Atelier Arduino

Initiation à la mise en oeuvre matérielle et logicielle de l'Arduino

novembre 2006

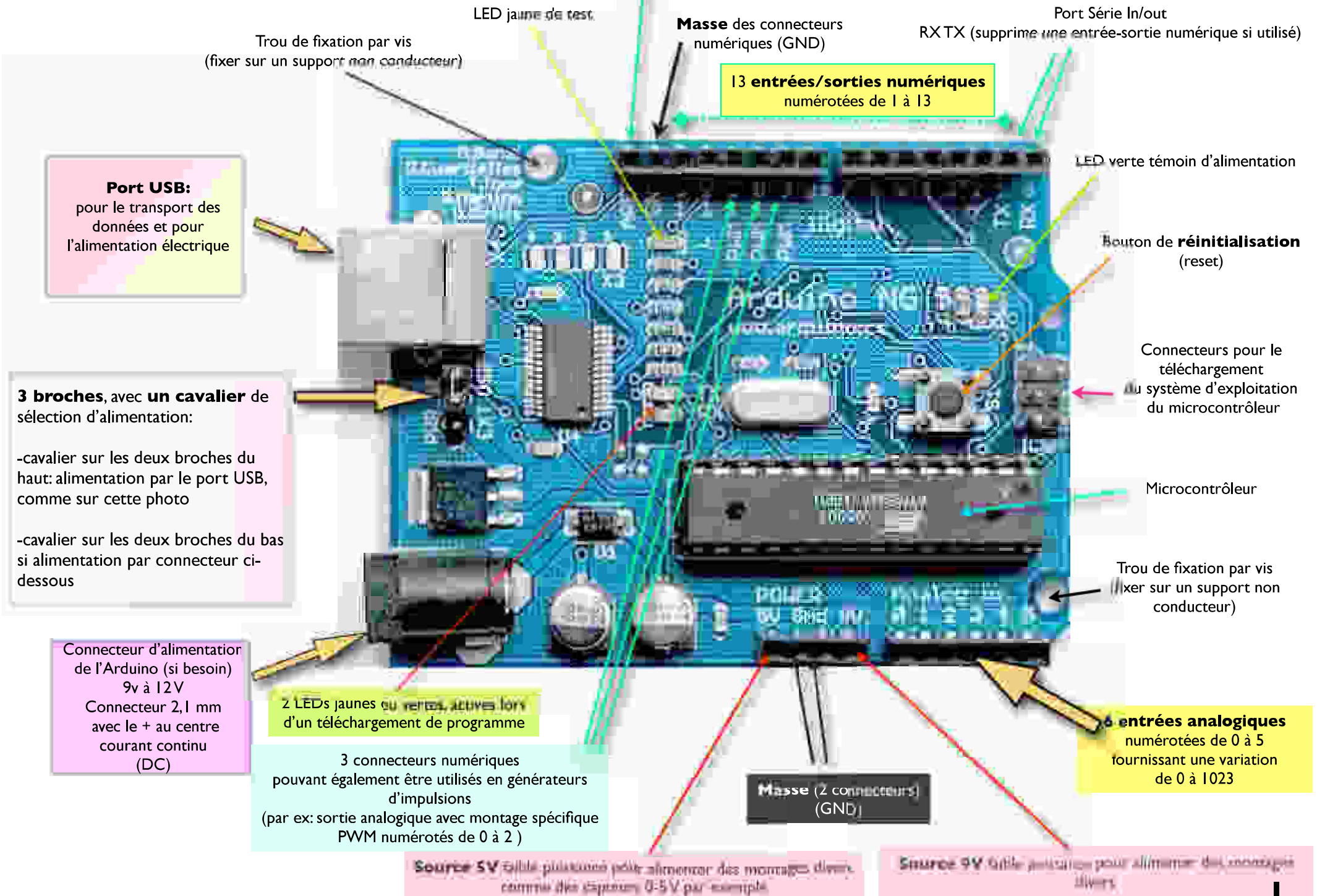
Centre de Ressources Art Sensitif
http://www.craslab.org
http://www.artsens.org

