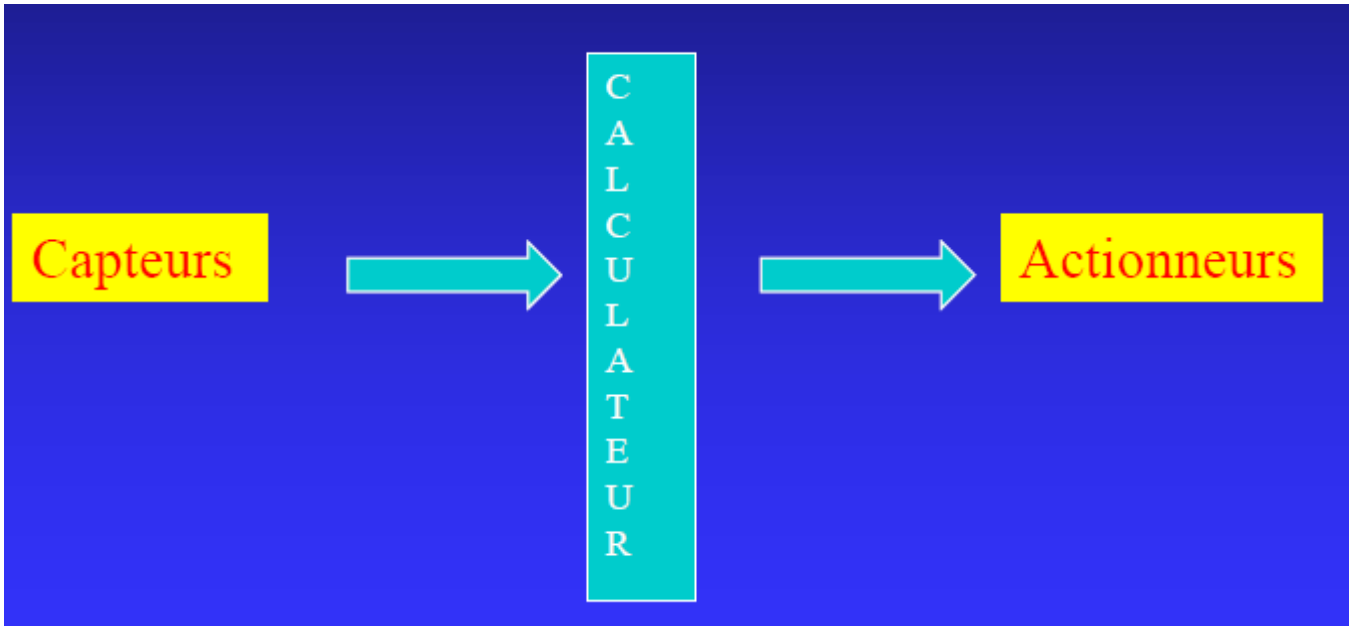
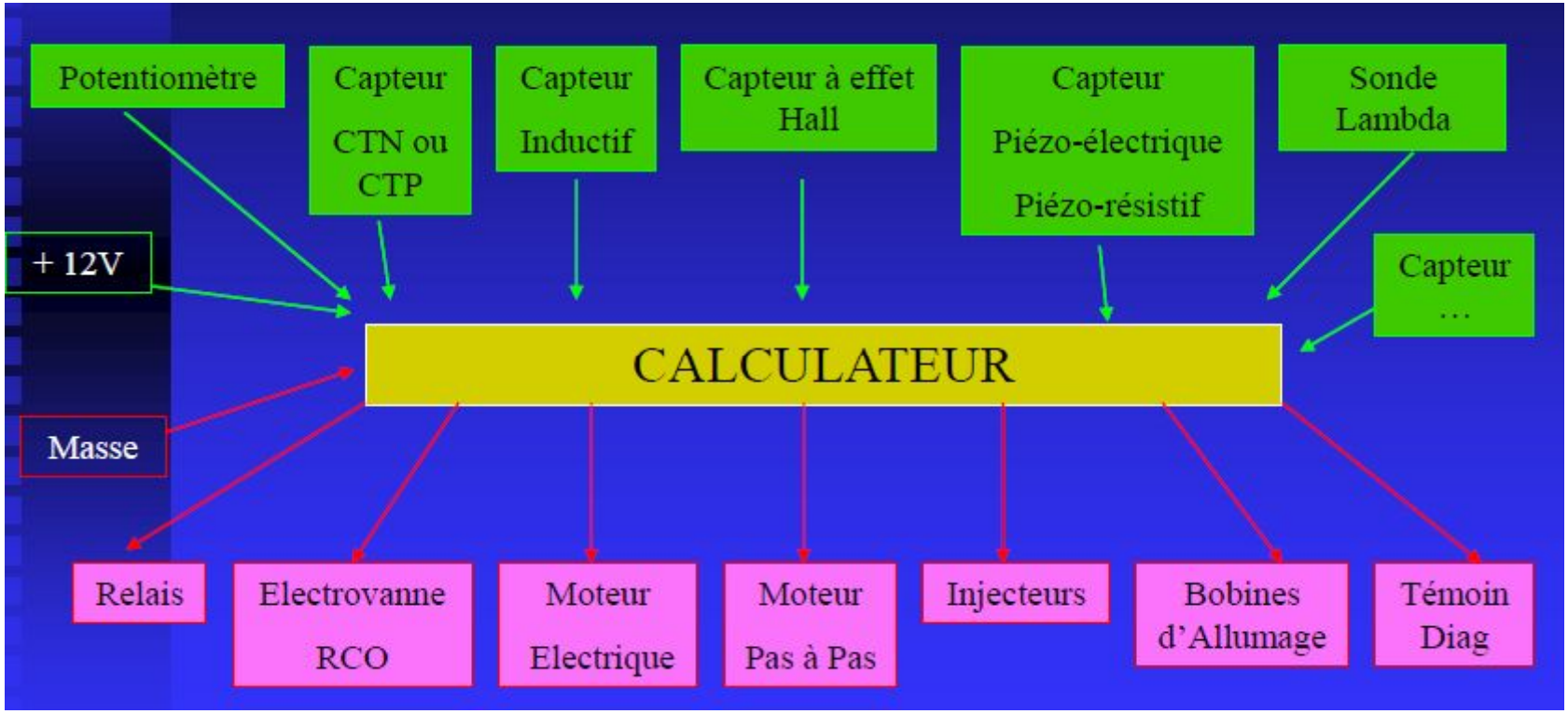
GENERALITES SUR LES SYSTEMES PROGRAMMABLES

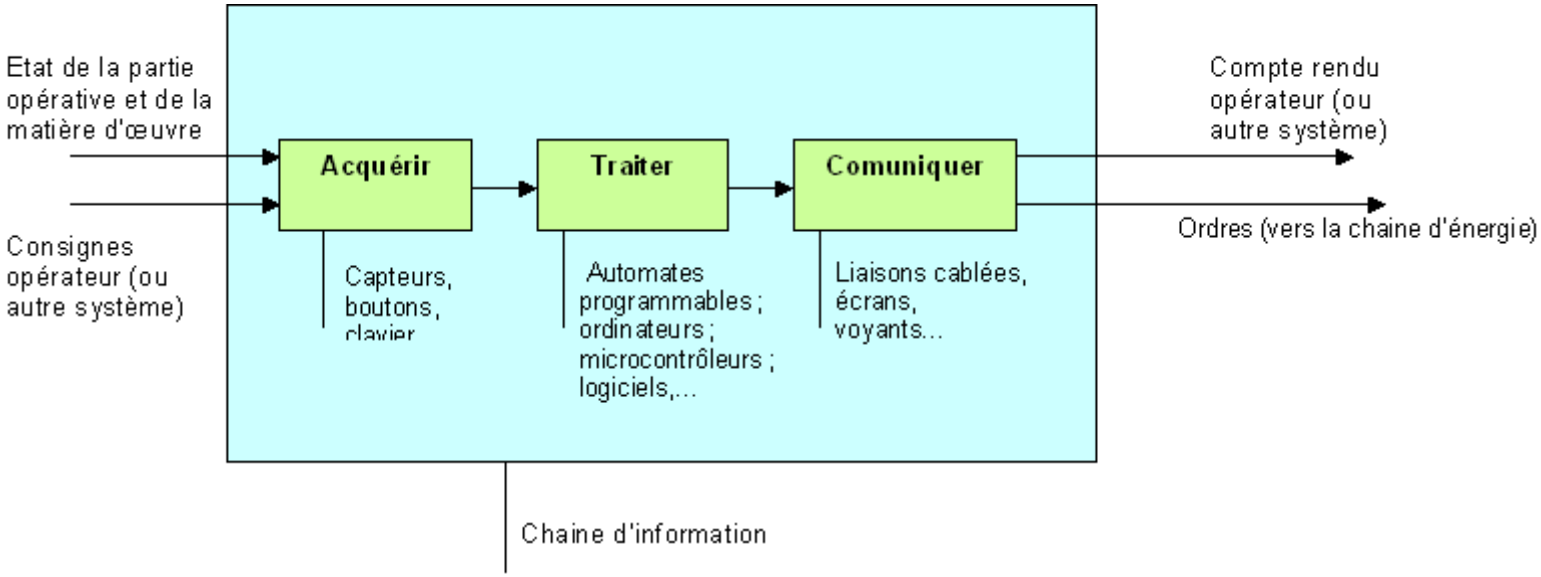


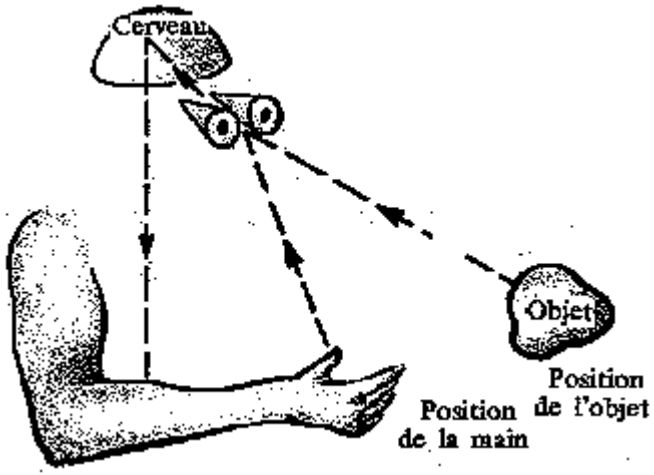
GENERALITES SUR LES SYSTEMES PROGRAMMABLES



GENERALITES SUR LES SYSTEMES PROGRAMMABLES

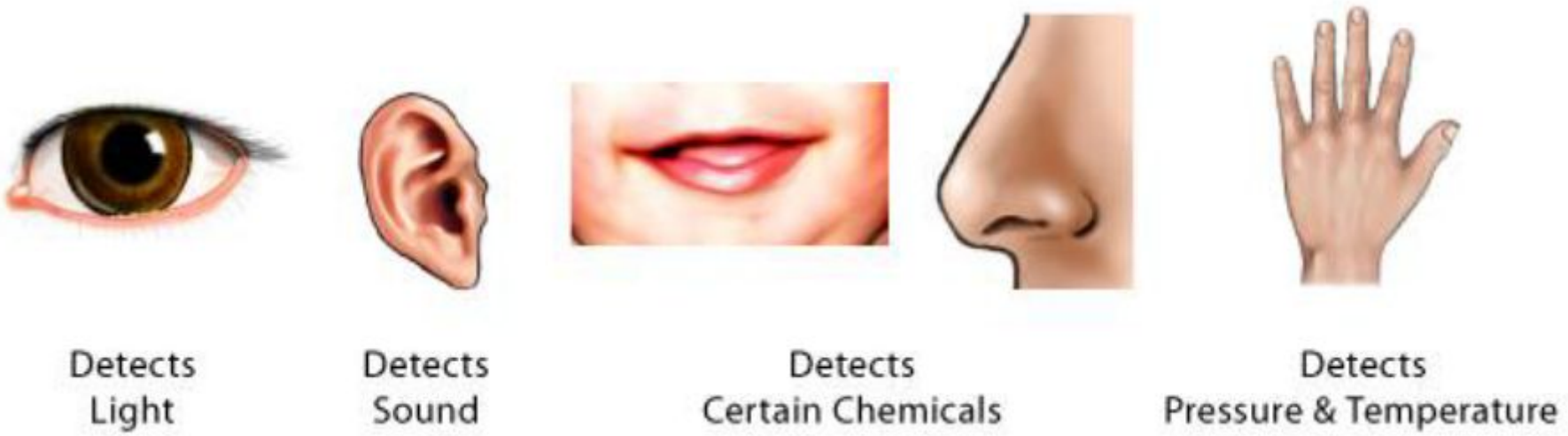
Comparaison Homme-machine





GENERALITES SUR LES SYSTEMES PROGRAMMABLES

CONCEPTS: INPUT VS. OUTPUT



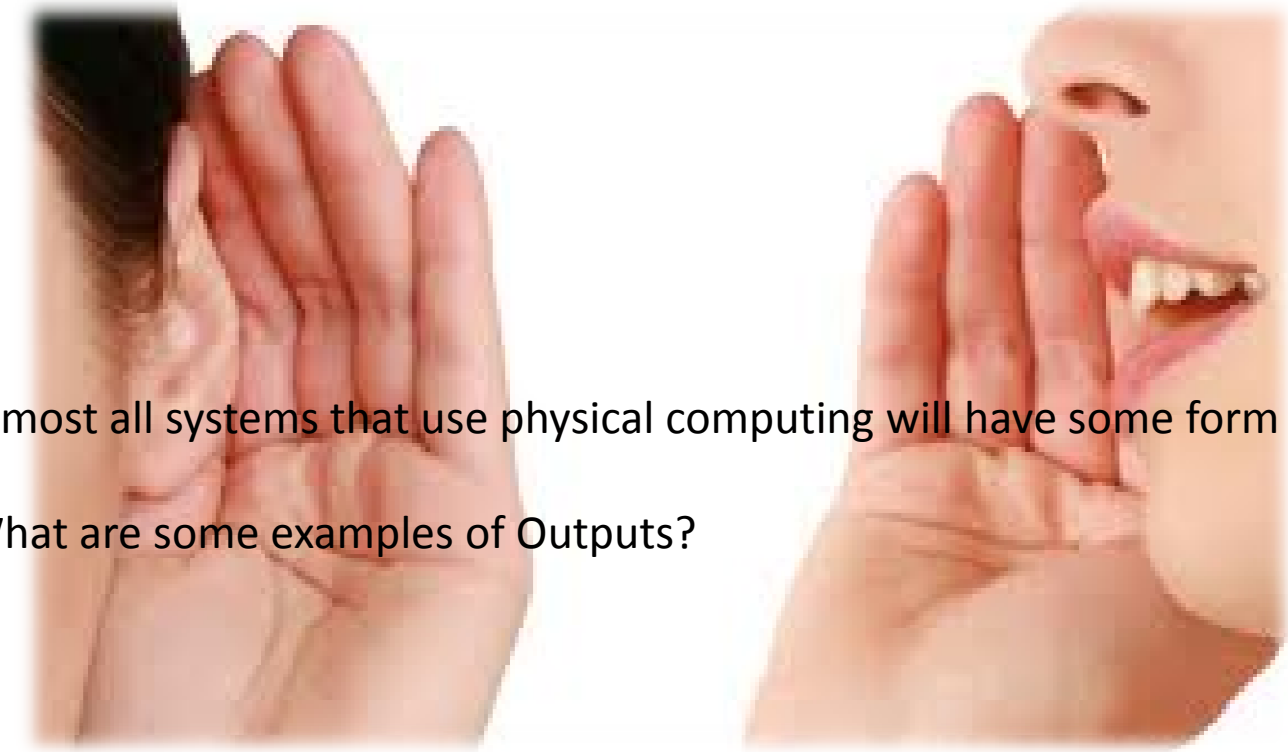
GENERALITES SUR LES SYSTEMES PROGRAMMABLES

Concepts: INPUT vs. OUTPUT

Referenced from the perspective of the microcontroller (electrical board).

Inputs is a signal / information going into the board.

Output is any signal exiting the board.



Almost all systems that use physical computing will have some form of output

What are some examples of Outputs?

Concepts: INPUT vs. OUTPUT

Inputs est un signal / information appliqué à l'entrée d'un système.

Output est le signal produit à la sortie d'un système

Exemples: Bottons Poussoirs, fins de course, Capteur de températre…	Exemples: LEDs, moteur DC, servo motor, a piezo buzzer, relais, RGB LED …etc

TRANSDUCTEURS



Les transducteurs permettent de changer un énergie en une autre forme d'énergie. Parmi les transducteurs nous avons:

- Les capteurs: exemple un microphone
- Les actionneurs : Un moteur
- autres : une lampe à incandescence

Actionneurs (Actuators)

Un actionneur est un exemple particulier de transducteur. Souvent il traduit un signal électrique en une variation d'un phénomène naturel.

Exemple d'actionneurs

- Moteur électrique DC ou AC
- Relais
- Electrovanne
- ….etc

Capteurs (Sensors)

Un capteur est un équipement permettant de traduire les variations d'un phénomène naturel en un signal souvent électrique.

Exemple:

- Bouton Poussoir
- Capteur de température
- Photorésistance
- …etc

Thermometers

Les actionneurs

Les actionneurs exécutent les actions (déplacement, émission de chaleur, de lumière, de son etc.) lorsque l'ordre leur en est donné par la partie commande.

Moto réducteur

Ventilateur

Afficheurs à LED et LCD

Voyants

Buzzers

Vérins électriques

Les différents systèmes programmables

- **Les circuits spécialisés ou ASIC (Application Specific Integrated Circuit) :**

Les circuits spécialisés sont des circuits spécialisés dès leur conception pour une application donnée.

Exemples : DSP (*Digital Signal Processing*), co-processeur arithmétique, processeur 3-D, contrôleur de bus, ...

Source : Texas Instruments

Source : NVidia

Avantages :

- Très rapide
- Consommation moindre
- Optimisé pour une application

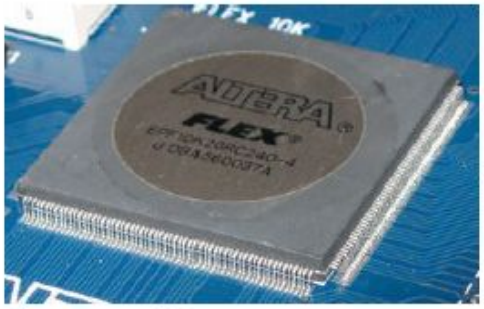
Inconvénients :

- Faible modularité
- Possibilité d'évolution limité
- Coût

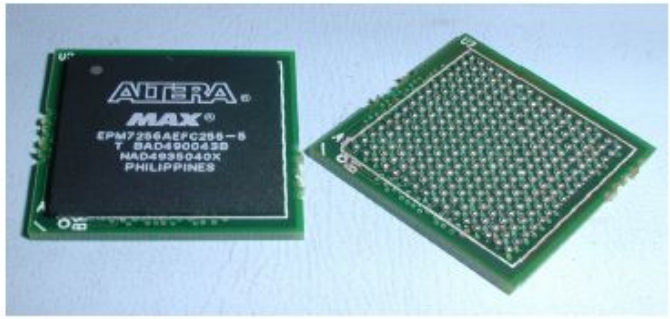
Les différents systèmes programmables

Les systèmes en logique programmée et/ou en logique programmable sont connus sous la désignation de PLD (*programmable logic device, circuit logique programmable*)

- **FPGA** (*field-programmable gate array, réseau de portes programmables in-situ*),
- **PAL** (*programmable array logic, réseau logique programmable*),
- ...



Source : Altera



Source : Altera

« Un circuit logique programmable, ou réseau logique programmable, est un circuit intégré logique qui peut être reprogrammé après sa fabrication. Il est composé de nombreuses cellules logiques élémentaires pouvant être librement assemblé.» (Wikipédia)

Avantages :

- Forte modularité
- Rapidité

Inconvénients :

- Mise en oeuvre plus complexe
- Coûts de développement élevé

Les différents systèmes programmables

- **Les systèmes micro-programmés :**

Les micro-contrôleurs sont typiquement des systèmes micro-programmés.

Micro-contrôleur Microchip
PIC16F690 en boîtier DIL20

Un **micro-contrôleur** est un :

« Circuit intégré comprenant essentiellement un microprocesseur, ses mémoires, et des éléments personnalisés selon l'application. » (Arrêté français du 14 septembre 1990 relatif à la terminologie des composants électroniques.)

Un micro-contrôleur contient un microprocesseur.

