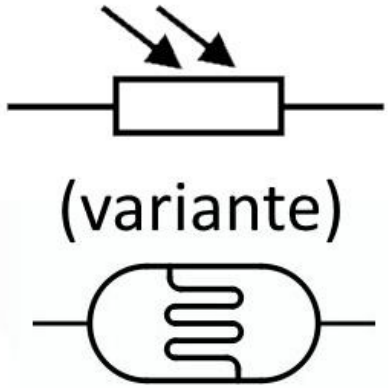
Échantillonnage analogique d’une résistance

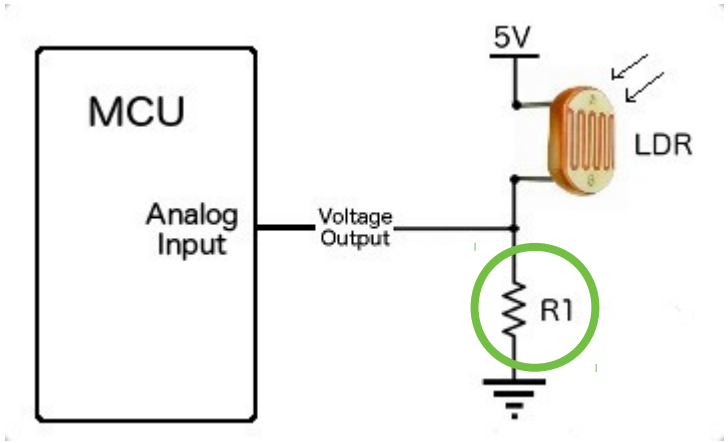
- Certains capteurs sont en fait des résistances qui varient selon la température ou la luminosité. Comment les échantillonner ?



Échantillonnage analogique d'une résistance

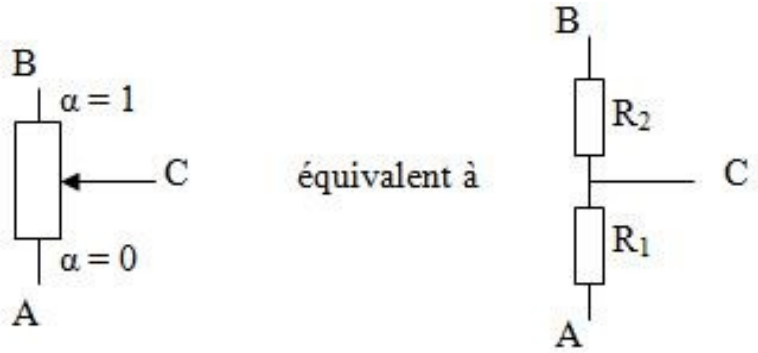
- On utilise simplement un pont diviseur de tension à l'aide d'une résistance fixe que l'on ajoute :