Sommaire

1. Description de l'Arduino
2. Présentation du projet Arduino
3. Téléchargement et installation du logiciel
4. Configuration du port série-USB
5. Interface du logiciel Arduino et fonctionnement d'un programme
6. Apprendre à programmer un fonctionnement autonome
7. Structure d'un programme
8. Introduction à la syntaxe Arduino
9. **Pas-à-pas**: Introduction
10. **Pas-à-pas**: Définition des variables
11. **Pas-à-pas**: Configuration logicielle de la carte
12. **Pas-à-pas**: Programmation des interactions
13. **Pas-à-pas**: Finitions
14. **Pas-à-pas**: Test et chargement du programme sur la carte
15. **Pas-à-pas**: Montage des composants
16. Syntaxe du langage Arduino
17. Syntaxe du langage Arduino suite
18. Librairies additionnelles
19. Un peu d'électronique interactive: précautions
20. Equipement pour l'électronique interactive
21. Electronique interactive : reconnaitre les composants 1
22. Electronique interactive : reconnaitre les composants 2
23. Electronique interactive : reconnaitre les composants 3
24. Electronique interactive : reconnaitre les composants 4
25. Alimenter l'Arduino
26. Quelques montages avec l'Arduino
27. Quelques schémas à expérimenter
28. Montage d'un capteur résistif
29. Montage d'interrupteurs et bouton-poussoirs
30. Exercices non corrigés 1
31. Exercices non corrigés 2
32. Le programme du montage de l'atelier
33. S'équiper en composants
34. Lexique Anglais technique/ Français
35. Ressources, remerciements, contact.

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, activés 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 Utile pour pouvoir alimenter des montages divers comme des capteurs 0-5 V par exemple

Source 9V Utile pour pouvoir alimenter des montages divers

L'Arduino est une carte électronique en Matériel Libre pour la création artistique interactive. Elle peut servir: 1/ pour des dispositifs interactifs *autonomes* simples 2/ comme *interface* entre capteurs/actionneurs et ordinateur 3/ comme programmateur de certains microcontrôleurs.

l'Arduino mini

l'Arduino USB

Le projet

Le projet Arduino comprend à la fois le développement matériel de cette carte, mais aussi le développement de son environnement de programmation, adaptation du logiciel de programmation pour la carte Wiring, lui-même construit sur le Logiciel Libre de gestion d'événements multimédia interactifs Processing (http://www.processing.org). L'Arduino n'est cependant pas exclusivement liée à Processing, et peut être utilisée en fonctionnement piloté avec la quasi totalité des logiciels de gestion d'événements multimédia interactifs. L'Arduino peut également être utilisée comme *carte de programmation* pour des microcontrôleurs AVR (http://fr.wikipedia.org/wiki/AVR) utilisables dans d'autres montages électroniques autonomes ou pilotés. Pour les utilisateurs chevronnés, la carte peut également être programmée en langage AVR-C.

La licence

L'Arduino est un Logiciel Libre et Matériel Libre sous license Creative Commons " paternité, non commercial et licence contaminante", disponible ici : http://creativecommons.org/licenses/by-nc-sa/2.5/deed.fr : toute liberté est permise à qui voudrait faire évoluer le matériel ou la plateforme de programmation dans le respect de la licence. Le site officiel du projet Arduino est http://www.arduino.cc

Technologie

L'Arduino est une carte basée sur un microcontrôleur (mini-ordinateur) Atmel ATMEGA8 ou ATMEGA168. Elle dispose dans sa version de base de 1 Ko de mémoire vive, et 8Ko de mémoire flash pour stocker ses programmes. Elle peut être connectée à 13 entrées ou sorties numériques, dont 3 PWM (pouvant donner 3 sorties analogiques : cf http://fr.wikipedia.org/wiki/PWM) et 6 entrées analogiques convertissant en 10 bit. Dans la version la plus courante, la communication avec l'ordinateur se fait par un port USB. Il existe plusieurs versions de l'Arduino, dont une version miniaturisée, et d'autres projets sont également en gestation. La carte dispose d'un logiciel système interne (modifiable) et des programmes utilisateur.

Téléchargement du logiciel et configuration de l'ordinateur:

Sur MacOSX

-Télécharger la version Mac PPC ou Intel du logiciel Arduino ici : http://www.arduino.cc/en/Main/Software 30 Mo environ
-Installer le logiciel Arduino dans le dossier Applications
-Installer le driver de la carte fourni dans le dossier Arduino (mot de passe puis redémarrage)
-Glisser déposer le script macosx_setup.command sur le logiciel "terminal" (qui est dans le dossier Applications/ Utilitaires, répondre "Y" , c'est à dire Yes, à la question qui se pose. Quitter le terminal.
-Et voilà ! la carte est prête à accueillir un programme Utilisateur