Avantages :

- Mise en oeuvre simple
- Coûts de développement réduits

Inconvénients :

- Plus lent
- Utilisation sous optimale

Microprocesseur vs. Microcontrôleur

Microprocessor

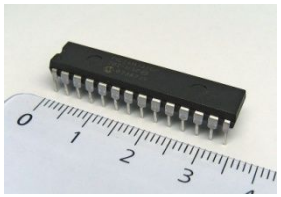
- CPU is stand-alone, RAM, ROM, I/O, timer are separate
- designer can decide on the amount of ROM, RAM and I/O ports.
- expensive
- versatility
- general-purpose
- High processing power
- High power consumption
- Instruction sets focus on processing-intensive operations
- Typically 32/64 – bit
- Typically deep pipeline (5-20 stages)

Microcontroller

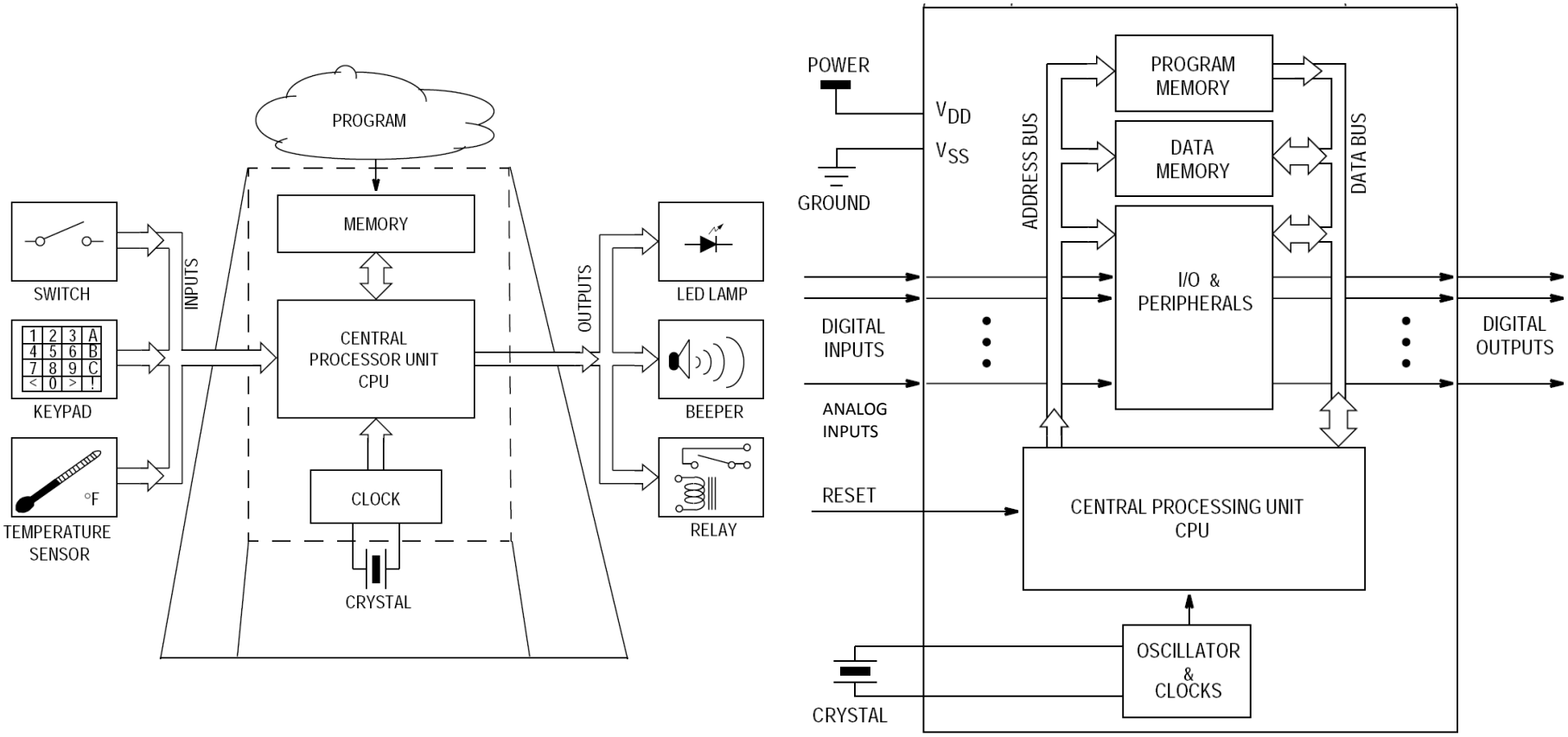
- CPU, RAM, ROM, I/O and timer are all on a single chip
- fixed amount of on-chip ROM, RAM, I/O ports
- for applications in which cost, power and space are critical
- single-purpose (control-oriented)
- Low processing power
- Low power consumption
- Bit-level operations
- Instruction sets focus on control and bit-level operations
- Typically 8/16 bit
- Typically single-cycle/two-stage pipeline

Some Popular Microcontrollers…

- 8051
- Microchip Technology PIC
- Atmel AVR
- Texas Instruments MSP430 (16-bit)



What is a Microcontroller?

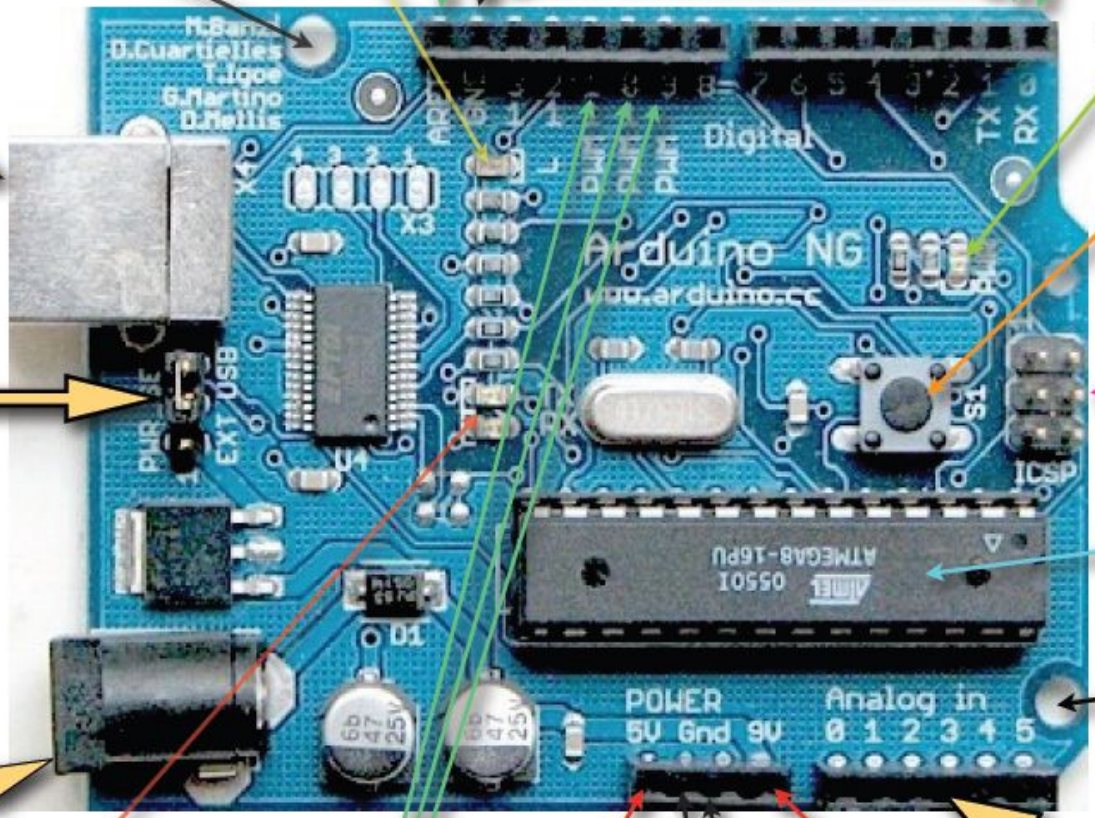


What is the difference between a '*Digital* Input' and an '*Analog* Input'?

LES MICROCONTROLEURS ARDUINO

Description de la carte

Version USB



Référence analogique (AREF)

LED jaune de test

Masse des connecteurs numériques (GND)

Port Série In/out
RX TX (supprime une entrée-sortie numérique si utilisé)

Trou de fixation par vis
(fixer sur un support non conducteur)

13 entrées/sorties numériques
numérotées de 1 à 13

LED verte témoin d'alimentation

Port USB:
pour le transport des données et pour l'alimentation électrique

Bouton de **réinitialisation** (reset)

Connecteurs pour le téléchargement du système d'exploitation du microcontrôleur

3 broches, avec **un cavalier** de sélection d'alimentation:

-cavalier sur les deux broches du haut: alimentation par le port USB, comme sur cette photo

-cavalier sur les deux broches du bas si alimentation par connecteur ci-dessous

Microcontrôleur

Trou de fixation par vis
(fixer sur un support non conducteur)

Connecteur d'alimentation de l'Arduino (si besoin)
9v à 12V
Connecteur 2,1 mm
avec le + au centre
courant continu
(DC)

2 LEDs jaunes ou vertes, actives lors d'un téléchargement de programme

6 entrées analogiques
numérotées de 0 à 5
fournissant une variation de 0 à 1023

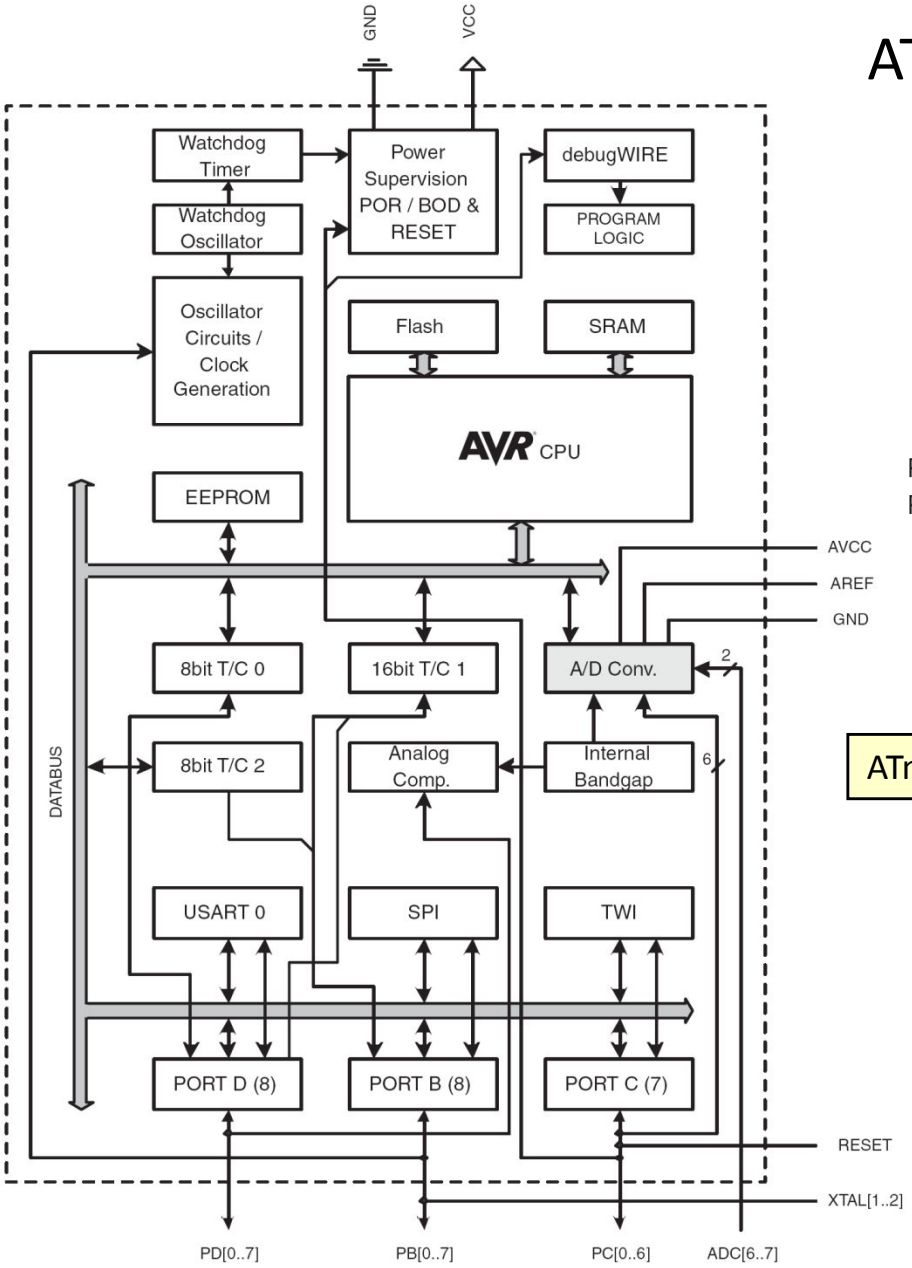
3 connecteurs numériques
pouvant également être utilisés en générateurs d'impulsions
(par ex: sortie analogique avec montage spécifique PWM numérotés de 0 à 2)

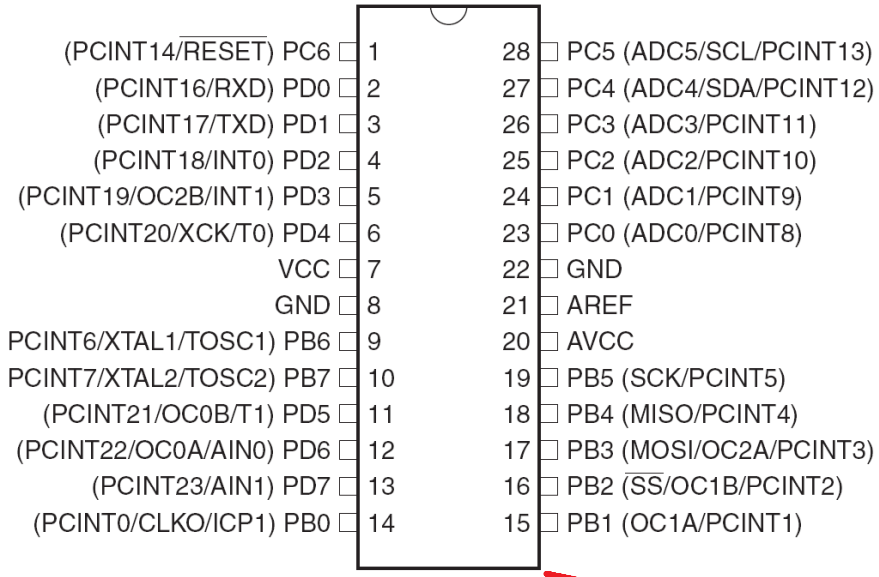
Masse (2 connecteurs)
(GND)

Source 5V faible puissance pour alimenter des montages divers comme des capteurs 0-5 V par exemple

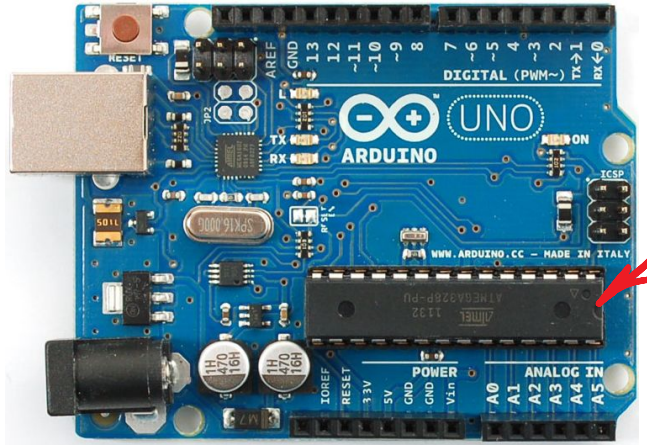
Source 9V faible puissance pour alimenter des montages divers

ATmega328 Internal Architecture





ATmega328 data sheet pp. 2, 5



http://www.adafruit.com/index.php?main_page=popup_image&pID=50

ATmega328 Features

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1K Bytes EEPROM
  - 512/1K/1K/2K Bytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 4 MHz@1.8 - 5.5V, 0 - 10 MHz@2.7 - 5.5.V, 0 - 20 MHz @ 4.5 - 5.5V
- Power Consumption at 1 MHz, 1.8V, 25°C
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)

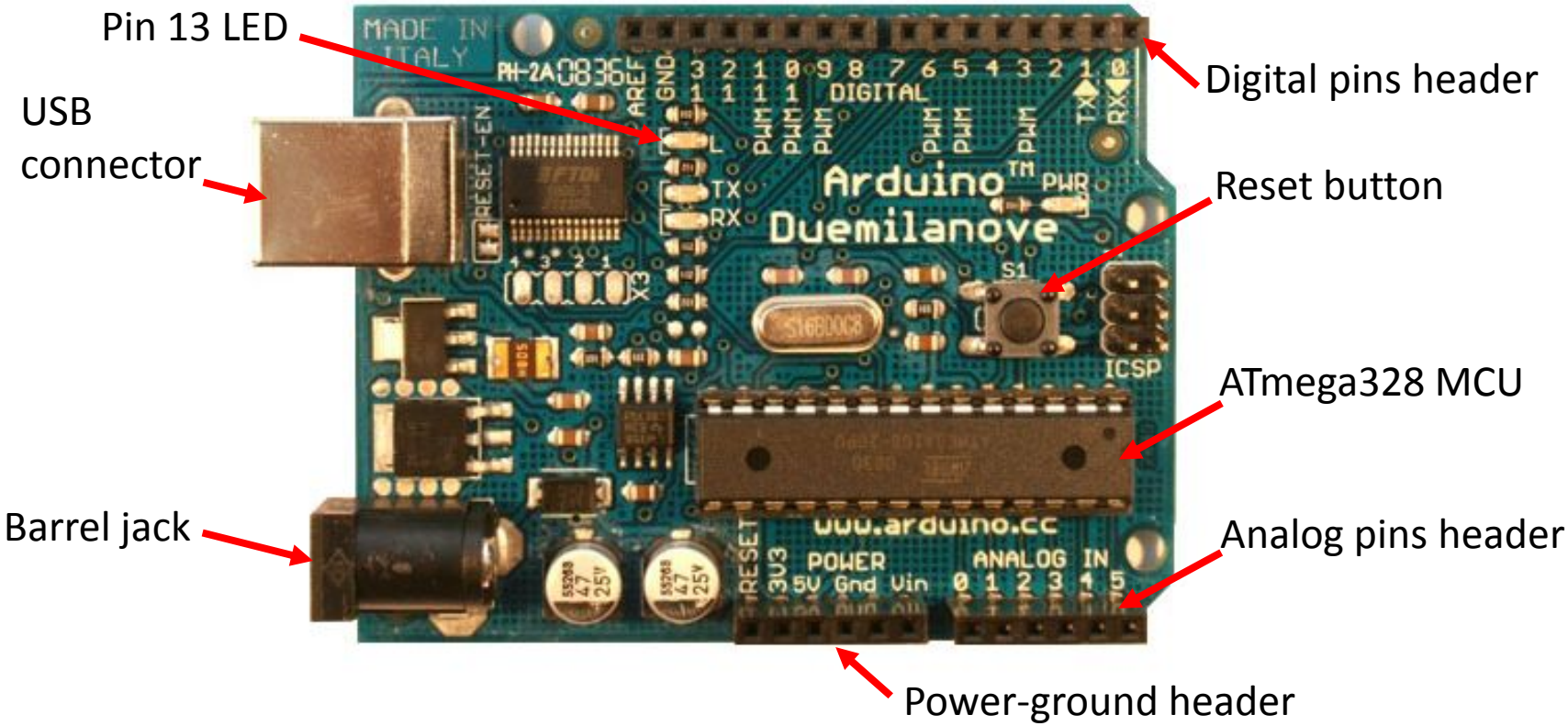
ATmega328 data sheet p. 1

http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet.pdf

Arduino Duemilanove

http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove

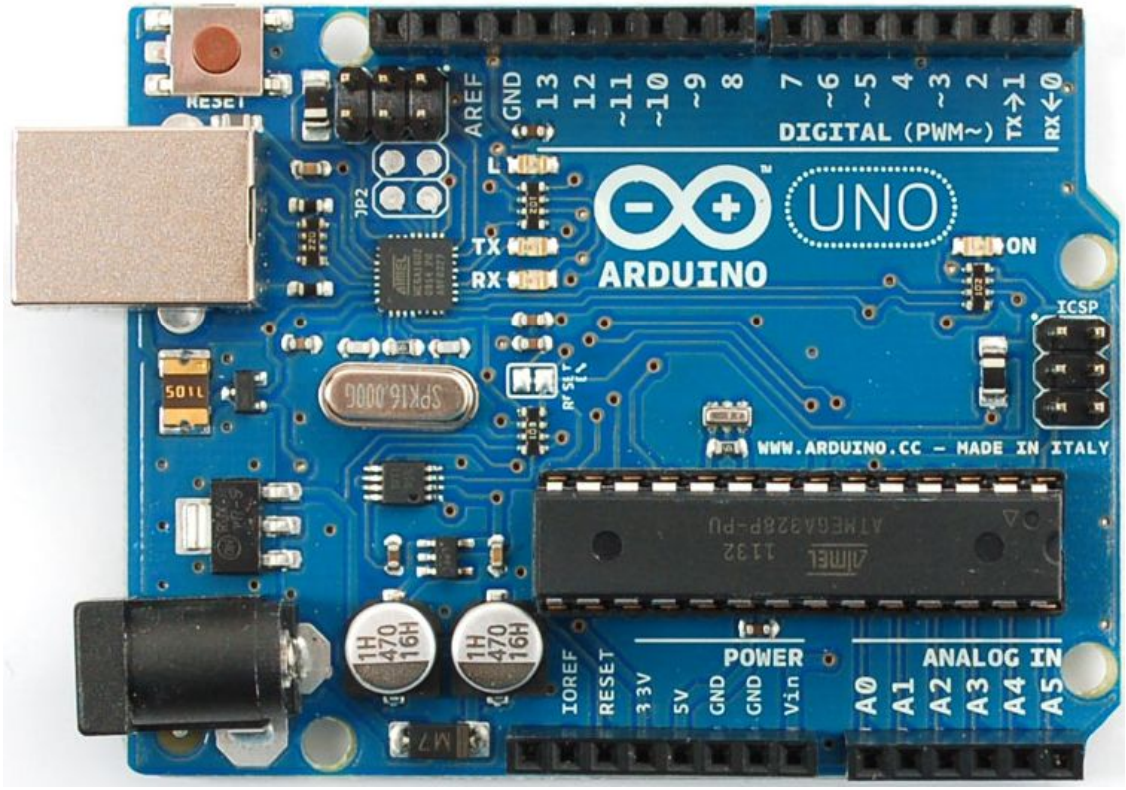
See the handout: **Arduino_ATmega328_pin_mapping_and_schematic**