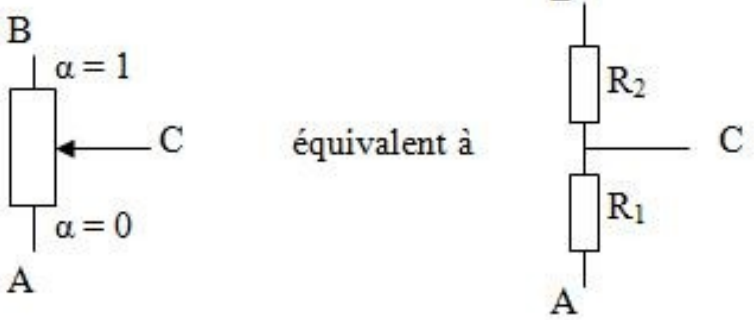
Échantillonnage d’un potentiomètre

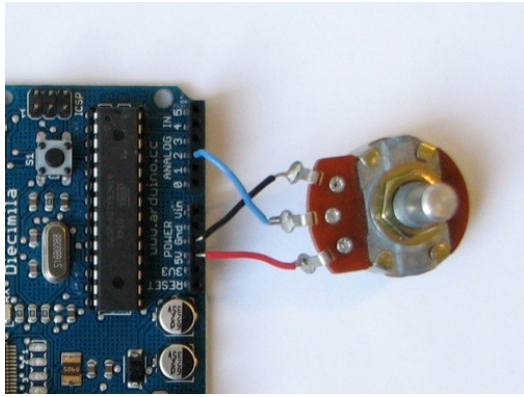
- Le potentiomètre est un type de capteur qui fonctionne en rotation et qui équivaut au circuit suivant :



Échantillonnage d’un potentiomètre

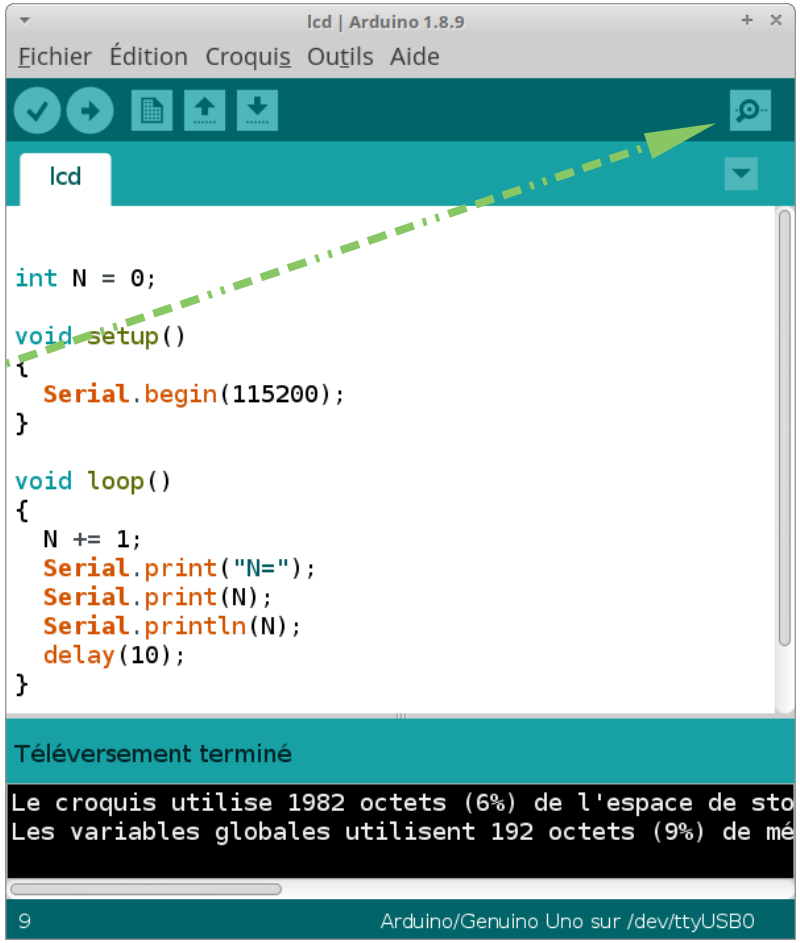
- Pour l’échantillonner, on applique donc un voltage aux extrémités (par ex. 5V) et on mesure le voltage au point central, qui variera de 0V à 5V :