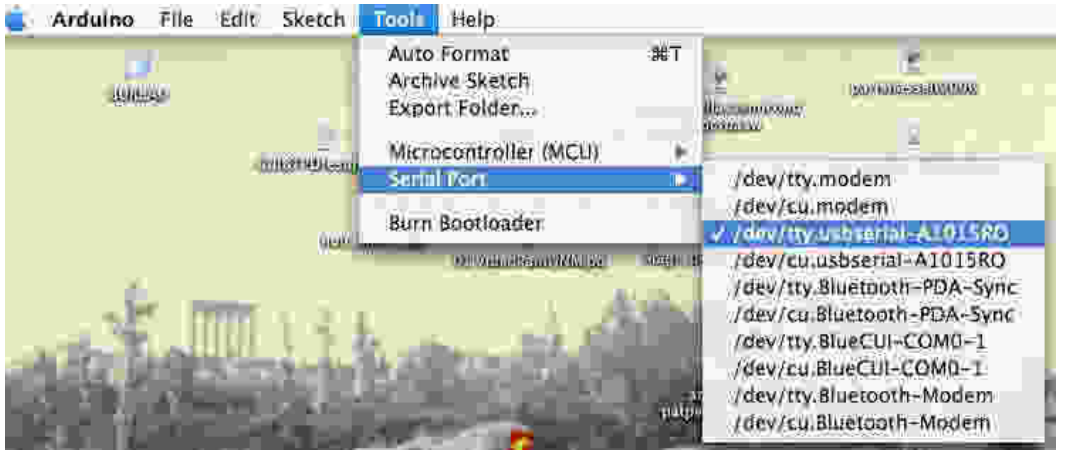
Sur Windows

-Télécharger la version Windows du logiciel Arduino ici : http://www.arduino.cc/en/Main/Software 50 Mo environ
-Installer le logiciel
-Dézipper le pilote FTDI USB Drivers.zip
-Brancher l'Arduino et pointer l'installeur Windows vers le pilote
-Et voilà ! la carte est prête à accueillir un programme Utilisateur

Sur Linux

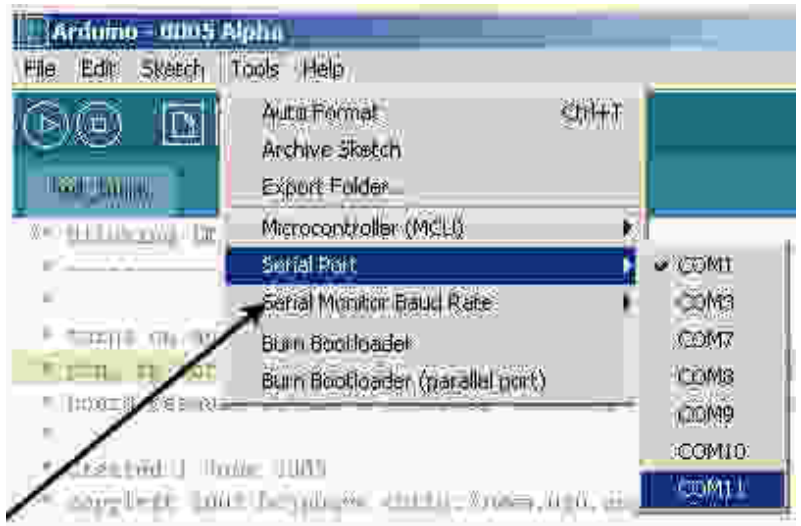
-Télécharger les sources du logiciel Arduino ici : http://www.arduino.cc/en/Main/Software 4 Mo environ
-Se diriger ici http://www.arduino.cc/en/Main/FAQ#linux pour les conseils de compilation sur les différentes plateformes Linux

Désigner le bon port Série (USB-Série)



Sur MacOSX
(les chiffres après tty.usbserial seront différents)

Si vous changez de carte arduino,
il faudra ré-indiquer le bon port.