http://arduino.cc/en/uploads/Main/ArduinoDuemilanove.jpg

Arduino Uno R3

ATmega16u2 replaces FT232RL for USB-serial communication



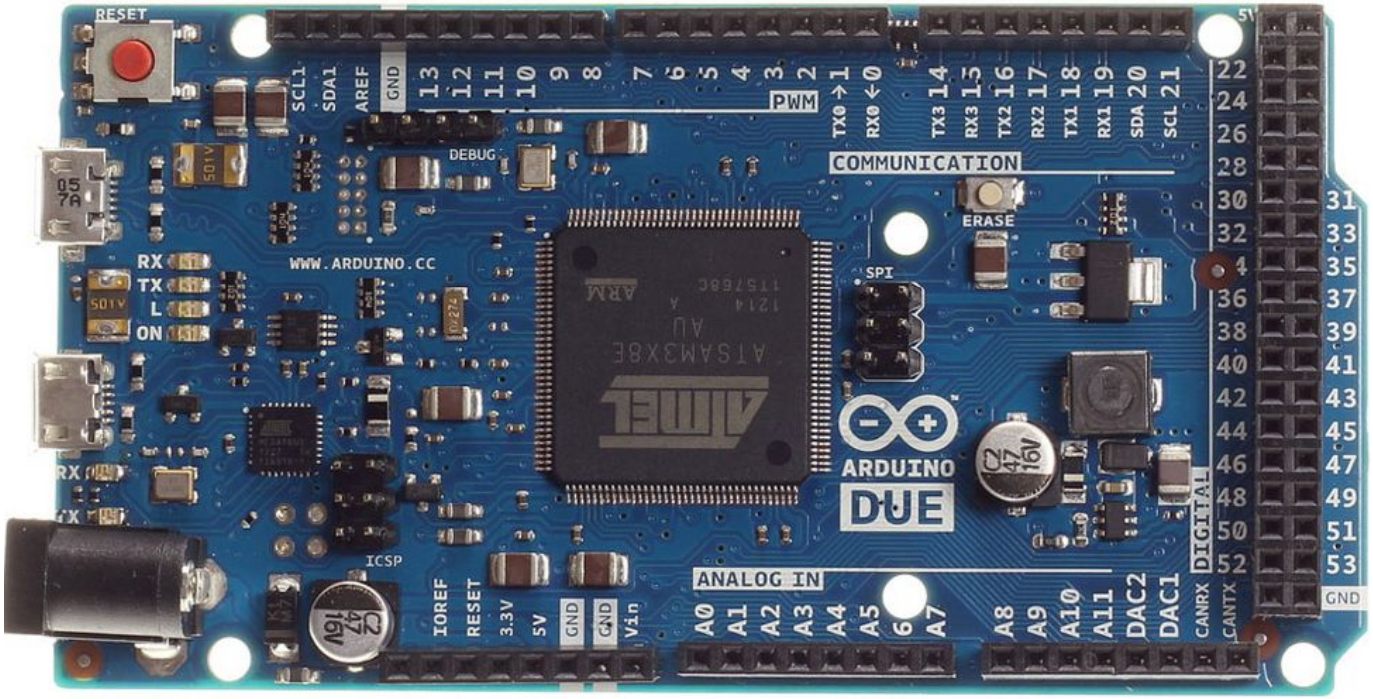
http://www.adafruit.com/index.php?main_page=popup_image&pID=50

See: http://learn.adafruit.com/arduino-tips-tricks-and-techniques/arduino-uno-faq

Arduino Due

Note: **3.3 V** !!

Atmel SAM3X8E processor (32 bit ARM Cortex M3 architecture, 84MHz)



http://www.adafruit.com/index.php?main_page=popup_image&pID=1076

See: http://arduino.cc/en/Main/ArduinoBoardDue

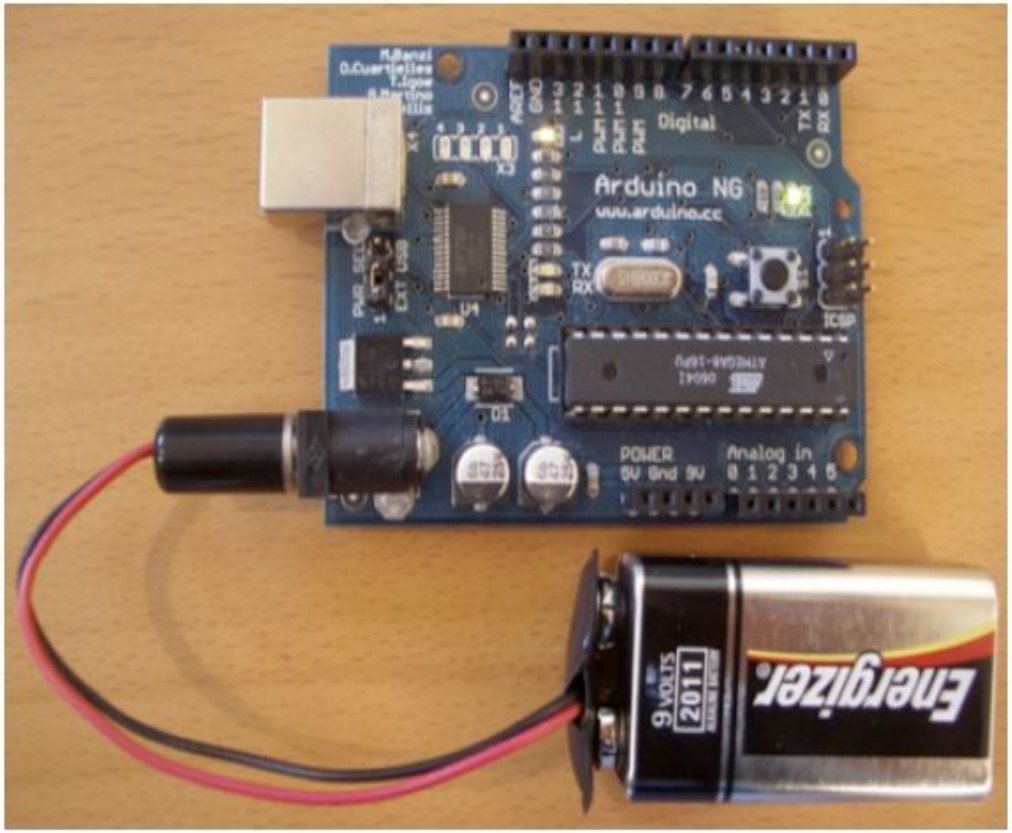
Arduino Duemilanove/Uno Features

Microcontroller	ATmega168/328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz

http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove

Alimenter l'Arduino

en mode autonome sans ordinateur



Avec une pile 9V et un connecteur
C'est une solution très pratique et sûre pour éviter tout problème avec le port USB de son ordinateur.

Avec un adaptateur secteur 9 à 12 V

Connecteur 2,1 mm avec le + au centre, courant continu (DC)

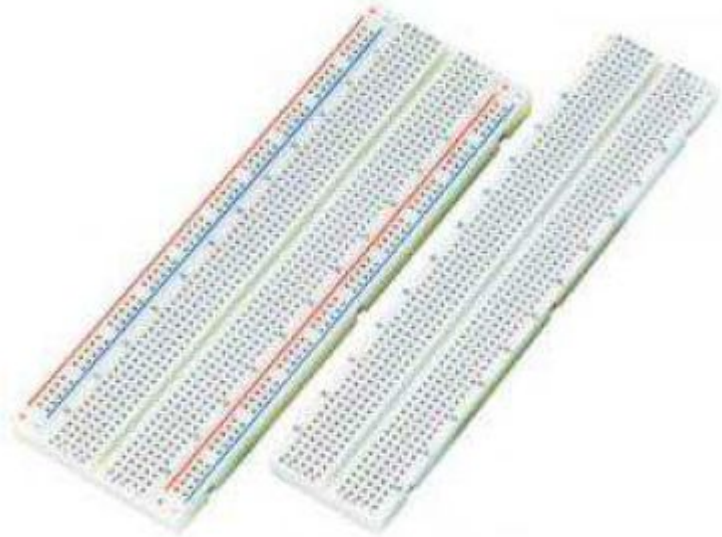
Ne pas oublier de déplacer le cavalier du coté de l'alimentation, comme indiqué en page 1

EQUIPEMENTS SUPPLEMENTAIRES

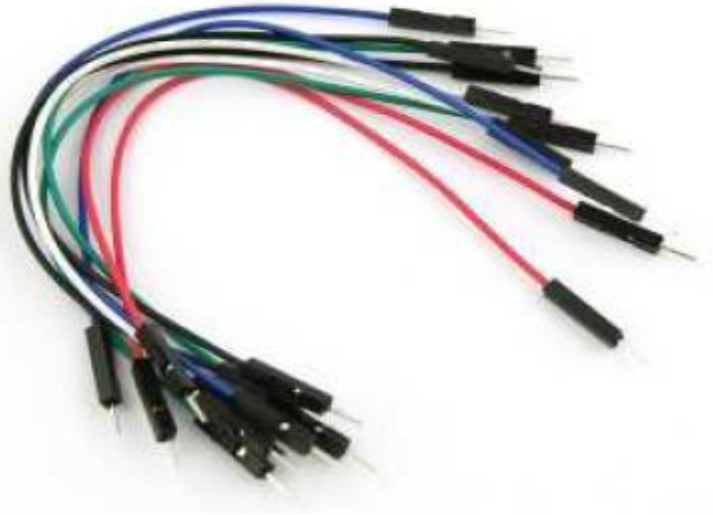
Une Arduino Uno ou Duemilanove

Un câble USB A mâle/B mâle

Une BreadBoard (plaque d'essai)

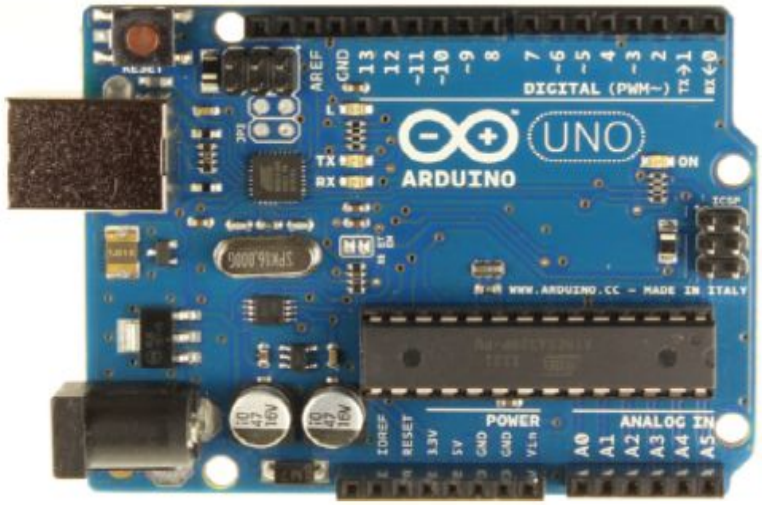


Un lot de fils pour brancher le tout !



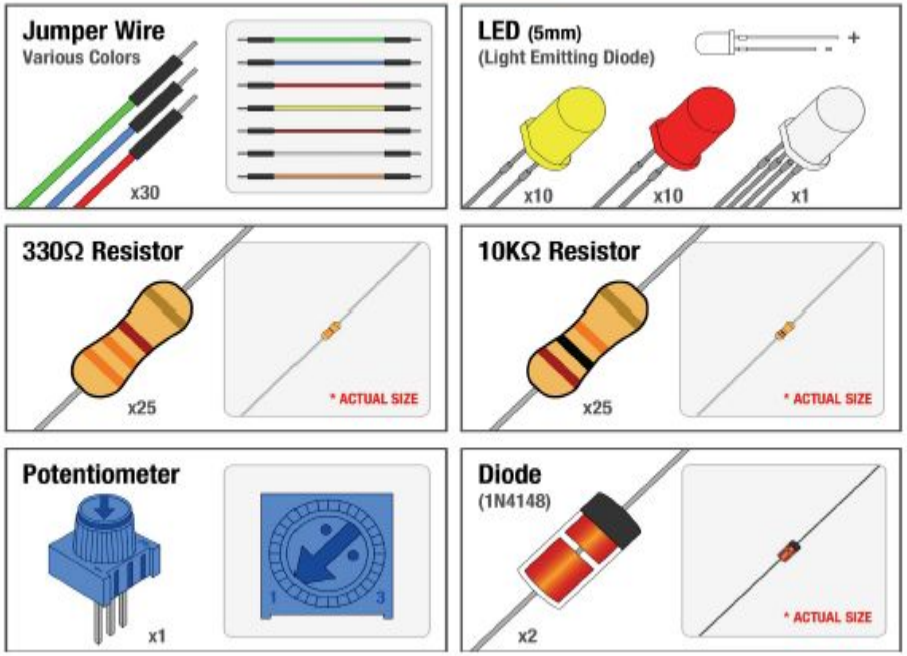
Arduino Basics

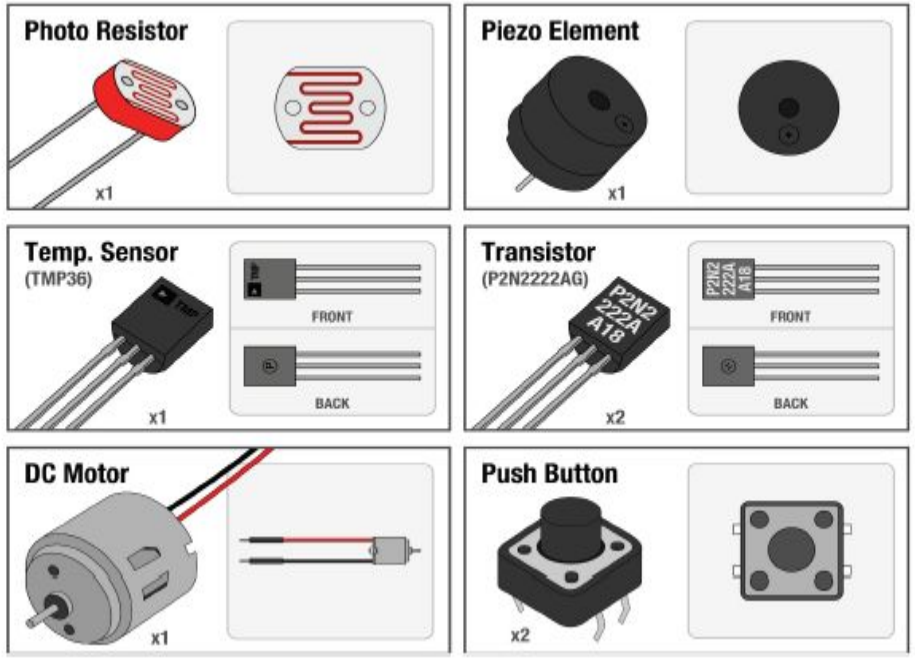
Arduino Uno Board



Breadboard

Sensors and Actuators



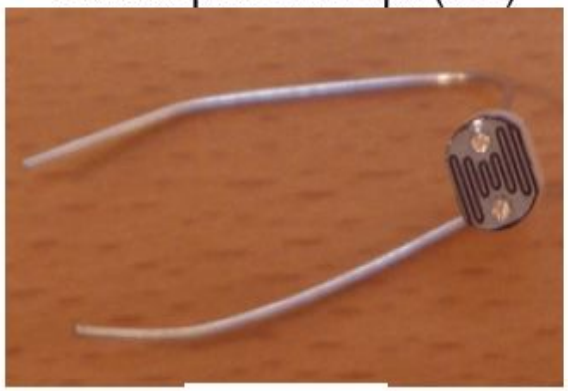


EXEMPLES ENTREES SORTIES POUR ARDUINO

L'interrupteur

L'interrupteur ouvre ou ferme un circuit. Il y a toutes sortes d'interrupteurs.
Sur l'Arduino, utiliser un interrupteur pour déclencher un événement nécessite d'utiliser un composant supplémentaire: une résistance de 10K ohms. Voir " Montages d'électronique interactive".

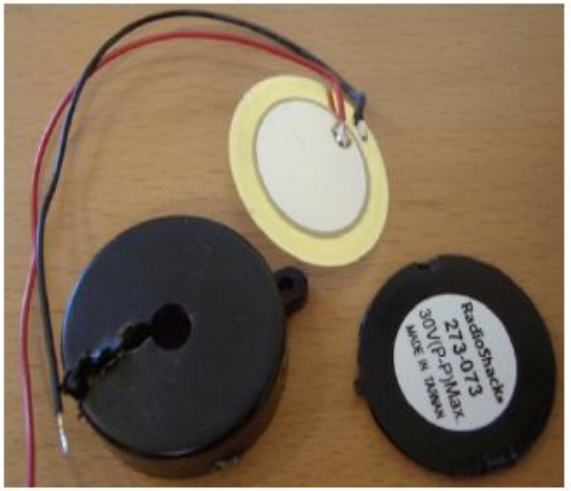
La cellule photo-électrique (LDR)



La cellule photo-électrique (LDR)

C'est une résistance variable, en fonction de la luminosité qu'elle reçoit. Sa résistance diminue quand elle reçoit de la lumière. On s'en sert donc de capteur de luminosité. Non polarisée. Pour lire sa valeur avec une Arduino, il faut également l'associer avec une résistance équivalente à sa résistance maxi (dans le noir) Voir " Montages d'électronique interactive".

Le piezo



Le transducteur piezo-électrique est un composant réversible: il peut aussi bien être utilisé en capteur de chocs ou de vibrations qu'en actionneur pouvant émettre des sons stridents parfois modulables.

EXEMPLES ENTREES SORTIES POUR ARDUINO

Le servo moteur

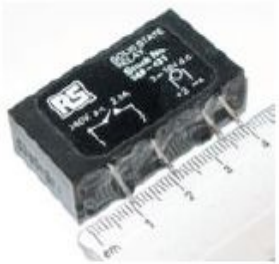
Le servo-moteur est un moteur (rotatif) qui peut effectuer des rotations très précises (dans une portion de tour seulement) et en un certain nombre de pas (de micro-déplacements). Il y a toutes sortes de servo moteurs.. Un des avantages des servo moteurs est sa possibilité de maintenir avec force une position donnée. On peut piloter des rotations avec l'Arduino, quelques fois directement avec la carte si le moteur n'est pas trop puissant, sinon en passant par un montage associé.

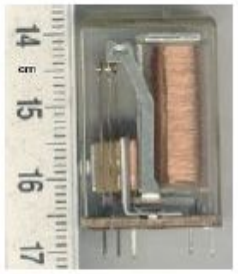
Le potentiomètre

Le potentiomètre

Le potentiomètre, rotatif comme ici, ou à glissière, est une résistance variable. Entre les extrémités, il y a la résistance maximale. La patte centrale est le curseur. C'est la résistance entre cette patte centrale et une extrémité que l'on peut faire varier en tournant le bouton. Le potentiomètre est donc un capteur. Il se branche sur les entrées analogiques de l'Arduino.
De très nombreux capteurs sont basés sur le principe de résistance variable et se cablent presque de la même façon: la cellule photo-électrique, le capteur de pression, le fil résistif, etc

Le relais





Le relais

Le relais est un composant à 4 broches minimum. C'est un interrupteur que l'on peut commander en envoyant un petit courant. Au repos, il est normalement fermé, ou normalement ouvert, selon le modèle. On peut s'en servir avec l'Arduino pour commander des machines en haute tension (230V par exemple), ou pour déclencher toute machine ou lumière.

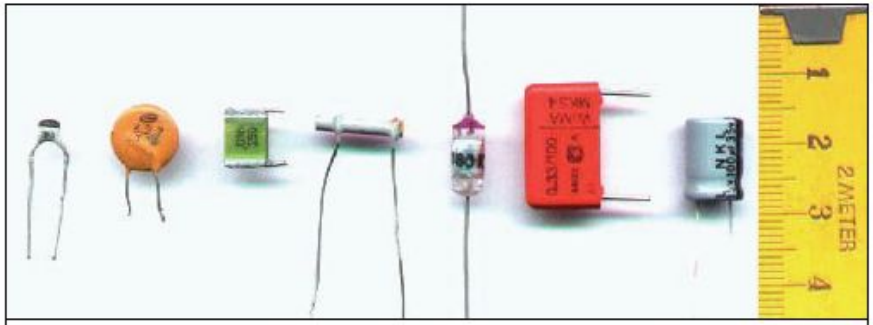
EXEMPLES COMOSANTS ELECTRONIQUES POUR ARDUINO

Savoir reconnaitre quelques composants électroniques, savoir reconnaitre leur symbole, connaitre leur usage, vous permettra de concevoir vos propres expériences et de progresser à partir de données trouvées sur le web. Vous saurez ainsi fabriquer quelques capteurs, et actionner quelques matériels.