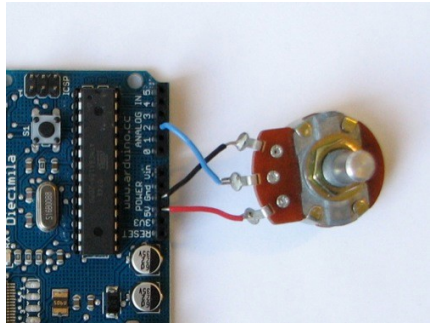
Moniteur série

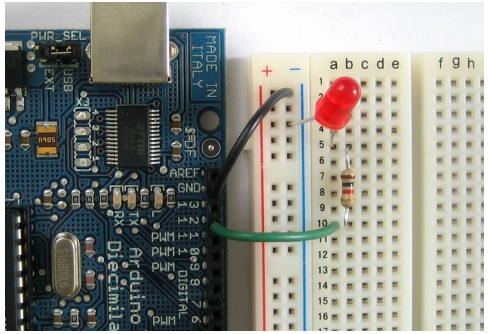
- La carte peut communiquer pendant qu'elle tourne à l'aide du moniteur série
- Pour cela, il faut utiliser la même fréquence (*baudrate*) de communication
- [serial-print.ino]



A vous de jouer !

- Affichez dans le moniteur série les valeurs du potentiomètre
- A l'aide de *analogWrite*, faites varier l'intensité de la LED en fonction du potentiomètre !
  - Attention : le potentiomètre est échantillonné entre 0 et 1023 (10 bits) alors que *analogWrite* attend une valeur entre 0 et 255 (8 bits)



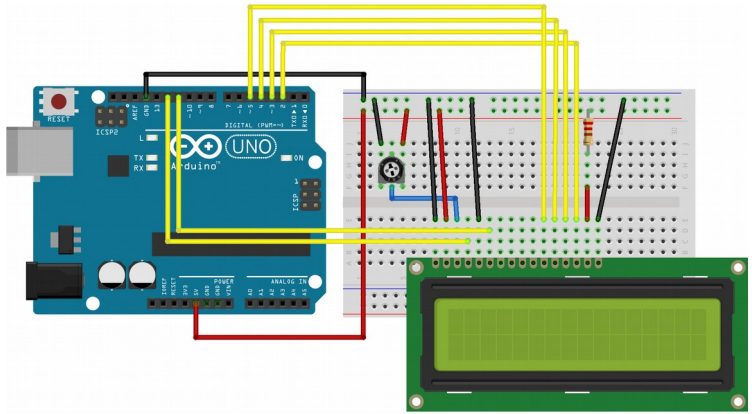


Manipulation 2

Pilotage de moteurs

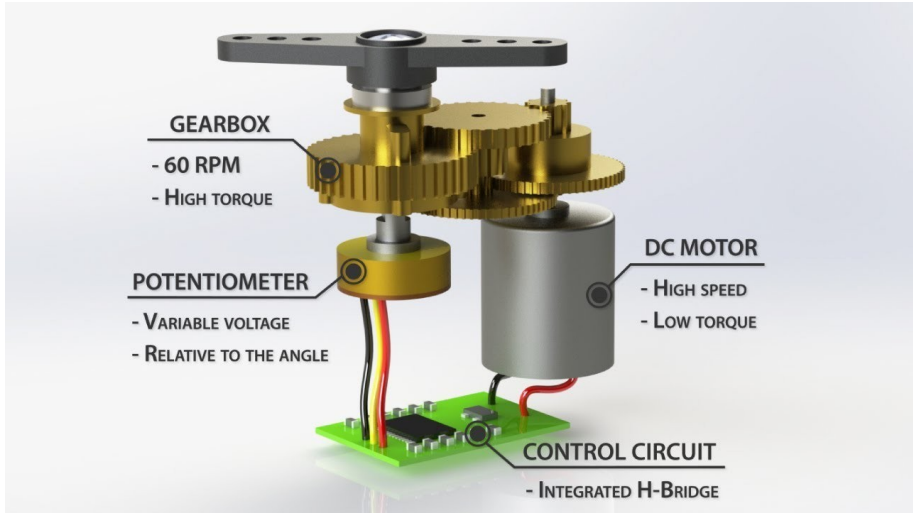
Bus de communication

- La carte peut également dialoguer avec des capteurs via des bus de communication (UART, I²C, SPI, …)
- Il existe de nombreuses bibliothèques, et la possibilité d'en télécharger des supplémentaires sur internet !