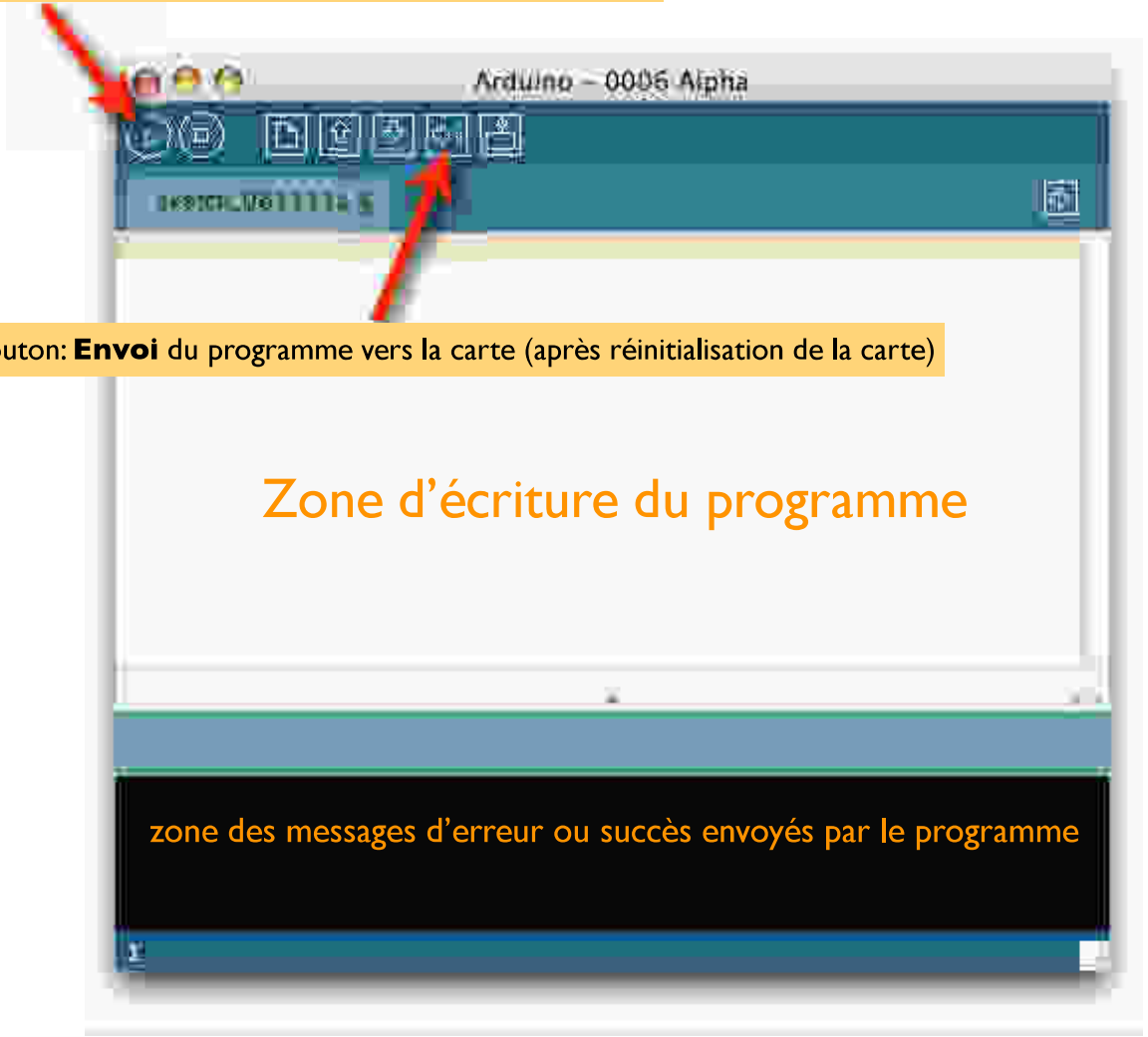


Sur Windows

Le logiciel:

C'est un logiciel de programmation par code, code qui contient une cinquantaine de commandes différentes. A l'ouverture, l'interface visuelle du logiciel ressemble à ceci: des boutons de commande en haut, une page blanche vierge, une bande noire en bas

Bouton :**Vérification** du programme après écriture = compilation



La méthode d'enregistrement d'un programme sur la carte:

Mise en oeuvre de l'environnement Arduino:

-On conçoit d'abord un programme avec le logiciel Arduino

-On vérifie ce programme avec le logiciel (compilation)

-Des messages d'erreur apparaissent éventuellement...on corrige puis vérifie à nouveau...

-On enlève le précédent programme sur la carte Arduino (Bouton réinitialisation)

-On envoie ce programme sur la carte Arduino dans les 5 secondes qui suivent l'initialisation

-L'exécution du programme sur la carte est automatique quelques secondes plus tard ou à ses prochains branchements sur une alimentation électrique (Alim 9/12V ou port USB).

Les deux pages suivantes vont être basées sur un exemple simple de programmation : faire clignoter une led.