La résistance

La résistance s'oppose au passage du courant, proportionnellement à sa " résistance" exprimée en Ohm. Un code de couleurs, ci dessous permet de reconnaître cette valeur.

Symbole européen



Les condensateurs

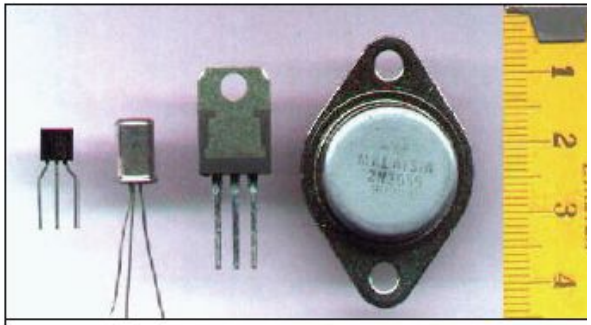
Les condensateurs peuvent stocker un peu de courant si on les charge, mais comme un tonneau percé, ils renvoient ce courant instantanément si ils sont branchés à un organe consommateur de courant.

Ils peuvent être polarisés ou non, dans des boitiers très divers. Leur valeur s'exprime en Farad (F)

non polarisé polarisé

Symbole

Reconnaissance de leur valeur:
A cause de la diversité des modèles, se reporter aux ressources sur le web



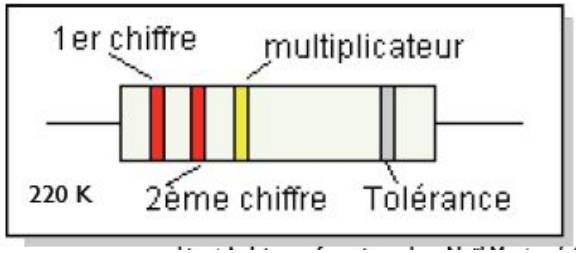
Le transistor

Le transistor est généralement utilisé comme une sorte de multiplicateur de puissance: lorsqu'on lui fait passer un courant faible, mais variable dans un de ses 3 pattes, il autorise proportionnellement le passage d'un gros courant dans une autre des 3 pattes.

transistor NPN transistor PNP

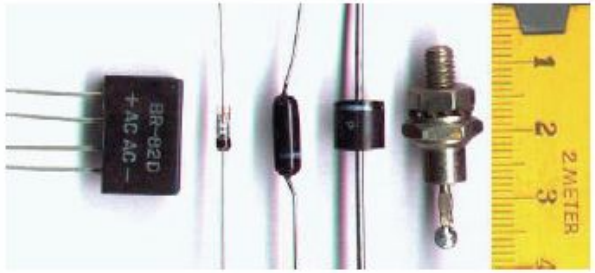
Code des couleurs des résistances

au delà de 1000 Ohms, on parle en KiloOhms, par exemple 10 K est 10 KiloOhms, puis en MegaOhms notés M



Chiffre	0	1	2	3	4	5	6	7	8	9	Or	Argent
Multiplicateur	1	10	10^2 100	10^3 1000	10^4 10000	10^5	10^6					
Précision	20%										5%	10%

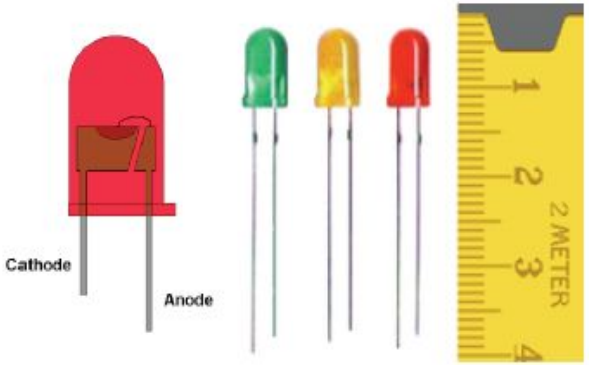
La diode



La diode

La diode ne laisse passer le courant que dans un seul sens. C'est un composant polarisé: on reconnait toujours son anneau coloré d'un coté du composant, correspondant à la cathode.

La LED



La diode électroluminescente (LED) émet de la lumière. Elle est polarisée: la patte "+" est la plus longue, l'autre patte est la patte "-" .
Les broches numériques de l'Arduino, lorsqu'elles sont configurées en sorties et qu'elles sont à l'état 1 ou haut (HIGH) , fournissent une tension de 5 volts, supérieure à ce que peut accepter une LED courante, sauf certaines LEDs. Les LED doivent donc être couplées en série avec une résistance.

La broche numérique 13 de l'Arduino est déja cablée en série avec une résistance de valeur moyenne pour une LED (1K ohm), on peut donc, dans la plupart des cas, directement lui brancher une LED, comme sur la photo-ci-dessous. Le ***moins*** sur la masse (nommée GND comme Ground) et le ***plus*** sur la broche 13. Il ne reste plus qu'à déclarer dans le programme que la broche 13 est configurée en sortie, et le tour est joué pour faire quelques essais. Si on a des LEDs particulières, ou si on est sur un autre port , on calcule et on rajoute une résistance.

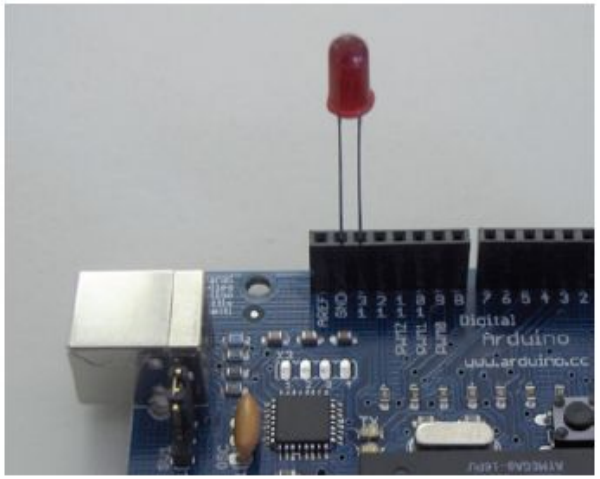
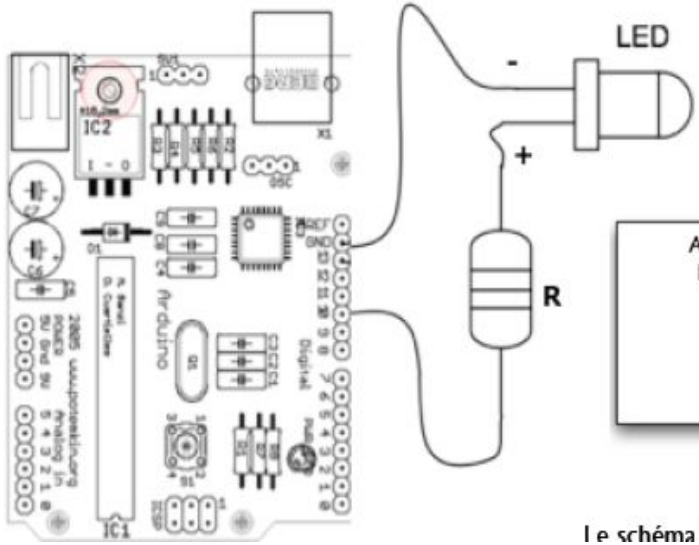
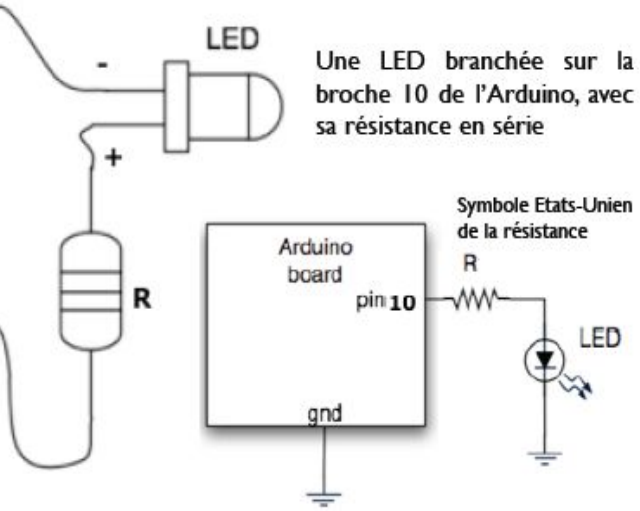


Tableau de puissance des LEDs

Couleurs	Tension de seuil ou Vf	If (mA)
Rouge	1,6 V à 2 V	6à20
Jaune	1,8 V à 2 V	6à20
Vert	1,8 V à 2 V	6à20
Bleu	2,7 V à 3,2 V	6à20
blanc	3,5 v à 3,8 v	30





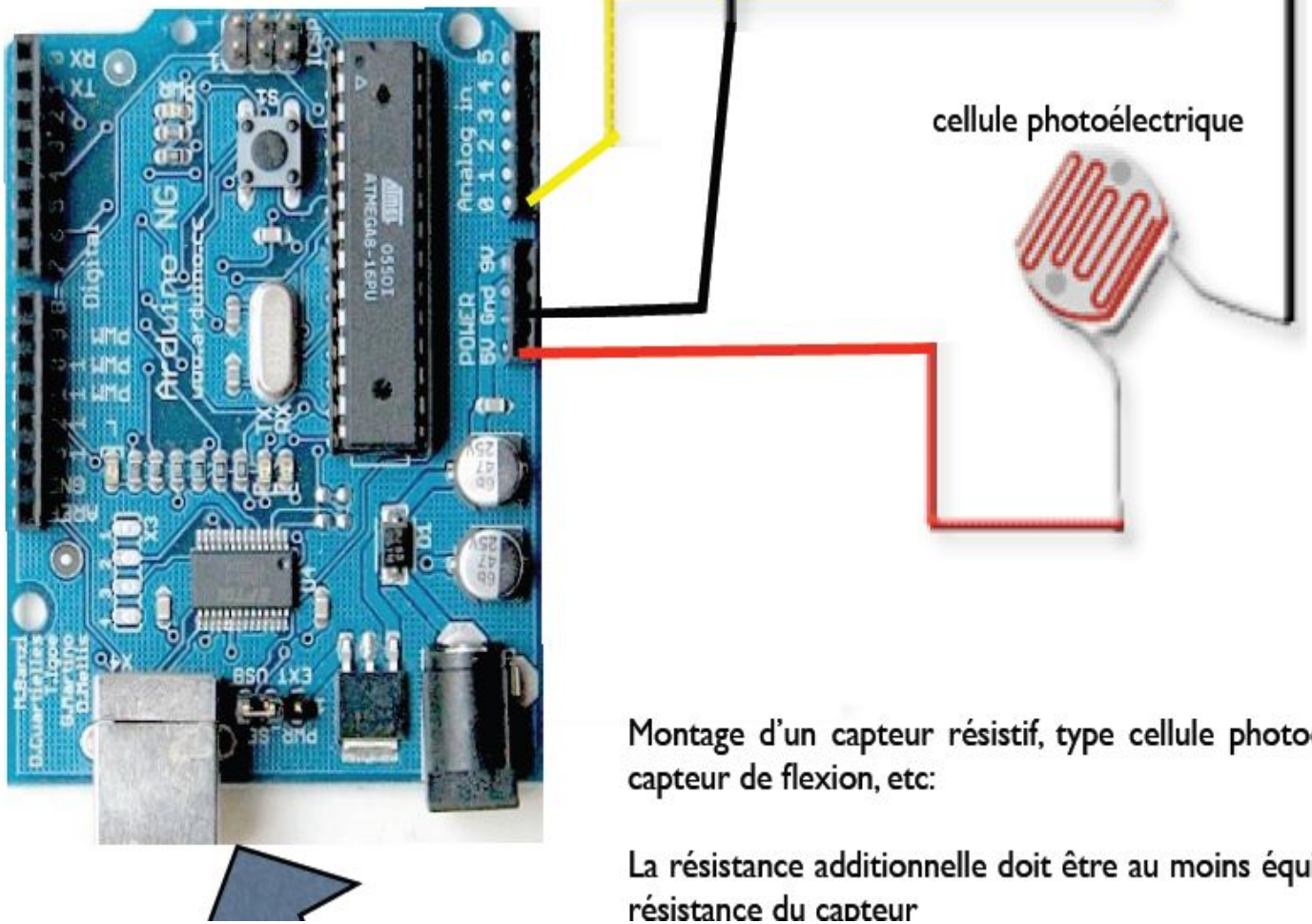
Une LED branchée sur la broche 10 de l'Arduino, avec sa résistance en série

Le schéma électronique correspondant

Comment calculer la résistance à appairer avec une LED ?

Résistance (en Ohms) = (5-Vf) / If

Montage d'un capteur résistif



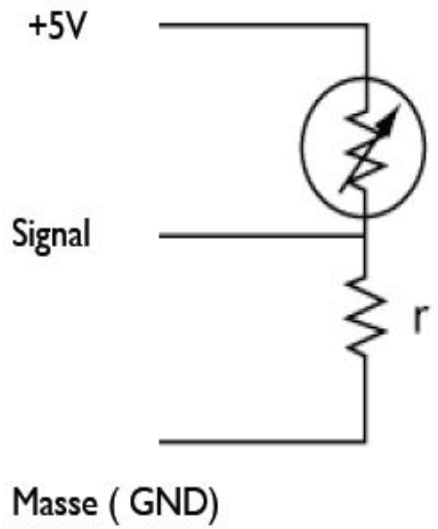


Schéma électronique (symboles U.S.A)

Montage d'un capteur résistif, type cellule photoélectrique, capteur de pression, capteur de flexion, etc:

La résistance additionnelle doit être au moins équivalente à la plus forte valeur de résistance du capteur

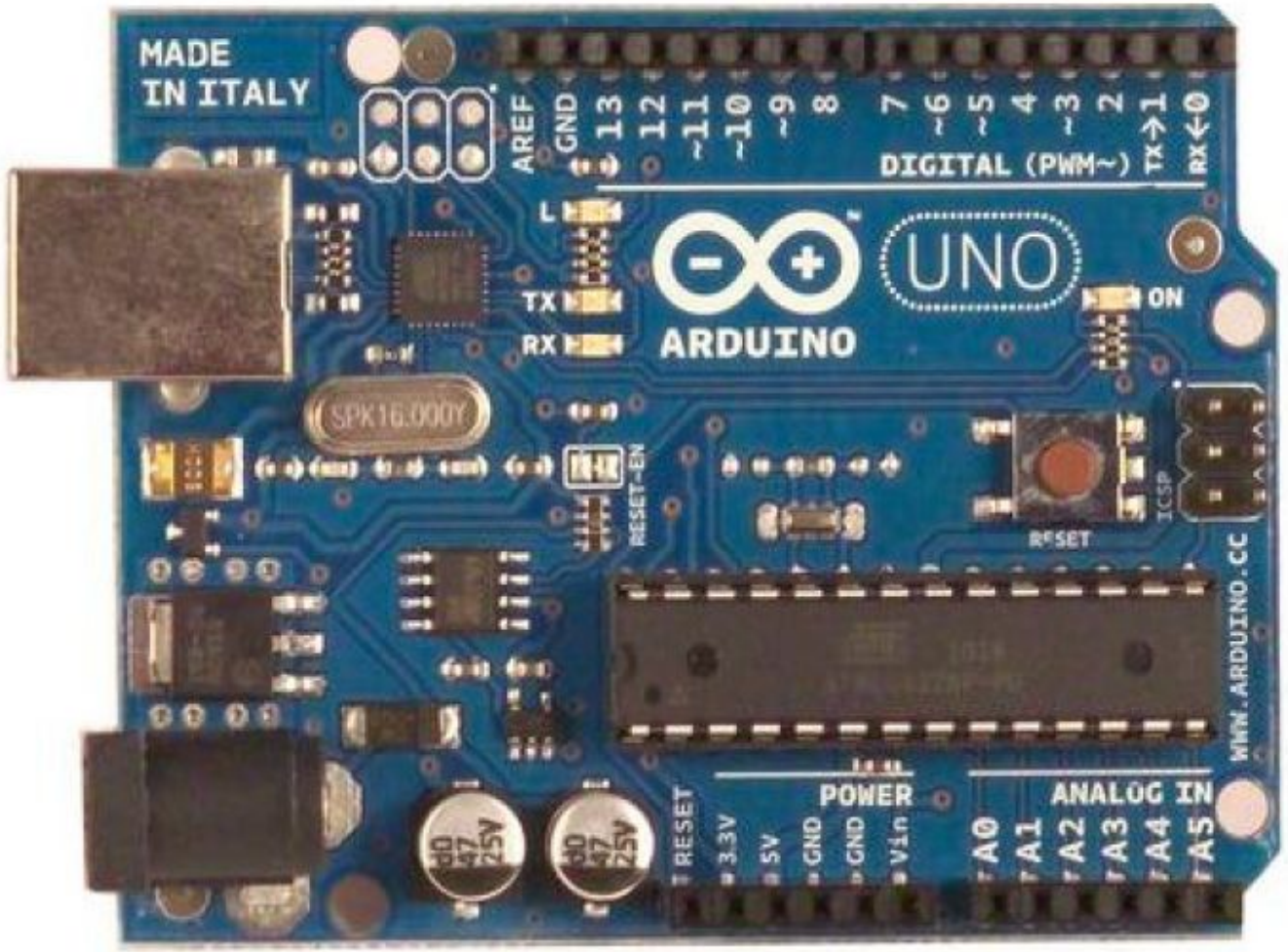

Success.

MATERIEL ET SOFTWARE POUR ARDUINO

Le système Arduino est composé de deux choses principales : le matériel et le logiciel.

- Le matériel

  Il s'agit d'une carte électronique basée autour d'un microcontrôleur Atmega du fabricant Atmel, dont le prix est relativement bas pour l'étendue possible des applications.



Un microcontrôleur est constitué par un ensemble d'éléments qui ont chacun une fonction bien déterminée. Il est en fait constitué des mêmes éléments que sur la carte mère d'un ordinateur :

1. La mémoire

   Il en possède 5 types :

   - La **mémoire Flash** : C'est celle qui contiendra le programme à exécuter.Cette mémoire est effaçable et ré-inscriptible.
   - **RAM** : c'est la mémoire dite "vive", elle va contenir les variables de votre programme. Elle est dite "volatile" car elle s'efface si on coupe l'alimentation du micro-contrôleur.
   - **EEPROM** : C'est le disque dur du microcontrôleur. Vous pourrez y enregistrer des infos qui ont besoin de survivre dans le temps, même si la carte doit être arrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme.
   - Les **registres** : c'est un type de mémoire utilisé par le processeur.
   - La mémoire **cache** : c'est une mémoire qui fait la liaison entre les registres et la RAM.

2. Le processeur

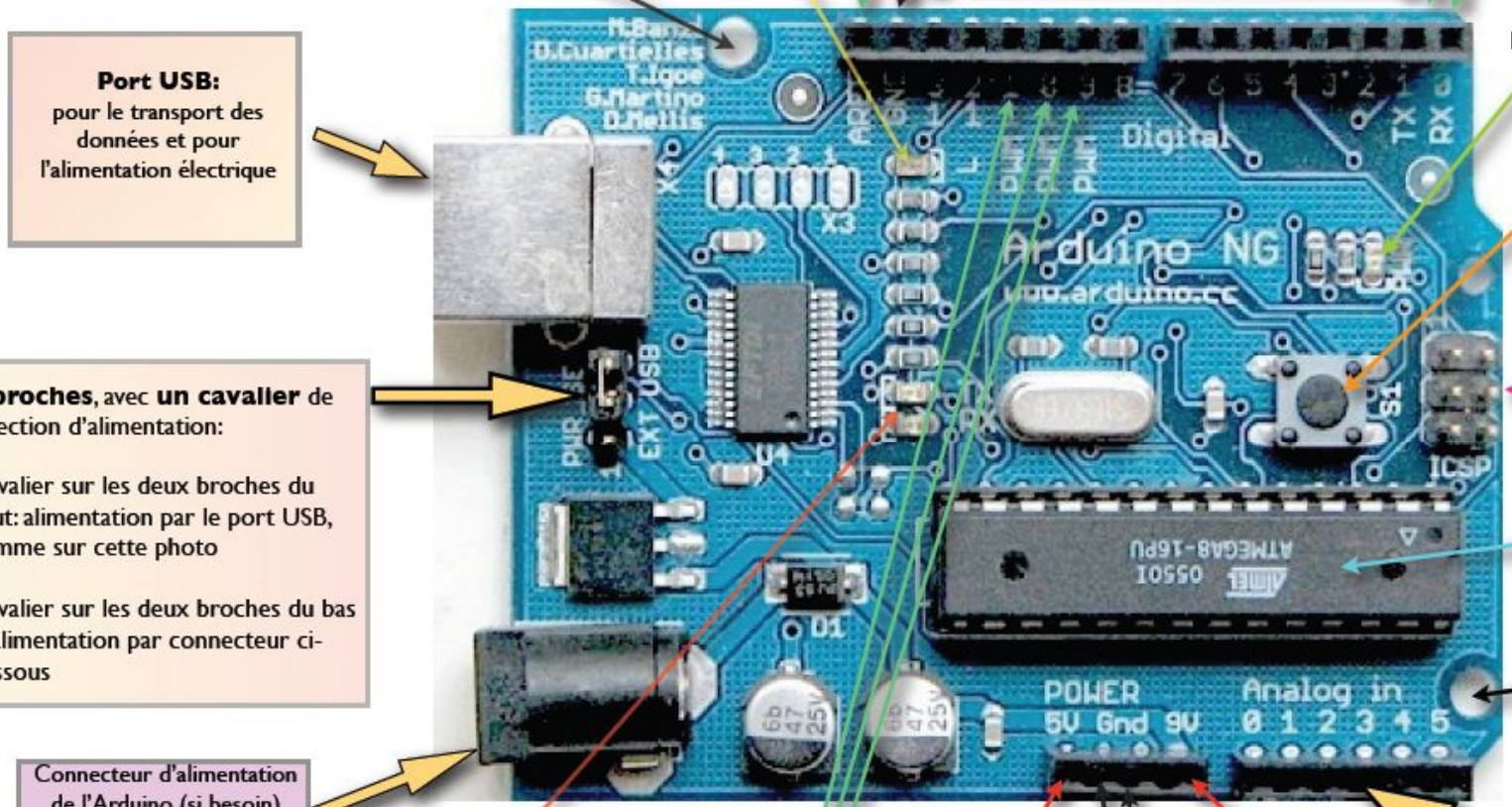
   C'est le composant principal du micro-contrôleur. C'est lui qui va exécuter le programme qu'on lui donnerons à traiter. On le nomme souvent le CPU.