Servomoteurs

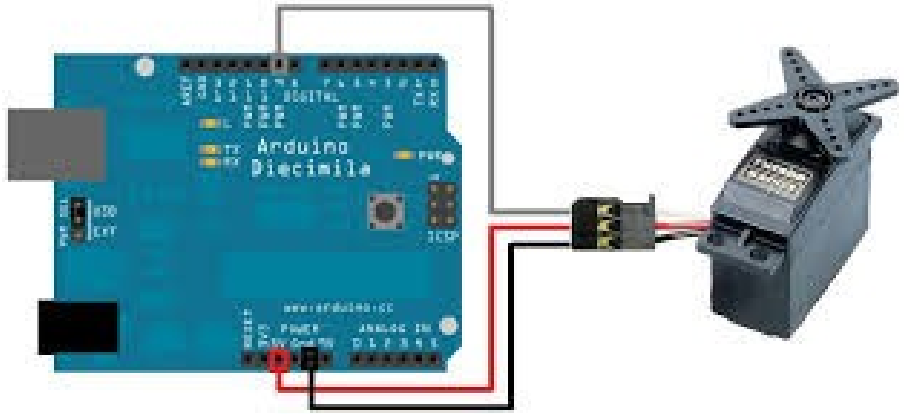
- Un servomoteur est un actuateur équipé :
  - D’un moteur DC
  - D’un étage de réduction (engrenages / *gearbox*)
  - D’un potentiomètre



Servomoteurs

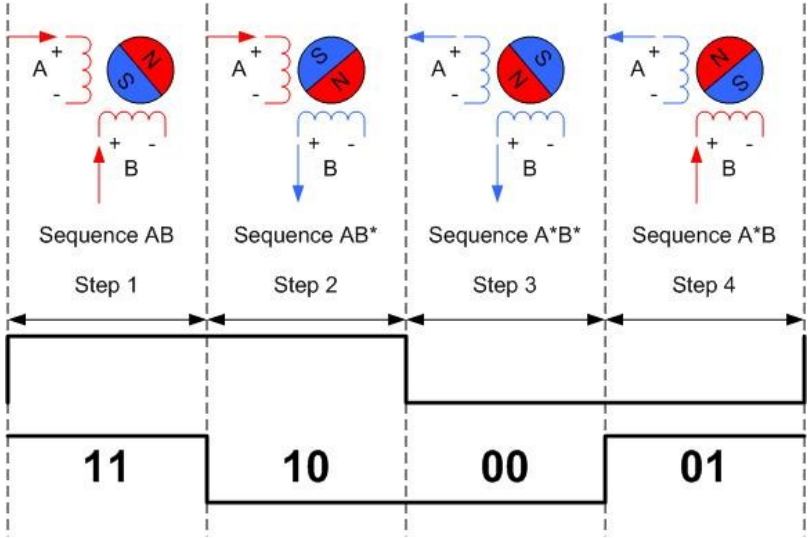
- La connexion se fait simplement (2 broches d'alimentation + 1 broche de signal), et la communication peut se faire via la bibliothèque « Servo.h » :

```
#include <Servo.h>

Servo servo;

void setup()
{
  servo.attach(9)
}

void loop()
{
  servo.write(0); // 0°
  delay(500);
  servo.write(90); // 90°
  delay(500);
}
```