- On verra tout d'abord quelle est la structure générale d'un programme

- On verra sur la page suivante la composition détaillée de ce même programme, en expliquant déja quelques mots de la syntaxe Arduino

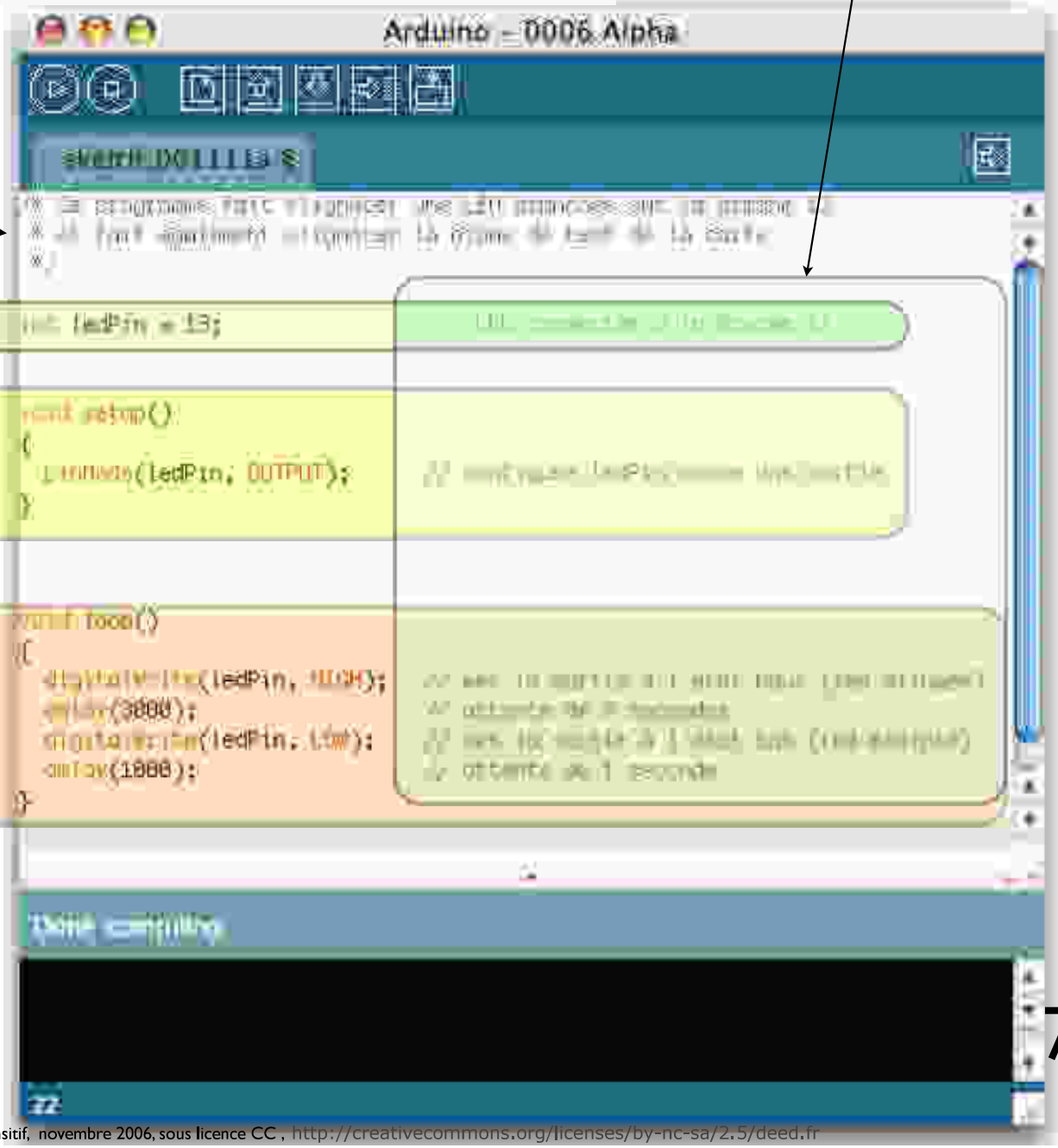
Un programme utilisateur Arduino est une suite d'instructions élémentaires sous forme textuelle, ligne par ligne. La carte lit puis effectue les instructions les unes après les autres, dans l'ordre défini par les lignes de code.

Structure d'un programme

Il y a trois phases consécutives:

Commentaires multilignes pour se souvenir du patch ==>

Commentaires



Une fois la dernière ligne exécutée, la carte revient au début de la troisième phase et recommence sa lecture et son exécution des instructions successives. Et ainsi de suite.