Pour que le microcontrôleur fonctionne, il lui faut une alimentation ! Cette alimentation se fait en générale par du +5V. D'autres ont besoin d'une tension plus faible, du +3,3V.

En plus d'une alimentation, il a besoin d'un signal d'horloge. C'est en fait une succession de 0 et de 1 ou plutôt une succession de tension 0V et 5V. Elle permet en outre de cadencer le fonctionnement du microcontrôleur à un rythme régulier. Grâce à elle, il peut introduire la notion de temps en programmation.

Description de la carte

Version USB



Référence analogique (AREF)

LED jaune de test

Masse des connecteurs numériques (GND)

Port Série In/out
RX TX (supprime une entrée-sortie numérique si utilisé)

Trou de fixation par vis
(fixer sur un support non conducteur)

13 entrées/sorties numériques
numérotées de 1 à 13

Port USB:
pour le transport des données et pour l'alimentation électrique

LED verte témoin d'alimentation

Bouton de **réinitialisation** (reset)

Connecteurs pour le téléchargement du système d'exploitation du microcontrôleur

3 broches, avec **un cavalier** de sélection d'alimentation:

-cavalier sur les deux broches du haut: alimentation par le port USB, comme sur cette photo

-cavalier sur les deux broches du bas si alimentation par connecteur ci-dessous

Microcontrôleur

Trou de fixation par vis
(fixer sur un support non conducteur)

Connecteur d'alimentation de l'Arduino (si besoin)
9v à 12V
Connecteur 2,1 mm
avec le + au centre
courant continu
(DC)

2 LEDs jaunes ou vertes, actives lors d'un téléchargement de programme

6 entrées analogiques
numérotées de 0 à 5
fournissant une variation
de 0 à 1023

3 connecteurs numériques
pouvant également être utilisés en générateurs d'impulsions
(par ex: sortie analogique avec montage spécifique
PWM numérotés de 0 à 2)

Masse (2 connecteurs)
(GND)

Source 5V faible puissance pour alimenter des montages divers comme des capteurs 0-5 V par exemple

Source 9V faible puissance pour alimenter des montages divers

LOGICIEL POUR ARDUINO

Ce logiciel gratuit peut être télécharger directement à partir du site web d'arduino. On le trouve pour plusieurs systèmes d'exploitation.

En ce qui nous concerne nous téléchargeons la version adaptée à windows.

https://www.arduino.cc/en/Main/Software#

Remarque: Après avoir installé ce logiciel, il ne faut pas oublier également d'installer le driver permettant le transfert du programme du PC vers la carte arduino

LOGICIEL POUR ARDUINO

Ce logiciel gratuit peut être télécharger directement à partir du site web d'arduino. On le trouve pour plusieurs systèmes d'exploitation.

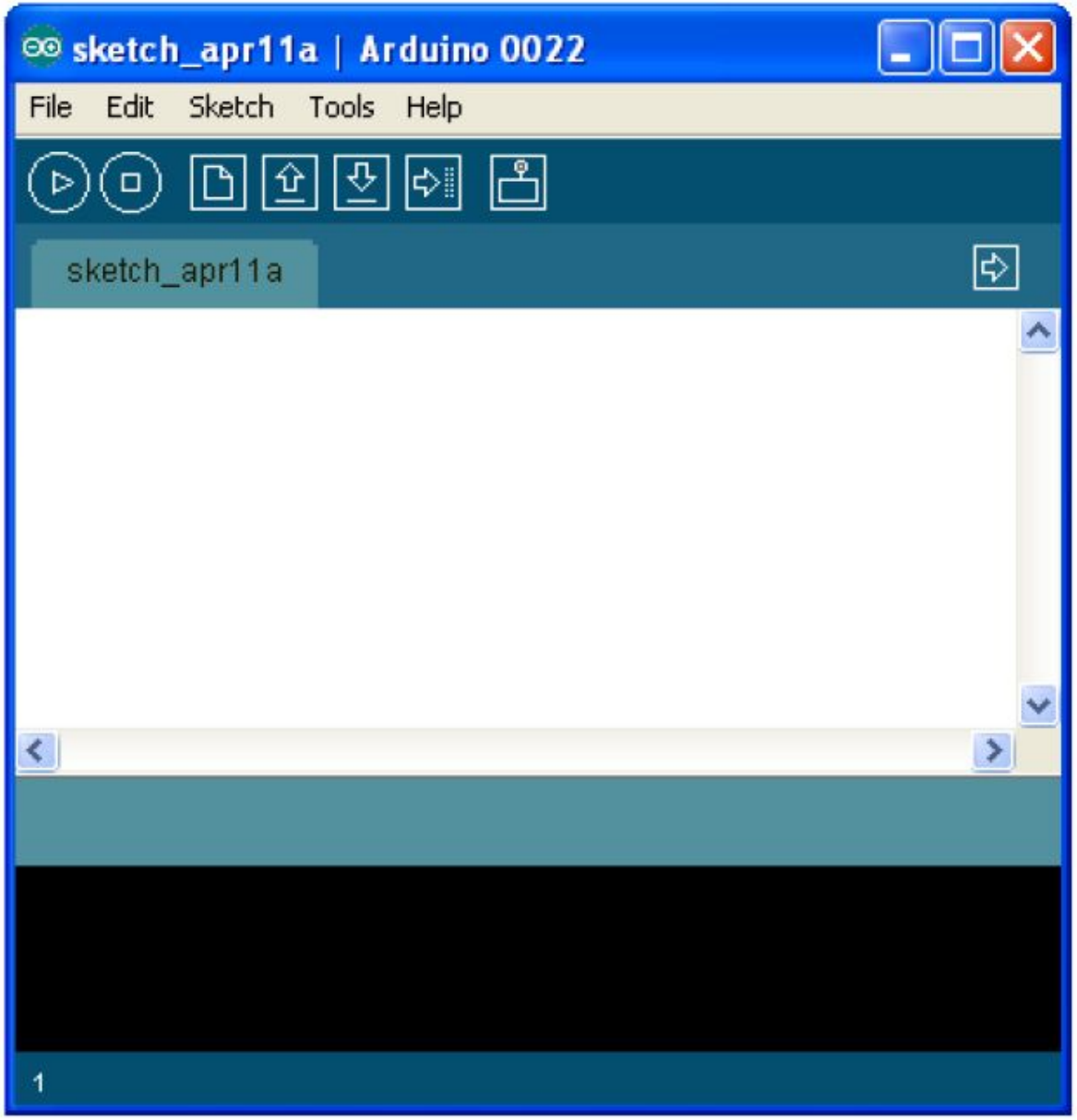
En ce qui nous concerne nous téléchargeons la version adaptée à windows.

https://www.arduino.cc/en/Main/Software#

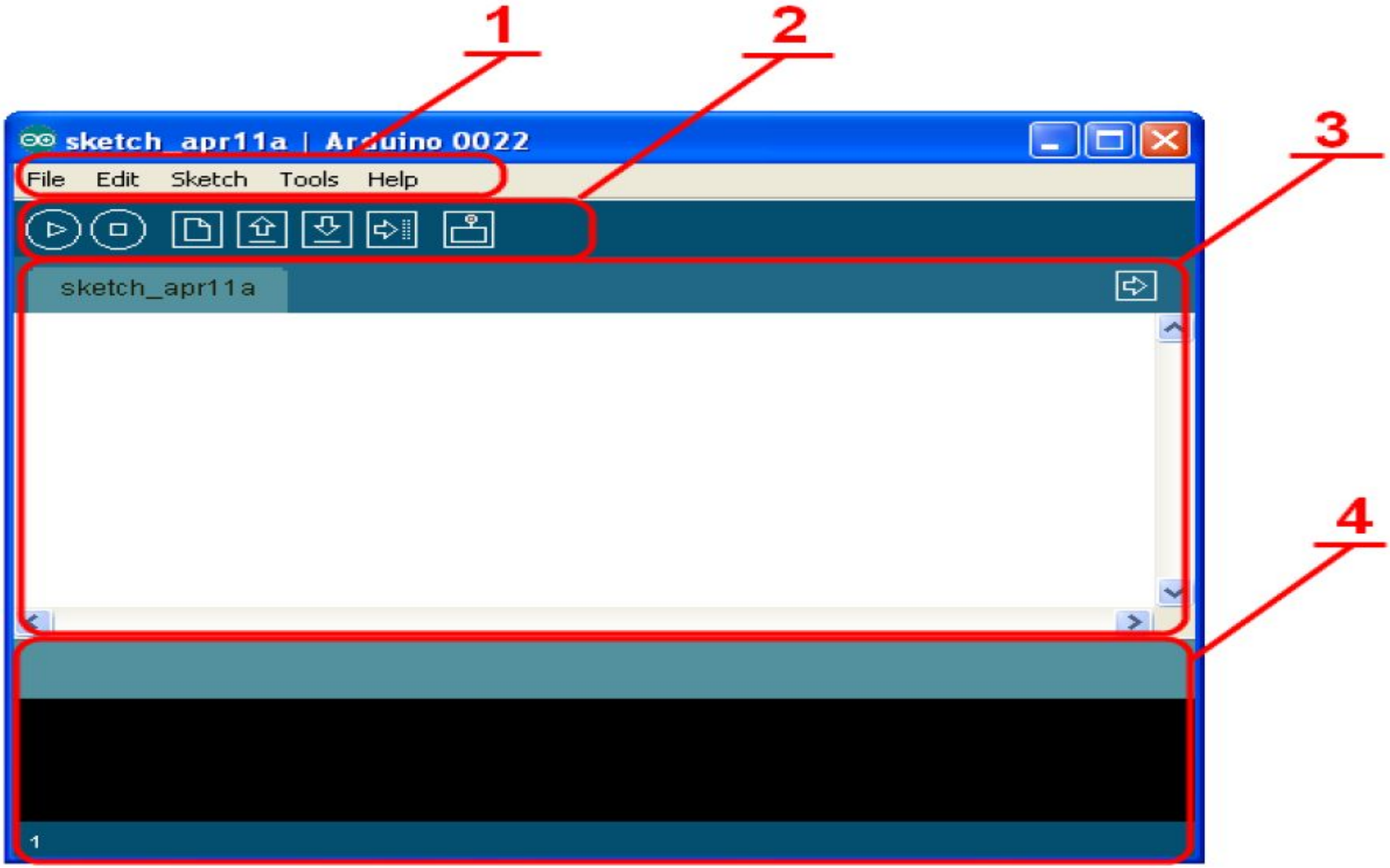
Remarque: Après avoir installé ce logiciel, il ne faut pas oublier également d'installer le driver permettant le transfert du programme du PC vers la carte arduino

- Le logiciel

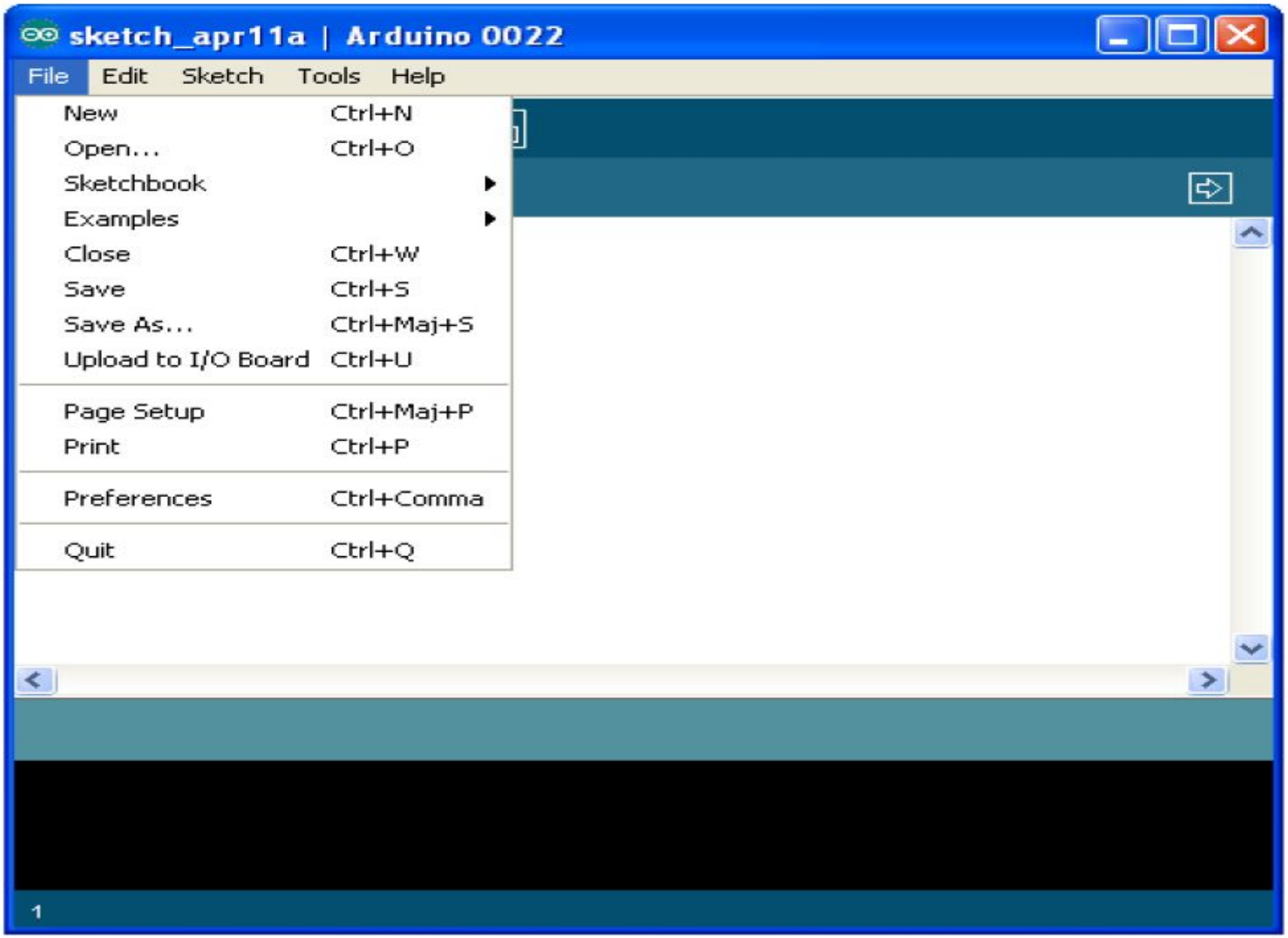
  Le logiciel permet de programmer la carte Arduino. Il offre une multitude de fonctionnalités.



L'interface du logiciel Arduino se présente de la façon suivante :

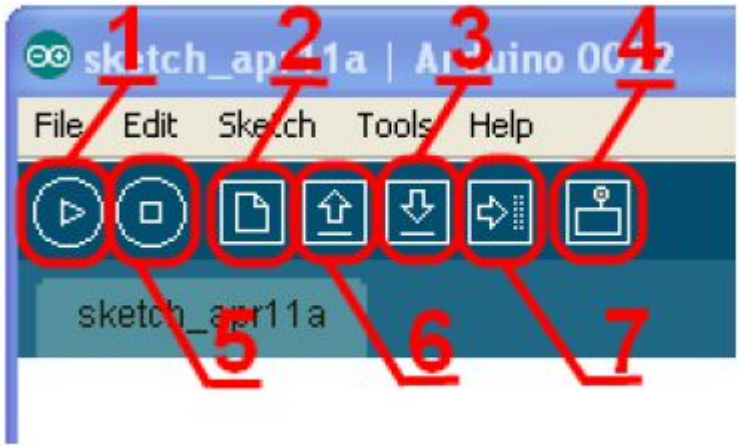


1. options de configuration du logiciel
2. boutons pout la programmation des cartes
3. programme à créer
4. débogueur (affichage des erreurs de programmation)



- New (nouveau) : va permettre de créer un nouveau programme. Quand on appuie sur ce bouton, une nouvelle fenêtre, identique à celle-ci, s'affiche à l'écran.
- Open... (ouvrir) : avec cette commande, on peut ouvrir un programme existant.
- Save / Save as... (enregistrer / enregistrer sous...) : enregistre le document en cours / demande où enregistrer le document en cours.
- Examples (exemples) : ceci est important, toute une liste se déroule pour afficher les noms d'exemples de programmes existant.

Les boutons



1. permet de vérifier le programme, il actionne un module qui cherche les erreurs dans le programme
2. Créer un nouveau fichier
3. Sauvegarder le programme en cours
4. Liaison série
5. Stoppe la vérification
6. Charger un programme existant
7. Compiler et envoyer le programme vers la carte

Le langage Arduino

Le projet Arduino était destiné à l'origine principalement à la programmation multimédia interactive en vue de spectacle ou d'animations artistiques. C'est une partie de l'explication de la descendance de son interface de programmation de Processing.

Processingest une librairie java et un environnement de développement libre. Le logiciel fonctionne sur Macintosh, Windows, Linux, BSD et Android.

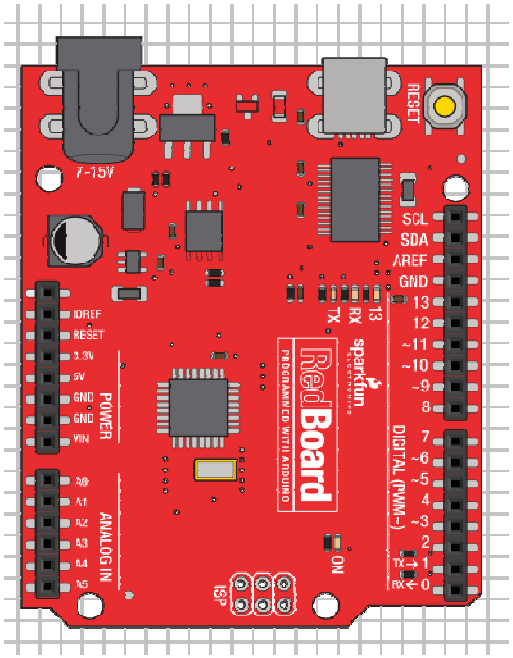
Cependant, le projet Arduino a développé des fonctions spécifiques à l'utilisation de la carte qui ont été listées ci-dessous. Vous obtiendrez la description de chacune d'elles dans le manuel de référence.