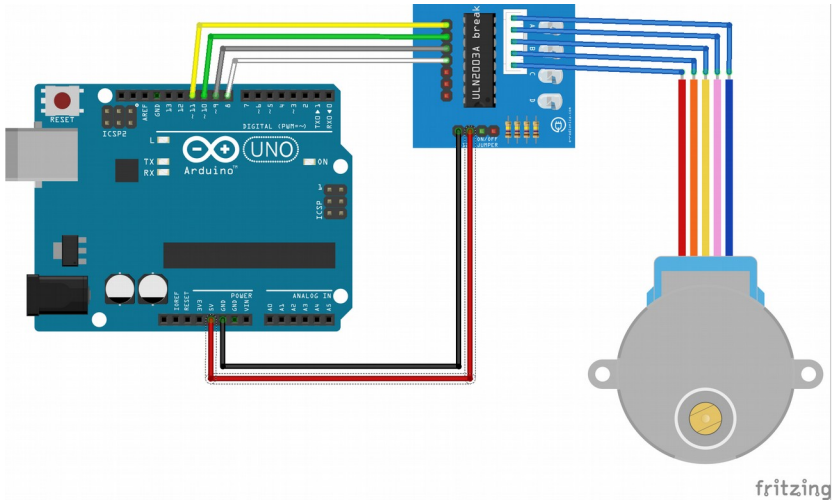
Moteurs à pas

- Les moteurs à pas sont basés sur des pôles, en général deux, qui sont activés avec des polarités différentes selon une certaine séquence, par exemple :



Moteurs à pas

- Nous utiliserons une carte de contrôle pour piloter le moteur, qui se branche comme cela :



A vous de jouer !

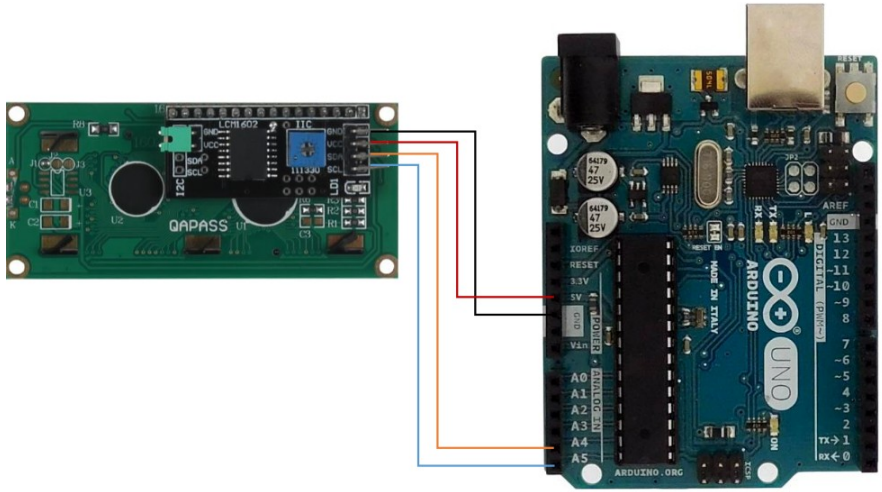
- Cette carte est dotée de 4 broches, IN1, IN2, IN3 et IN4
- A partir du tableau ci dessous, faites tourner le moteur dans un sens, puis dans l'autre !