Le langage Arduino

Structure	Constants	Functions
• setup() • loop()	• HIGH, LOW • INPUT, OUTPUT, INPUT_PULLUP • LED_BUILTIN	E/S numérique • pinMode() • digitalWrite() • digitalRead() E/S analogique • analogReference() • analogRead() • analogWrite() - PWM E/S avancée • tone() • noTone() • shiftOut() • shiftIn() • pulseIn() Temps • millis() • micros()

		• delay() • delayMicroseconds() Bits et octets • lowByte() • highByte() • bitRead() • bitWrite() • bitSet() • bitClear() • bit() Interruptions externes • attachInterrupt() • detachInterrupt() Interruptions • interrupts() • noInterrupts() Communication • Serial • Stream

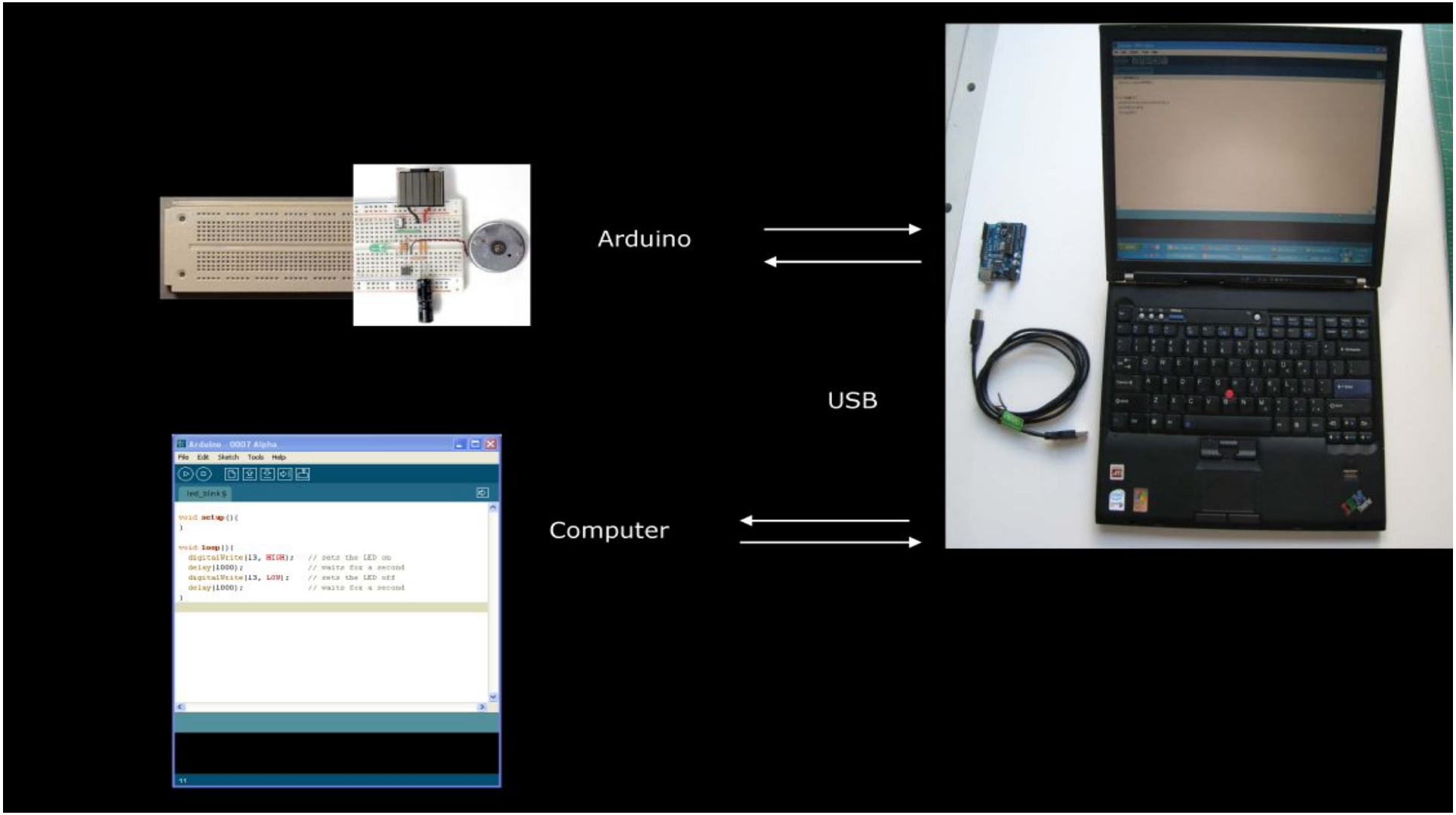
Test de la carte



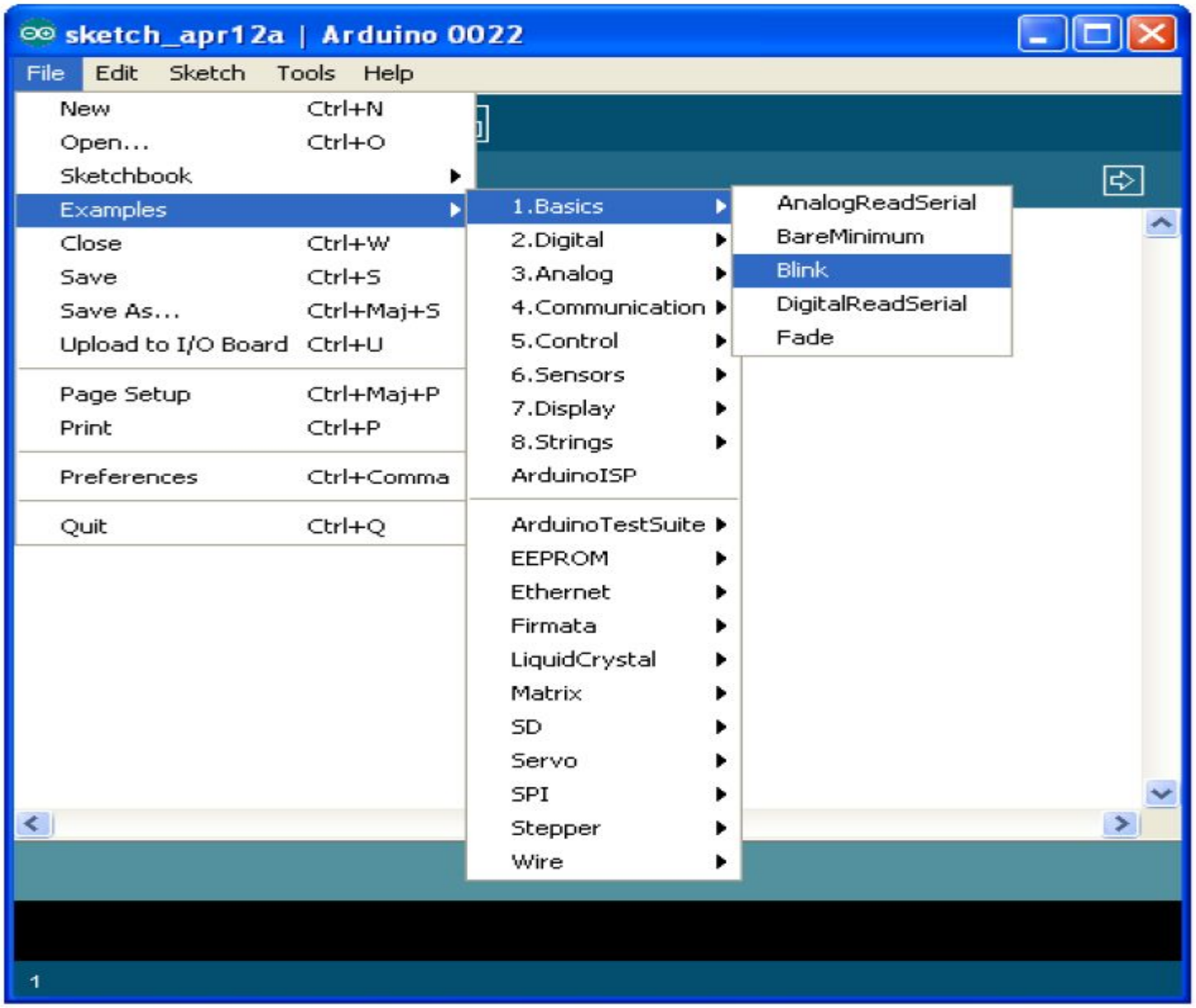
Test de la carte

On teste le matériel en chargeant un programme d'exemple que l'on utilisera pour tester la carte.

On choisira un exemple qui consiste à faire clignoter une LED. Son nom est Blink et il se trouve dans la catégorie Basics :

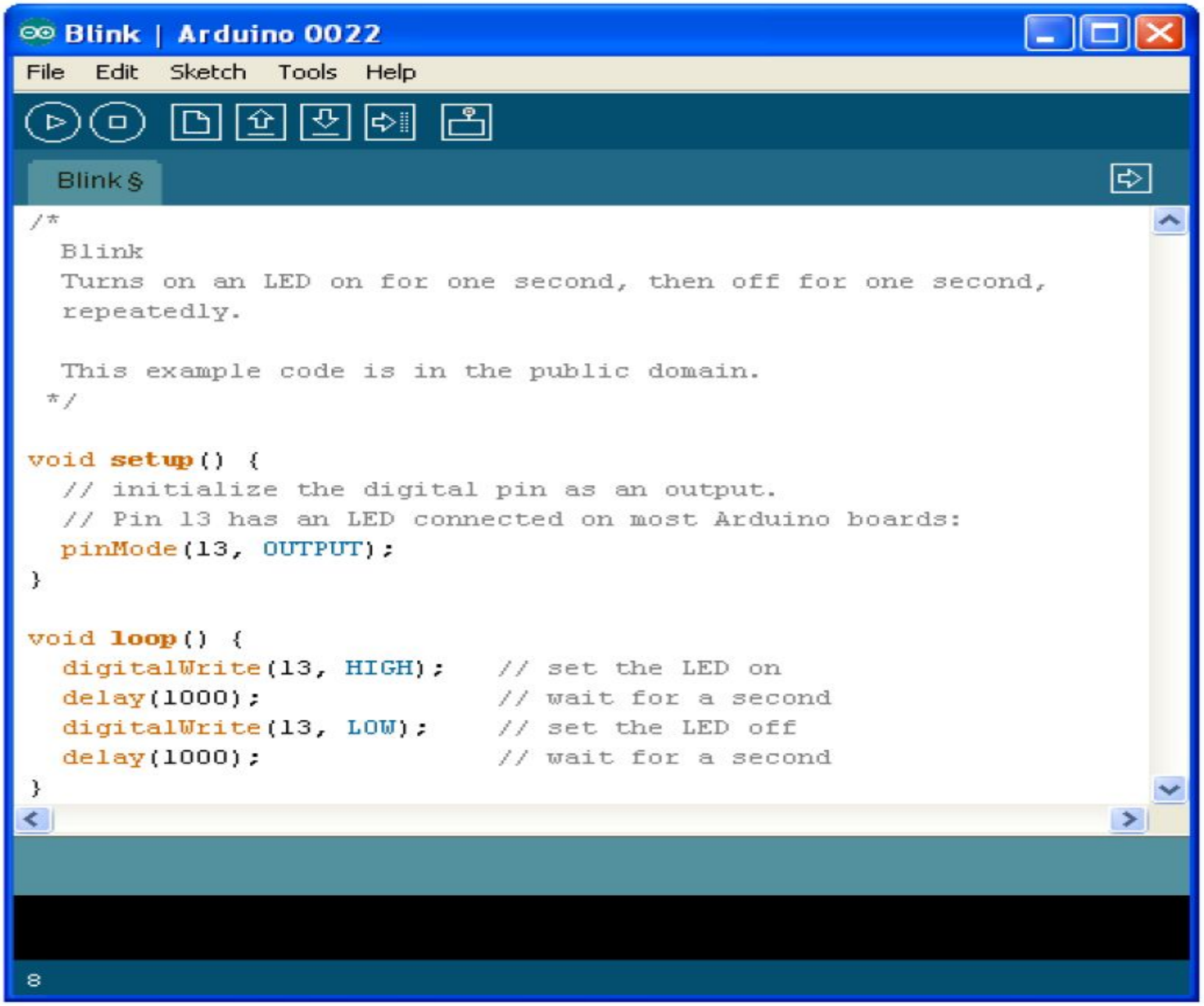


Test de la carte



Ouvrir le programme Blink

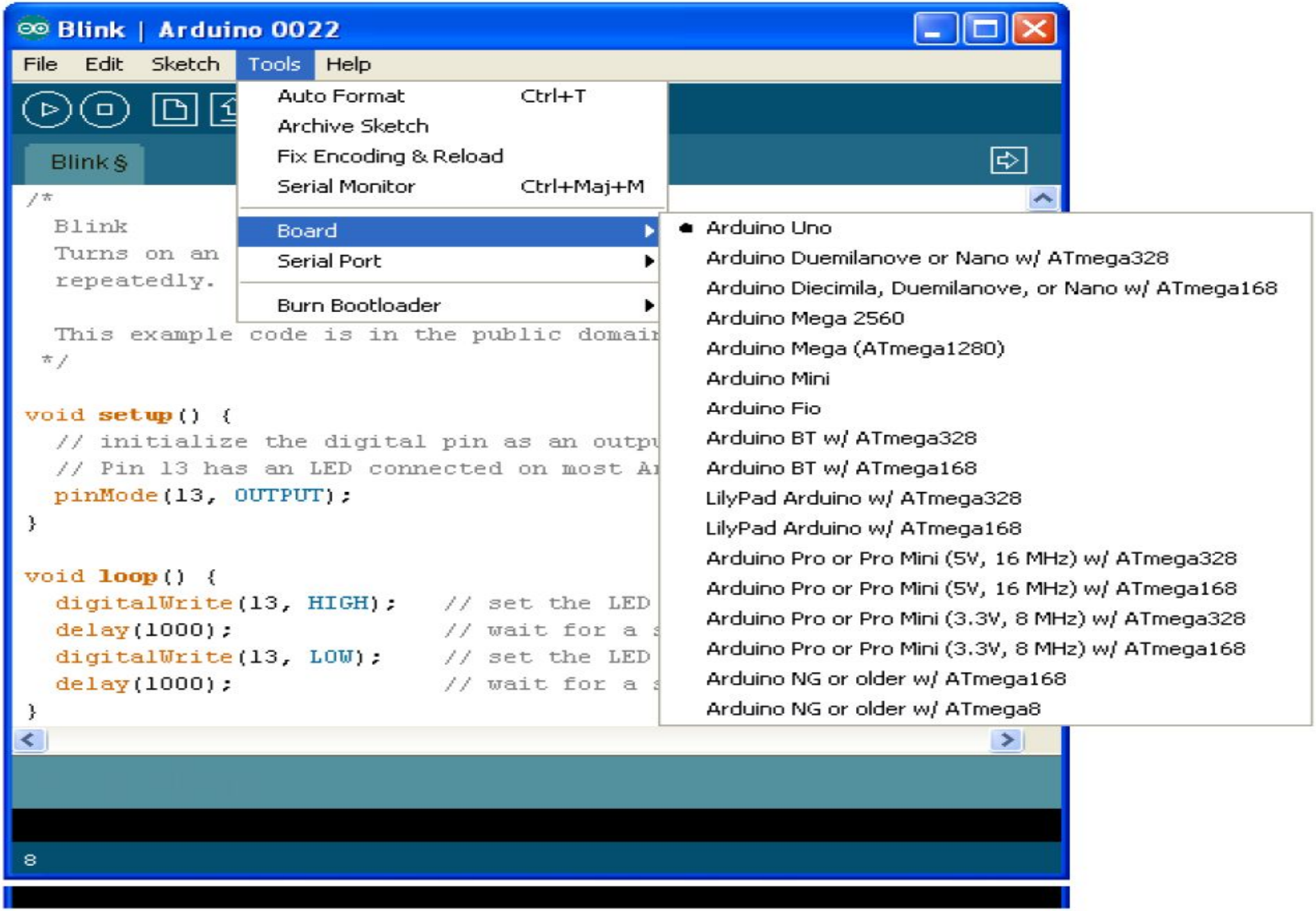
Test de la carte



Test de la carte

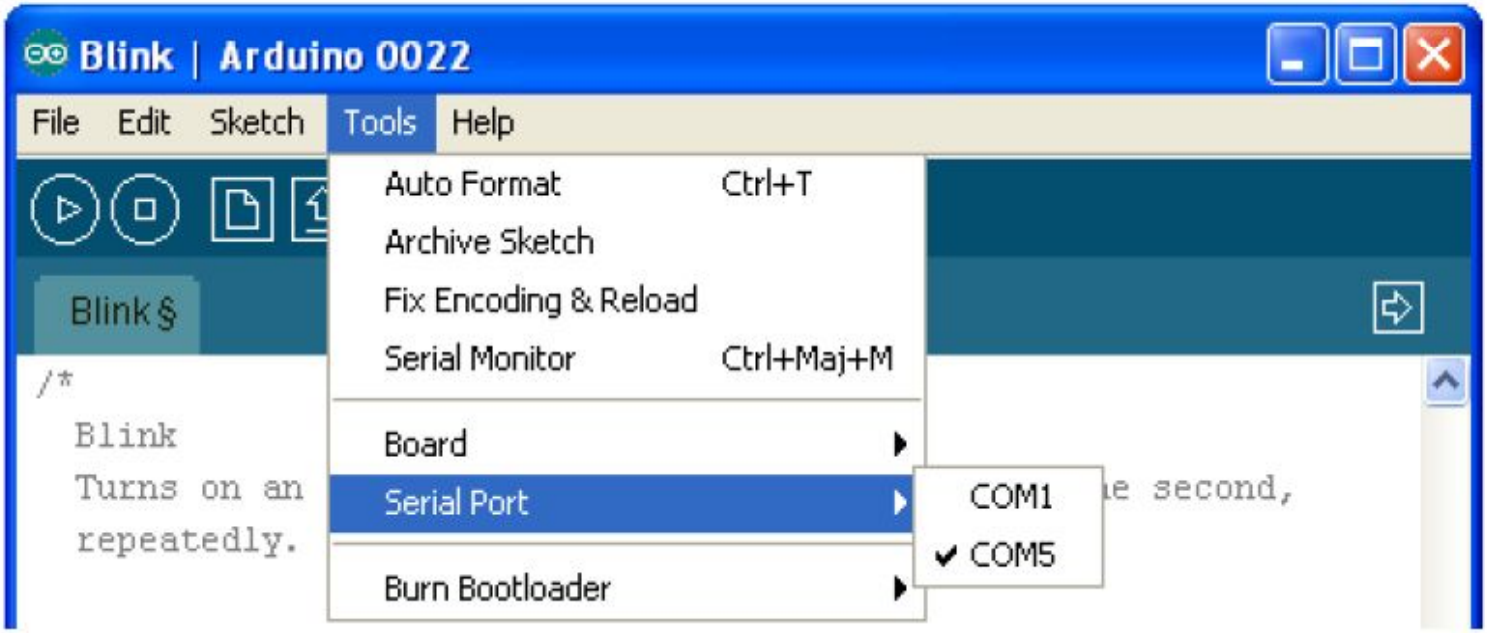
Avant d'envoyer le programme Blink vers la carte, il faut dire au logiciel quel est le nom de la carte et sur quel port elle est branchée. Pour cela, allez dans le menu "Tools" ("outils" en français) puis dans "Board" ("carte" en français). Vérifiez que c'est bien le nom "Arduin Uno" qui est coché. Si ce n'est pas le cas, cochez-le.

Test de la carte



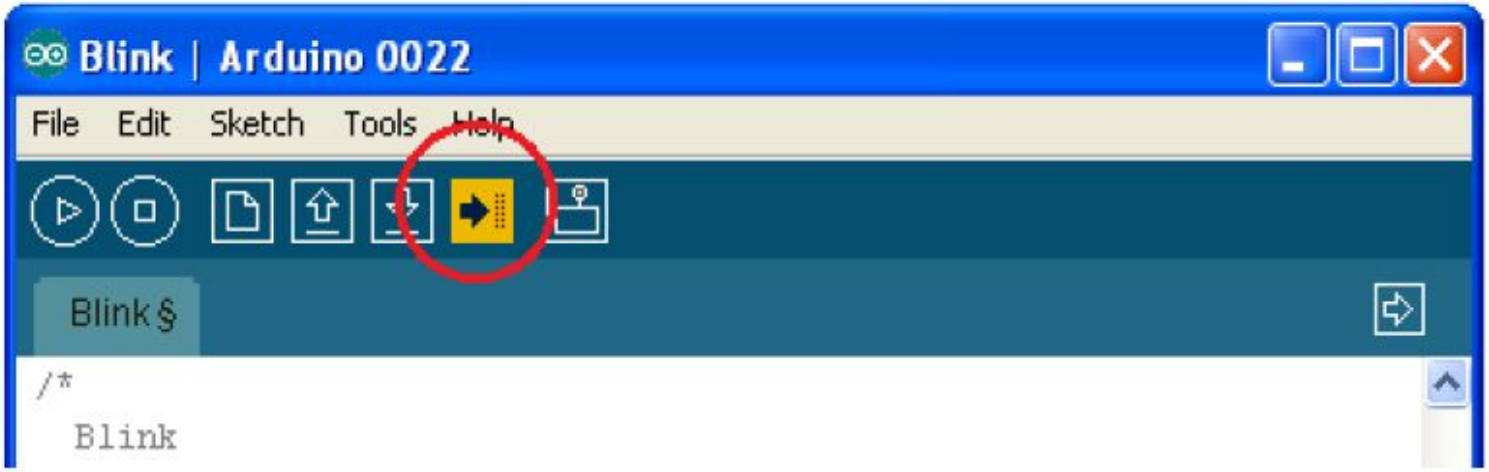
Test de la carte

Allez ensuite dans le menu Tools, puis Serial port. Choisissez le port COMX, X étant le numéro du port qui est affiché. Ne choisissez pas COM1 car il n'est quasiment jamais connecté à la carte. Dans l'exemple, il s'agit de COM5 :

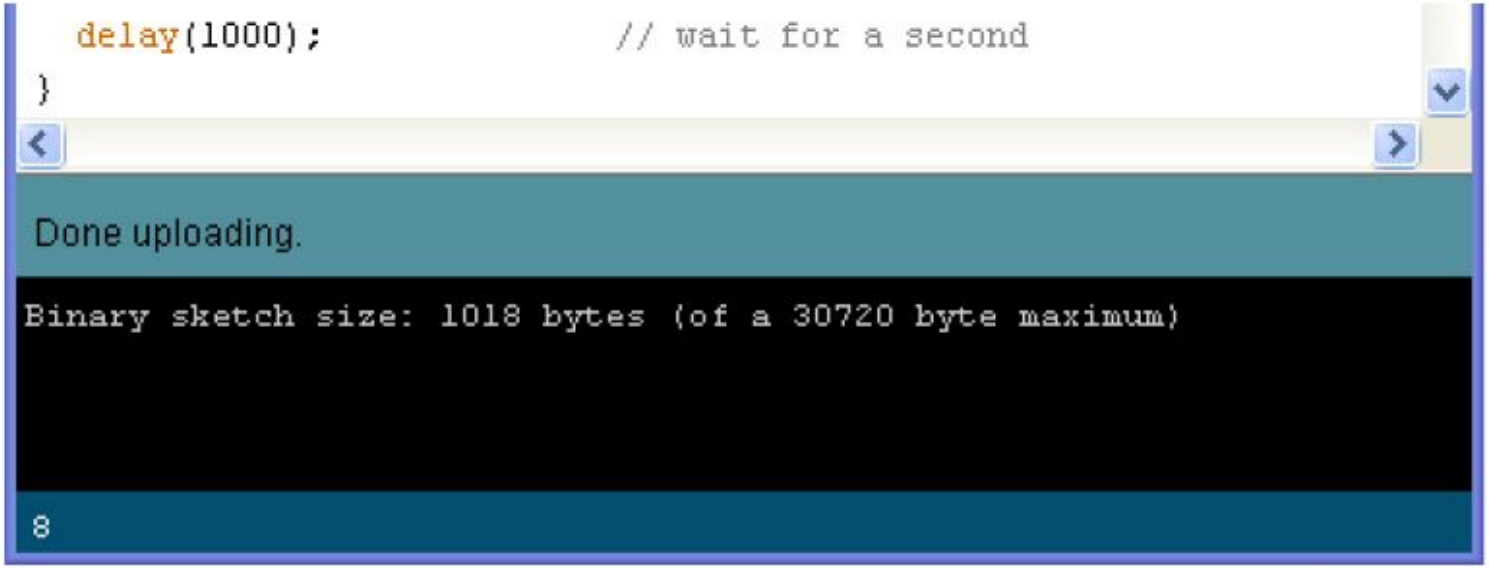


Test de la carte

Maintenant, il va falloir envoyer le programme dans la carte. Pour ce faire, il suffit de cliquer sur le bouton Upload (ou "Télécharger" en Français), en jaune-orangé sur la photo :



En bas dans l'image, vous voyez le texte : "Uploading to I/O Board...", cela signifie que le logiciel est en train d'envoyer le programme dans la carte. Une fois qu'il a fini, il affiche un autre message :



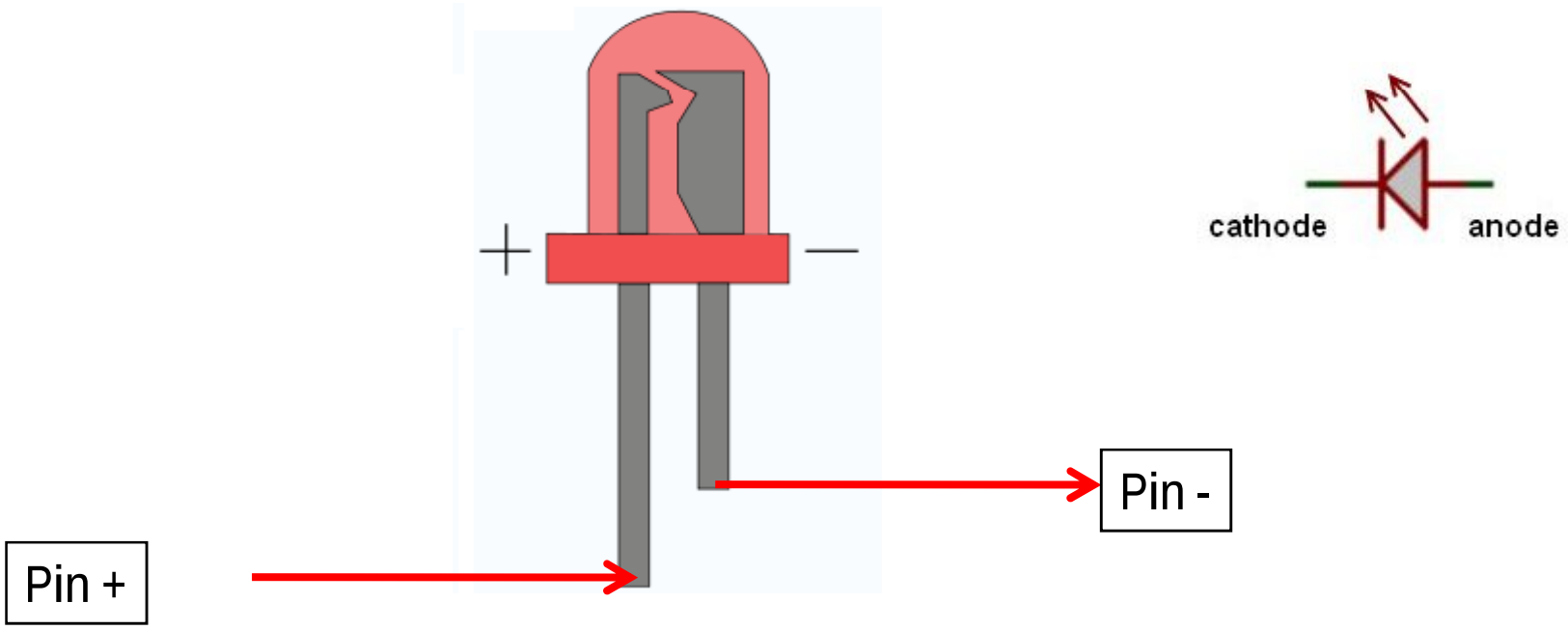
Test de la carte

Le message afficher : "Done uploading" signale que le programme à bien été chargé dans la carte.
Si votre matériel fonctionne, vous devriez avoir une LED sur la carte qui clignote :

Gestion des entrées / sorties

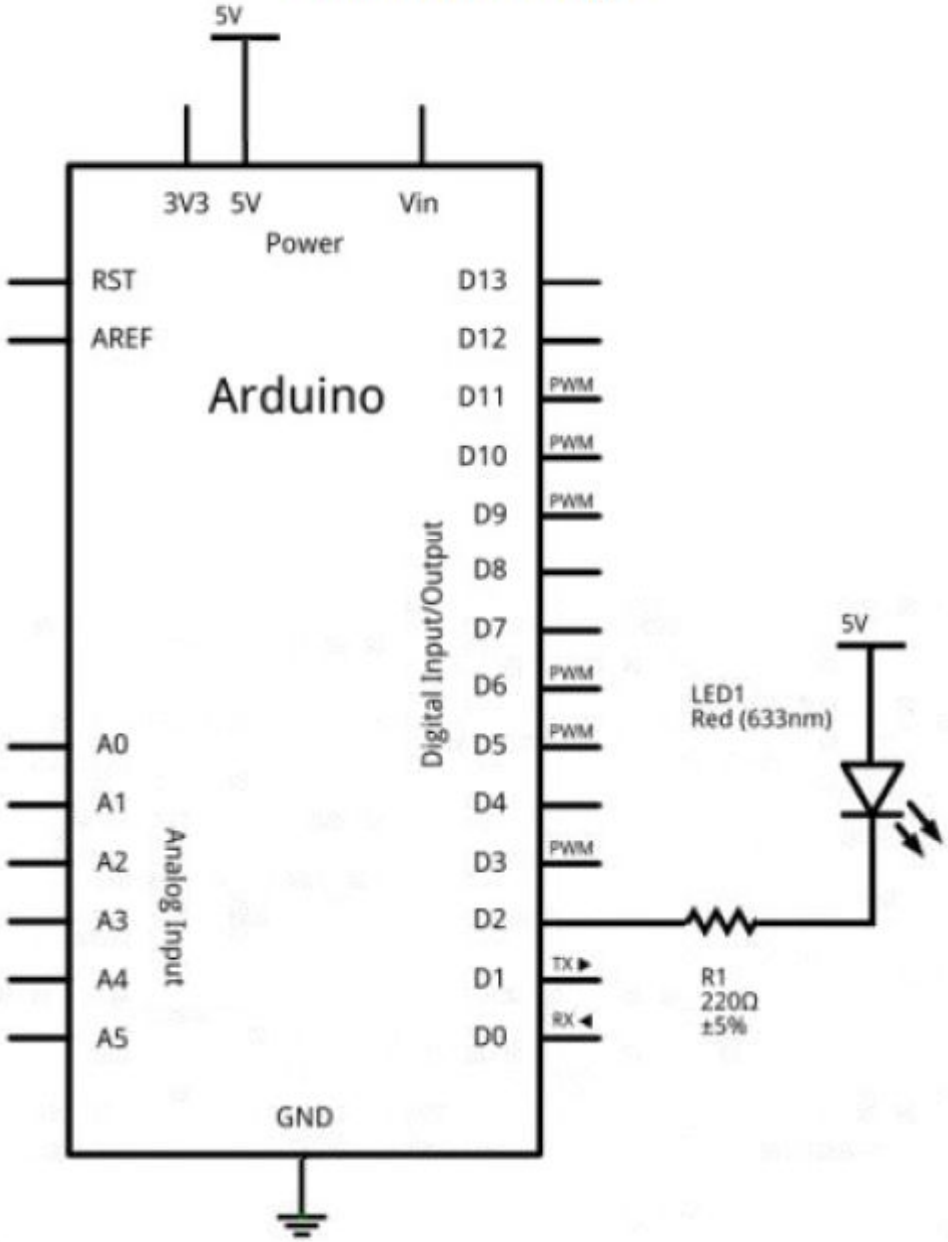
L'objectif de ce premier programme va consister à allumer une LED.

Avec le brochage de la carte Arduino, vous devrez connecter la plus grande patte au +5V (broche 5V). La plus petite patte étant reliée à la résistance, elle-même reliée à la broche numéro 2 de la carte. On pourrait faire le contraire, brancher la LED vers la masse et l'allumer en fournissant le 5V depuis la broche de signal. Cependant, les composants comme les microcontrôleurs n'aiment pas trop délivrer du courant, ils préfèrent l'absorber. Pour cela, on préférera donc alimenter la LED en la placant au +5V et en mettant la broche de Arduino à la masse pour faire passer le courant.



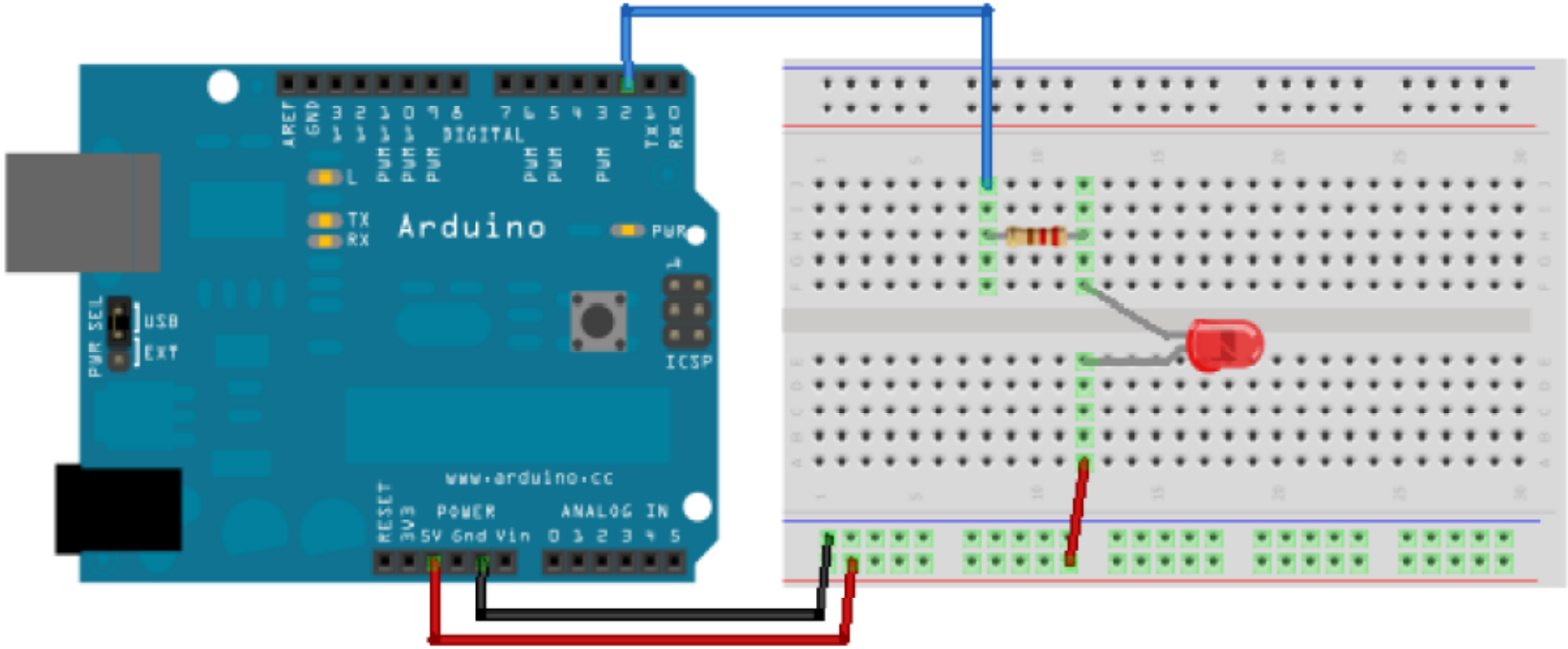
Gestion des entrées / sorties

Schéma électrique



Gestion des entrées / sorties

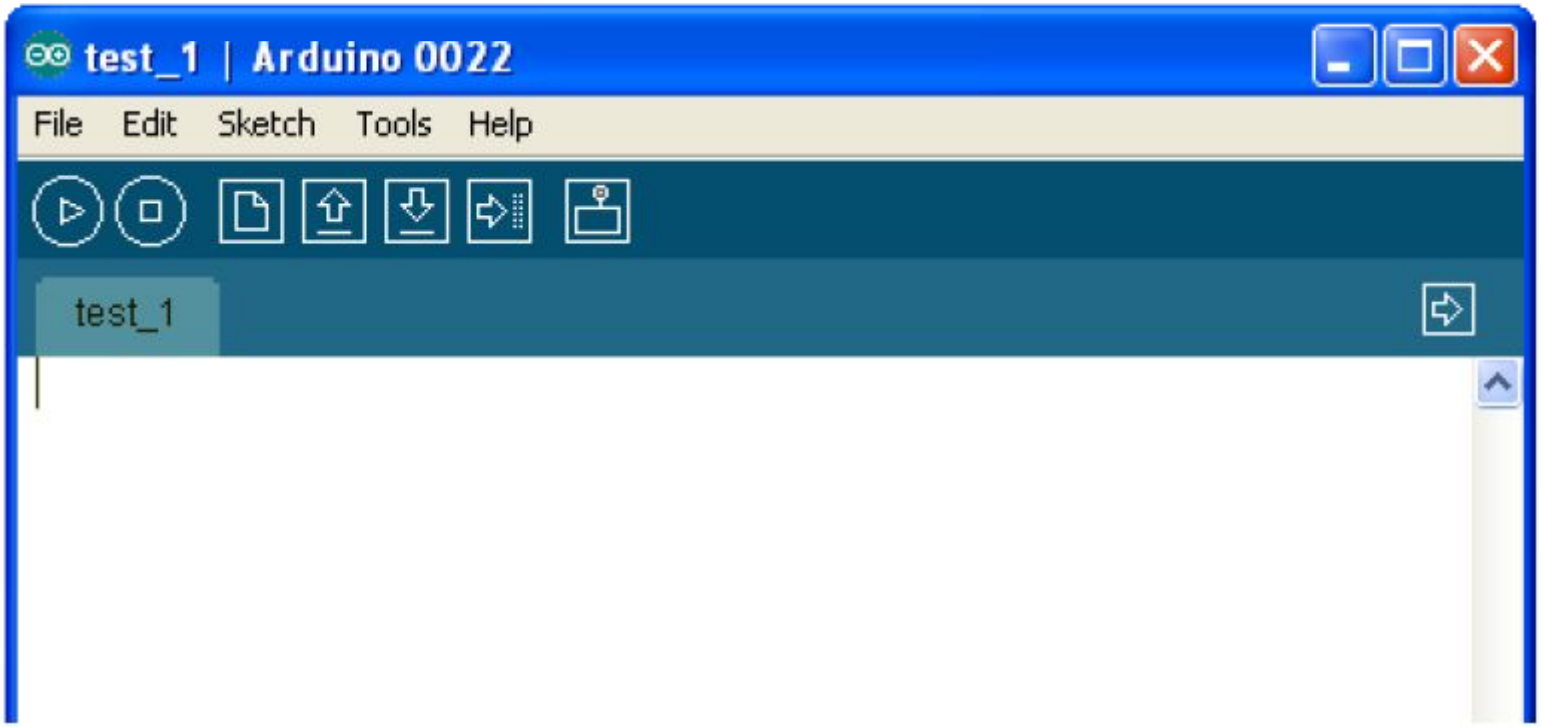
Réalisation



Gestion des entrées / sorties

Pour commencer le programme, il faut un code minimal. Ce code va permettre d'initialiser la carte :

```
void setup()    //fonction d'initialisation de la carte
{
        //contenu de l'initialisation
}
void loop()     //fonction principale, elle se répète (s'exécute) à l'infini
{
        //contenu du programme
}
```



Gestion des entrées / sorties

Premièrement, définissons la broche utilisée du micro-contrôleur :

```
const int led_rouge = 2;   //définition de la broche 2 de la carte en tant que variable
```

Il faut maintenant dire si cette broche est une entrée ou une sortie. Cette ligne de code doit se trouver dans la fonction setup(). La fonction à utiliser est pinMode(). Pour utiliser cette fonction, il faut lui envoyer deux paramètres :

- Le nom de la variable que l'on a défini à la broche
- Le type de broche que cela va être (entrée ou sortie)

```
void setup()
{
        pinMode(led_rouge, OUTPUT);       // initialisation de la broche 2 comme étant une sortie
}
```

La deuxième étape consiste à créer le contenu du programme. Celui qui va aller remplacer le commentaire dans la fonction loop() pour allumer la LED.

Gestion des entrées / sorties

On va utiliser la fonction digitalWrite() qui va écrire une valeur HAUTE (+5V) ou BASSE (0V) sur une sortie numérique. La LED étant connecté au pôle positif de l'alimentation, il faut qu'elle soit reliée au 0V. Par conséquent, on doit mettre un état bas sur la broche du microcontrôleur. Ainsi, la différence de potentiel aux bornes de la LED permettra à celle-ci de s'allumer

La fonction digitalWrite() requiert deux paramètres : le nom de la broche que l'on veut mettre à un état logique et la valeur de cet état logique.

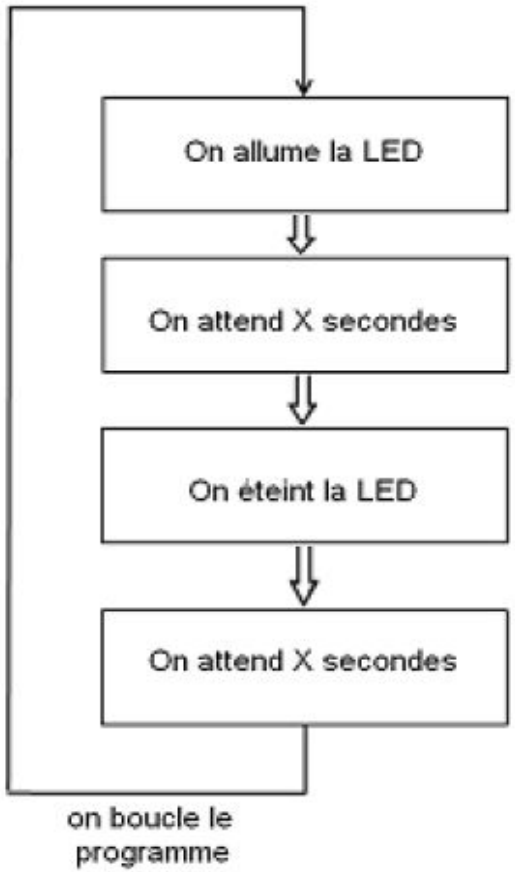
Voici le code entier :

```
const int led_rouge = 2;        //définition de la broche 2 de la carte en tant que variable

void setup()    //fonction d'initialisation de la carte
{
        pinMode(led_rouge, OUTPUT);      //initialisation de la broche 2 comme étant une sortie
}

void loop()      //fonction principale, elle se répète (s'exécute) à l'infini
{
        digitalWrite(led_rouge, LOW);        // écriture en sortie (broche 2) d'un état BAS
}
```

Temporisation

Fonction delay()

La fonction delay() va servir à mettre en pause le programme pendant un temps prédéterminé. Elle admet un paramètre qui est le temps pendant lequel on veut mettre en pause le programme. Ce temps doit être donné en millisecondes.



Pour faire clignoter la LED on fait intervenir la fonction delay(), qui va mettre le programme en pause pendant un certain temps.

Ensuite, on éteint la LED.

On met en pause le programme.

Puis on revient au début du programme. On recommence et ainsi de suite.

Temporisation

Fonction delay()

Ce qui donne le code suivant :

```
const int led_rouge = 2;        //définition de la broche 2 de la carte en tant que variable
void setup()     //fonction d'initialisation de la carte
{
        pinMode(led_rouge, OUTPUT);      //initialisation de la broche 2 comme étant une sortie
}
void loop()      //fonction principale, elle se répète (s'exécute) à l'infini
{
        digitalWrite(led_rouge, LOW);        // allume la LED
        delay(1000);                         // fait une pause de 1 seconde

        digitalWrite(led_rouge, HIGH);       // éteint la LED
        delay(1000);                         // fait une pause de 1 seconde
}
```

PROGRAMMATION D'UN ARDUINO

La syntaxe du langage

La syntaxe d'un langage de programmation est l'ensemble des règles d'écritures liées à ce langage.