Phase	IN1	IN2	IN3	IN4
1	HIGH	LOW	LOW	LOW
2	LOW	HIGH	LOW	LOW
3	LOW	LOW	HIGH	LOW
4	LOW	LOW	LOW	HIGH

Manipulation 3
Écran LCD

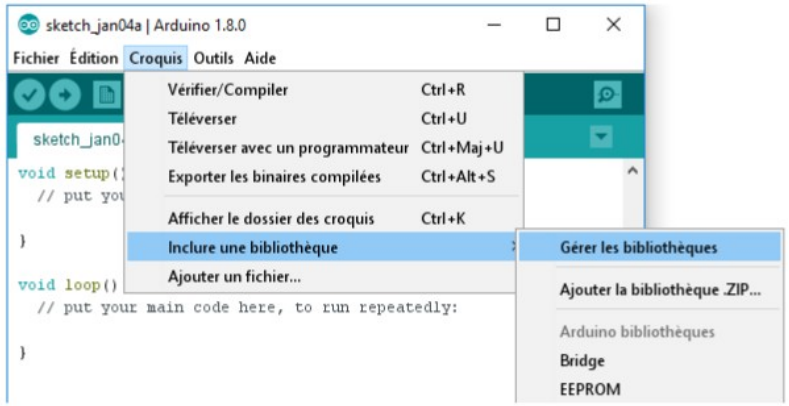
Ecran LCD en I²C

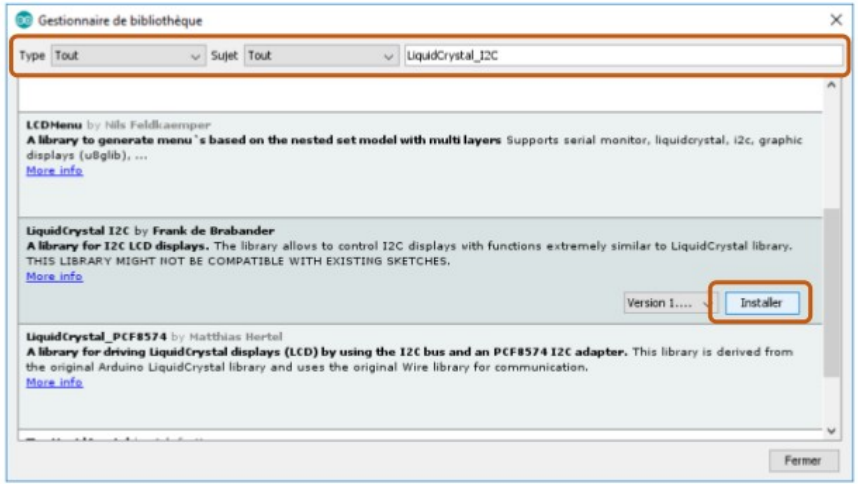
- Le kit fourni contient un écran LCD en I²C, pour le brancher nous allons utiliser le câblage suivant :



Écran LCD en I²C

- Rendez-vous ensuite dans « Gérer les bibliothèques » et installez LiquidCrystal_I2C :





Écran LCD en I²C

- Voici un exemple de programme qui affiche le temps écoulé :
  - 0x27 est l'adresse I²C du périphérique
  - 16 et 2 sont le nombre de lignes et colonnes
  - *setCursor* permet de se placer sur une ligne et une colonne données
- [lcd.ino]

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{
  lcd.init();
}

void loop()
{
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Temps:");
  lcd.setCursor(0, 1);
  lcd.print(millis());
  lcd.print(" ms");
}
```

A vous de jouer !

- Le kit contient un joystick qui fournit une position X, Y et un bouton appuyable (qui sont des entrées sorties)
- Faites le fonctionner, et faites un programme qui déplace un « X » sur l'écran en fonction du joystick !

Manipulation 4
Communication avec Python

Communication avec Python

- Il est possible de communiquer avec Python à travers l'interface série
- Un premier exemple qui permet d'envoyer des messages du code python vers l'écran LCD :
  - [lcd-receive.ino]
  - [send.py]

Communication avec Python

- Voici un exemple qui fonctionne dans l'autre sens, qui échantillonne le JoyStick et le lis depuis Python :
  - [joystick-send.ino]
  - [joystick.py]

Récupérer des valeurs

- Le programme [read.py] permet de récupérer des valeurs envoyées par *Serial.println()* et les stocke finalement dans un fichier CSV
- Le programme [plot.py] utilise la bibliothèque *matplotlib* pour dessiner une courbe des valeurs échantillonnées

A vous de jouer !

- En vous aidant des exemples précédent, ainsi que de [date.py], un programme Python qui affiche la date, ecrivez un programme qui affiche la date sur la première ligne de l'écran LCD, et l'heure sur la deuxième ligne

  On devrait voir les secondes bouger !

Il y a encore plein de choses dans le kit que vous n'avez pas utilisé, n'hésitez pas à les découvrir !