Le code minimal

Avec Arduino, nous devons utiliser un code minimal lorsque l'on crée un programme. Ce code permet de diviser le programme que nous allons créer en deux grosses parties.

```
//fonction d'initialisation de la carte
void setup()
{
   //contenu de l'initialisation
}

//fonction principale, elle se répète (s'exécute) à l'infini
void loop()
{
   //contenu de votre programme
}
```

PROGRAMMATION D'UN ARDUINO

La fonction Setup()

Dans ce code se trouvent deux fonctions. Les fonctions sont en fait des portions de code.

```
//fonction d'initialisation de la carte

void setup()
{
   //contenu de l'initialisation
   //on écrit le code à l'intérieur
}
```

Cette fonction **setup()** est appelée une seule fois lorsque le programme commence. C'est pourquoi c'est dans cette fonction que l'on va écrire le code qui n'a besoin d'être exécuté une seule fois. On appelle cette fonction : « **fonction d'initialisation**« .

PROGRAMMATION D'UN ARDUINO

La fonction loop()

C'est donc dans cette fonction **loop()** où l'on va écrire le contenu du programme. Il faut savoir que cette fonction est appelée en permanence, c'est-à-dire qu'elle est exécutée une fois, puis lorsque son exécution est terminée, on la ré-exécute et encore et encore. On parle de **boucle infinie**.

```
//fonction principale, elle se répète (s'exécute) à l'infini
void loop()
{
   //contenu de votre programme
}
```

A titre informatif, on n'est pas obligé d'écrire quelque chose dans ces deux fonctions. En revanche, il est **obligatoire** de les écrire, même si elles ne contiennent aucun code !

PROGRAMMATION D'UN ARDUINO

Soit un exemple d'un programme avec les deux fonctions setup() et loop()

Tout ce qui commence par // est un commentaire qui ne s'exécute pas

```
void setup() {
  // initialize la pin 13 (digital pin) comme SORTIE.
  pinMode(13, OUTPUT);
}

// La fonction loop() va être exécutée en continue
void loop() {
  digitalWrite(13, HIGH);   // La sortie 13 se met en niveau HAUT soit 5V
  delay(1000);                  // On attend 1s ou bien 1000ms
  digitalWrite(13, LOW);    // La sortie 13 se met en niveau BAS soit 0V
  delay(1000);                  // wait for a second
}
```

PROGRAMMATION D'UN ARDUINO

Soit un autre exemple d'un programme utilisant deux sorties TOR la pin 13 et la pin 10
Les deux LEDS connectées sur les pins 13 et 10 de l'arduino vont clignoter l'une en inverse de l'autre

```
void setup() {
  // initialize la pin 13  et la pin 10(digital pin) comme SORTIES.
  pinMode(13, OUTPUT);
  pinMode(10, OUTPUT);
}

// La fonction loop() va être exécutée en continue
void loop() {
  digitalWrite(13, HIGH);   // La sortie 13 se met en niveau HAUT soit 5V
 digitalWrite(10, LOW);   // La sortie 10 se met en niveau BAS soit 0V
  delay(1000);                  // On attend 1s ou bien 1000ms
  digitalWrite(13, LOW);    // La sortie 13 se met en niveau BAS soit 0V
 digitalWrite(10, HIGH);    // La sortie 10 se met en niveau HAUT soit 5V
  delay(1000);                  // wait for a second
}
```

PROGRAMMATION D'UN ARDUINO

Soit un troisième exemple d'un programme utilisant deux sorties TOR la pin 13 et la pin 10
Les deux LEDS connectées sur les pins 13 et 10 de l'arduino vont clignoter avec des fréquences différentes

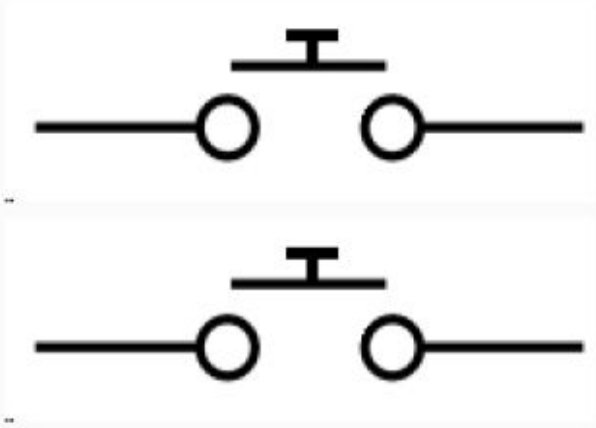
```
void setup() {
  // initialize la pin 13  et la pin 10(digital pin) comme SORTIES.
  pinMode(13, OUTPUT);
  pinMode(10, OUTPUT);
}

// La fonction loop() va être exécutée en continue
void loop() {
  digitalWrite(13, HIGH);   // La sortie 13 se met en niveau HAUT soit 5V
 digitalWrite(10, HIGH);   // La sortie 10 se met en niveau HAUT soit 5V
 delay(500);                // On attend 250ms
 digitalWrite(10, LOW);    // La sortie 10 se met en niveau BAS soit 0V
  delay(500);               // Attendre 250 ms
digitalWrite(13, LOW);    // La sortie 13 se met en niveau BAS soit 0V
digitalWrite(10, HIGH);   // La sortie 10 se met en niveau HAUT soit 5V
  delay(250);               // Attendre 250 ms
digitalWrite(10, HIGH);   // La sortie 10 se met en niveau HAUT soit 5V
  delay(500);               // Attendre 250 ms
}
```

Le bouton poussoir

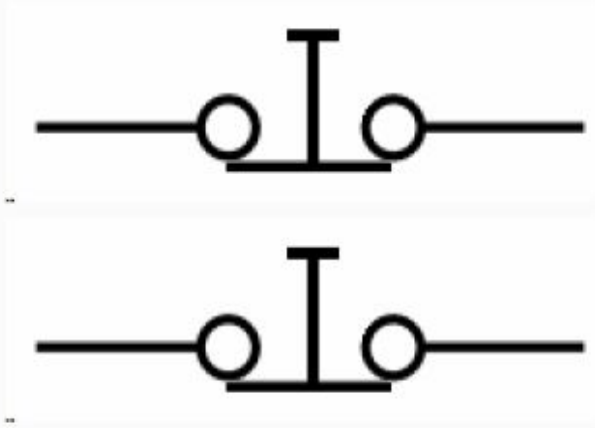
Les boutons poussoirs (BP) normalement ouvert (NO) ont deux positions :

- Relâché : le courant ne passe pas, le circuit est "ouvert".
- Appuyé : le courant passe, le circuit est fermé.

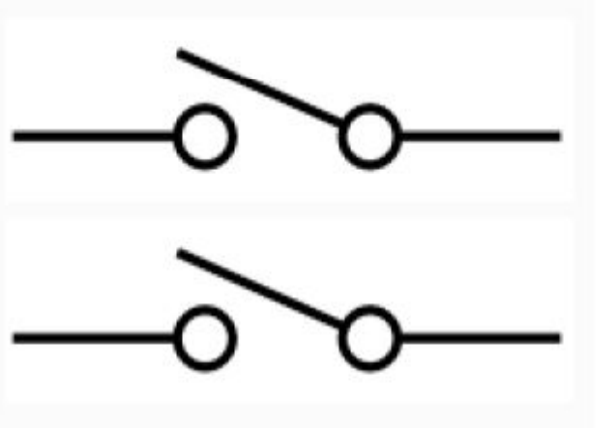
Le bouton poussoir normalement fermé (NF) est l'opposé du type précédent, c'est-à-dire que lorsque le bouton est relâché, il laisse passer le courant. Et inversement :



Bouton Poussoir NO

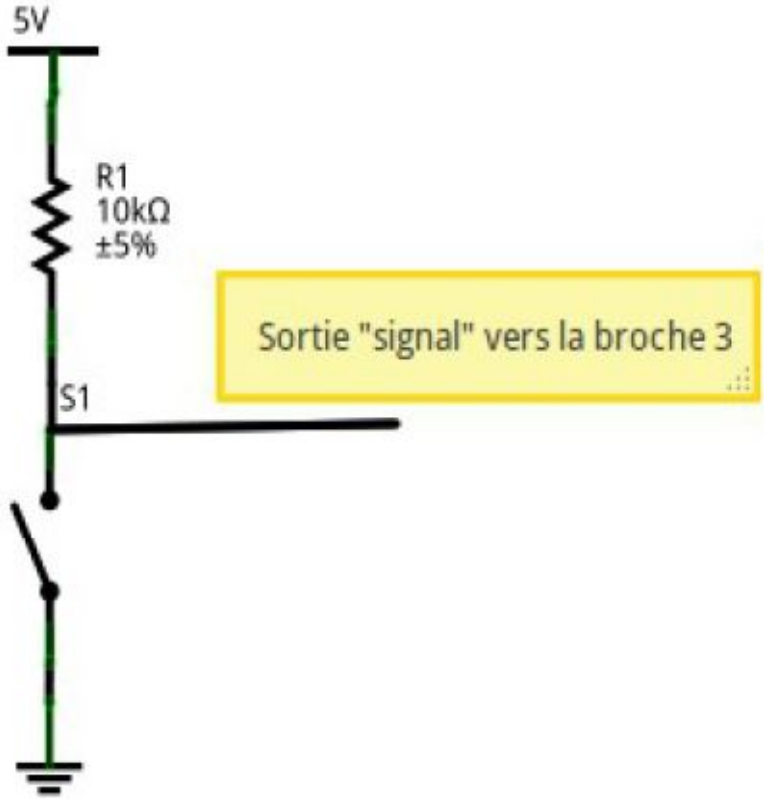


Bouton Poussoir NF



Interrupteur

Le bouton poussoir

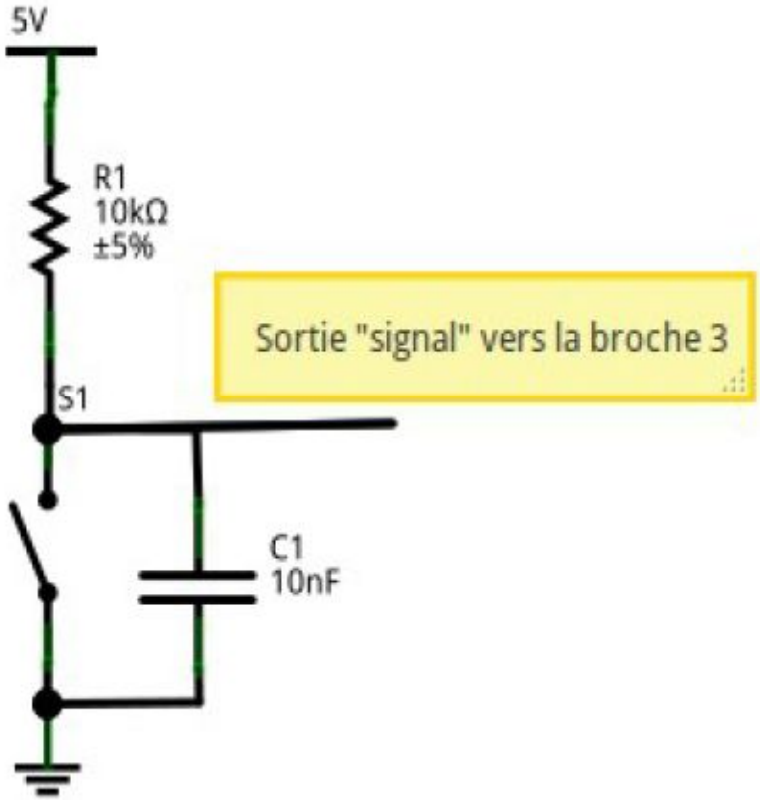


En électronique, on a toujours des perturbations (générées par des lampes à proximité, un téléphone portable, ...). On appelle ça des contraintes de CEM. Pour contrer ces effets nuisibles, ont place en série avec le bouton une résistance de pull-up. Cette résistance sert à "tirer" ("to pull" in english) le potentiel vers le haut (up) afin d'avoir un signal clair sur la broche étudiée.

Sur le schéma suivant, on voit ainsi qu'en temps normal le "signal" à un potentiel de 5V. Ensuite, lorsque l'utilisateur appuiera sur le bouton une connexion sera faite avec la masse. On lira alors une valeur de 0V pour le signal. Voici donc un deuxième intérêt de la résistance de pull-up, éviter le court-circuit qui serait généré à l'appui !

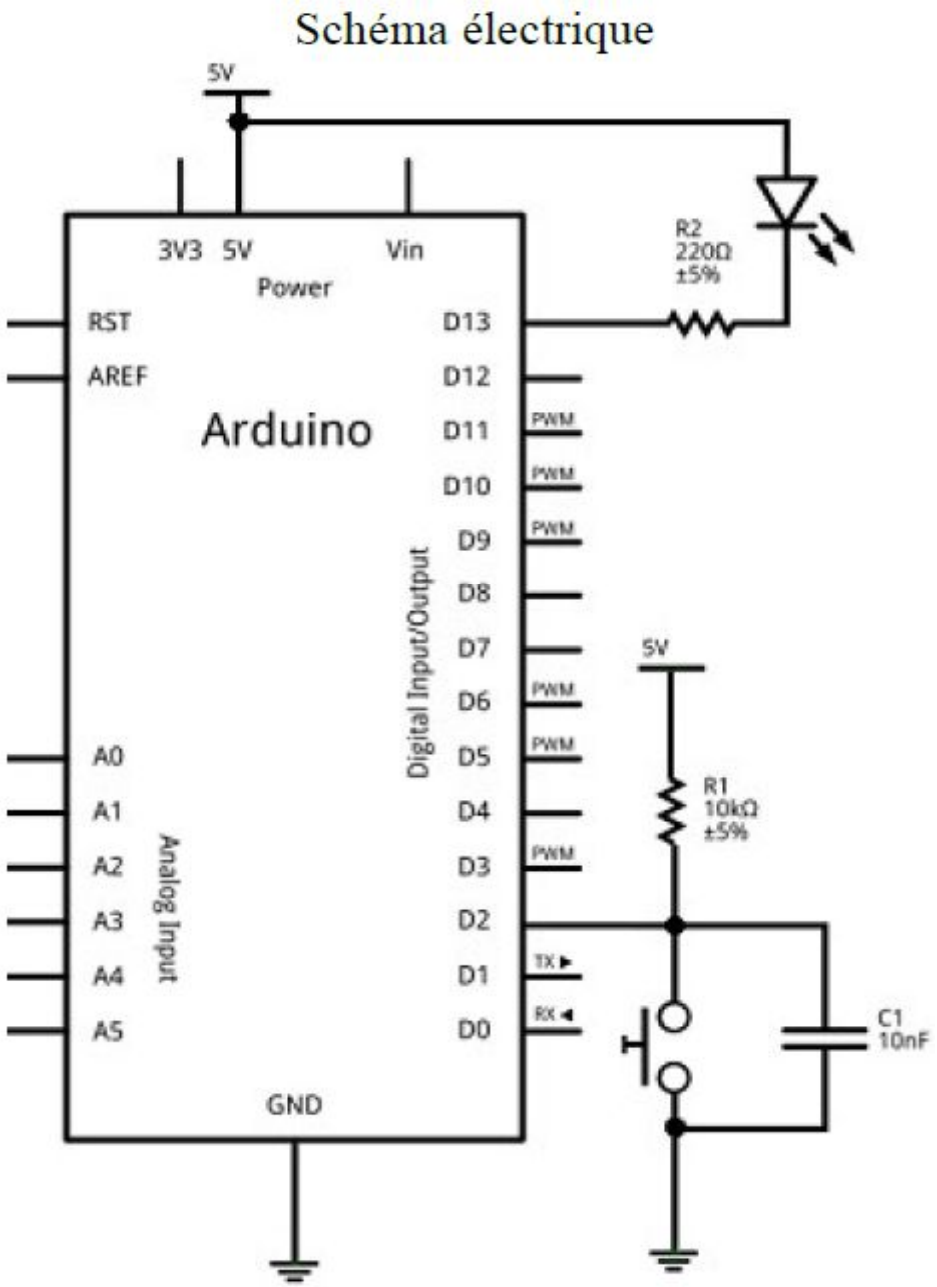
Les boutons ne sont pas des systèmes mécaniques parfaits. Du coup, lorsqu'un appui est fait dessus, le signal ne passe pas immédiatement et proprement de 5V à 0V. En l'espace de quelques millisecondes, le signal va "sauter" entre 5V et 0V plusieurs fois avant de se stabiliser. Il se passe le même phénomène lorsque l'utilisateur relâche le bouton. Ce genre d'effet n'est pas désirable, car il peut engendrer des parasites au sein du programme.

Le bouton poussoir



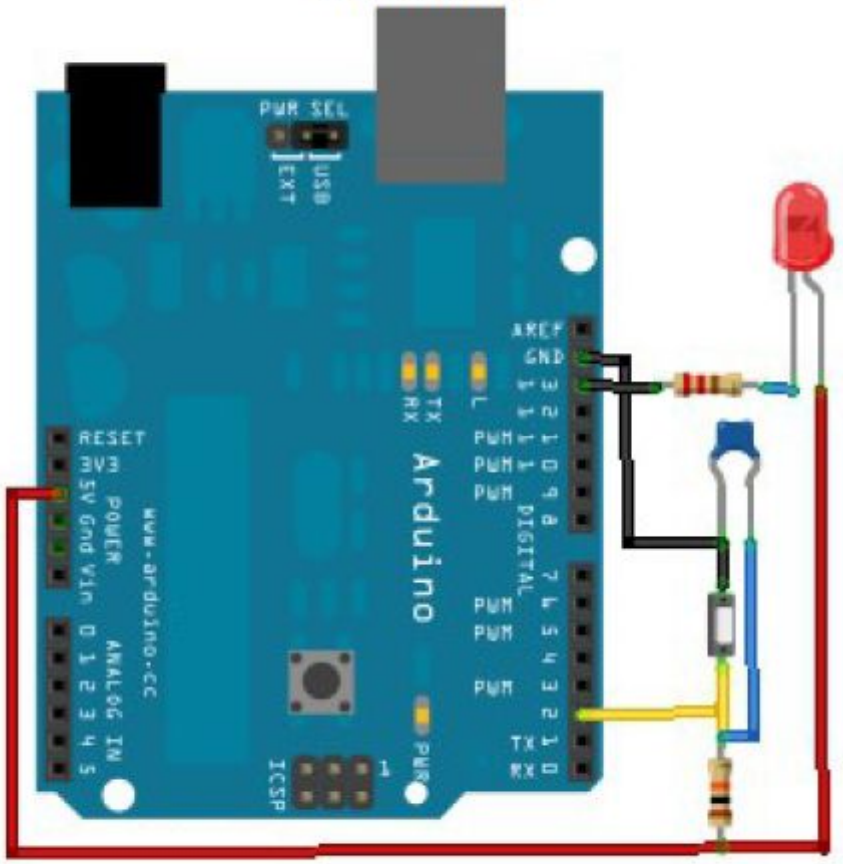
Pour atténuer ce phénomène, on utilise un condensateur en parallèle avec le bouton. Ce composant servira ici "d'amortisseur" qui absorbera les rebonds (comme sur une voiture avec les cahots de la route). Le condensateur, initialement chargé, va se décharger lors de l'appui sur le bouton. S'il y a des rebonds, ils seront encaissés par le condensateur durant cette décharge. Il se passera le phénomène inverse (charge du condensateur) lors du relâchement du bouton.

Le bouton poussoir



Le bouton poussoir

Réalisation



Lorsque le bouton est relâché, la tension à ses bornes sera de +5V, donc un état logique HIGH. S'il est appuyé, elle sera de 0V, donc LOW.

Le bouton poussoir

La fonction digitalRead() permet de lire l'état logique d'une entrée logique. Cette fonction prend u paramètre qui est la broche à tester et elle retourne une variable de type int.

Le programme suivant allume une LED lorsque le bouton est appuyé. Lorsque l'on relâche le bouton, la LED doit s'éteindre.

```
const int       bouton = 2;     //le bouton est connecté à la broche 2 de la carte Adruino
const int       led = 13;       //la LED à la broche 13

int             etatBouton;     //variable qui enregistre l'état du bouton

void setup()
{
        pinMode(led, OUTPUT);       //la led est une sortie
        pinMode(bouton, INPUT);     //le bouton est une entrée
        etatBouton = HIGH;          //on initialise l'état du bouton comme "relaché"
}

void loop()
{
        etatBouton = digitalRead(bouton);   //Rappel : bouton = 2

        if ( etatBouton == HIGH )     //test si le bouton a un niveau logique HAUT
        {
                digitalWrite(led, HIGH);      //la LED reste éteinte
        }
        else    //test si le bouton a un niveau logique différent de HAUT (donc BAS)
        {
                digitalWrite(led, LOW);       //le bouton est appuyé, la LED est allumée
        }
}
```

PROGRAMMATION D'UN ARDUINO

```
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // ledPin ou bien la pin 13 est configurée en sortie:
  pinMode(ledPin, OUTPUT);
  //  buttonPin ou pin 2 est configurée en entrée
  pinMode(buttonPin, INPUT);
}

void loop() {
  // On commence par lire l'état de la pin 2 qui est connectée à un bouton poussoir
  buttonState = digitalRead(buttonPin);

// Si le bouton poussoir est fermé il va donné un état HAUT :
  if (buttonState == HIGH) {
    // Dans ce cas la led connectée à la pin 13 en sortie va s'allumer
    digitalWrite(ledPin, HIGH);
  } else {
    // Sinon la Led va s'éteindre:
    digitalWrite(ledPin, LOW);
  }
}
```