Cette **boucle** se déroule des milliers de fois par seconde et anime la carte.

Introduction à la syntaxe des commandes Arduino

La cinquantaine d'éléments de la syntaxe Arduino est visible ici http://www.arduino.cc/en/Reference/HomePage ainsi qu'à partir du document "index.html" (dans le dossier "Reference" que vous avez téléchargé avec Arduino), également accessible dans le menu "Aide" du logiciel.
Revoyons d'un peu plus près le programme de la page précédente, qui sert à faire clignoter une LED à partir d'une sortie numérique:

(Syntaxe en marron, paramètres utilisateur en vert)

Commentaires
Toujours écrire des commentaires sur le programme: soit en multiligne, en écrivant entre des /****/, soit sur une ligne de code en se séparant du code avec //

Définition des variables:
Pour notre montage, on va utiliser une sortie numérique de la carte, qui est par exemple la 13 ème sortie numérique. Cette variable doit être définie et nommée ici: on lui donne un nom arbitraire BrocheLED . Le mot de la syntaxe est pour désigner un nombre *entier* est int

Configuration des entrées-sorties void setup()**:**
Les broches numériques de l'Arduino peuvent aussi bien être configurées en entrées numériques ou en sorties numériques. Ici on va configurer BrocheLED en sortie.
pinMode (nom, état) est une des quatre fonctions relatives aux entrées-sorties numériques.

Programmation des interactions void loop()**:**
Dans cette boucle, on définit les opérations à effectuer, dans l'ordre:
• digitalWrite (nom, état) est une autre des quatre fonctions relatives aux entrées-sorties numériques.
•delay(temps en millisecondes) est la commande d'attente entre deux autres instruction
•Chaque ligne d'instruction est terminée par un point virgule
•Ne pas oublier les accolades, qui encadrent la boucle.

```
/* Ce programme fait clignoter une LED branchée sur la broche 13
 * et fait également clignoter la diode de test de la carte
 */



int BrocheLED = 13;     // Définition de la valeur 13 et du nom de la broche à utiliser




void setup()
{
  pinMode(BrocheLED, OUTPUT);      // configure BrocheLED comme une sortie
}



void loop()
{
  digitalWrite(BrocheLED, HIGH);   // met la sortie num. à l'état haut (led allumée)
  delay(3000);                     // attente de 3 secondes
  digitalWrite(BrocheLED, LOW);    // met la sortie num. à l'état bas (led éteinte)
  delay(1000);                     // attente de 1 seconde
}
```

Pas-à-pas d'une programmation en 5 étapes

Thème: une chaine capteur ==> actionneur

En rajoutant un capteur de luminosité sur le programme précédent, on voudrait faire varier la vitesse de clignotement. Construisons le programme étape par étape.

- **1/ Définition des variables**
- **2/ Configuration logicielle du matériel**
- **3/ Programmation des interactions**
- **4/ Finitions**
- **5/ Test et téléchargement**
- **6/ Montage**

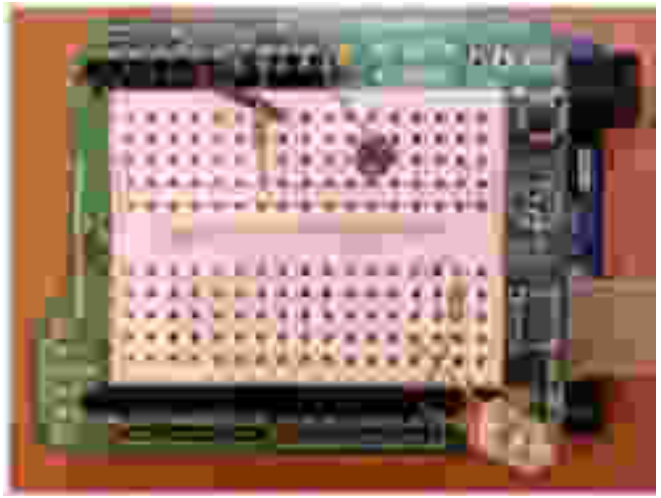
Pas à pas:

1/ Définition des variables

Qu'est ce qu'une variable ?
Une variable est un espace de stockage nommé qui permet de stocker une valeur utilisable par la suite dans la boucle d'un programme. Une variable peut aussi bien représenter des données lues ou envoyées sur un des ports analogiques ou numériques, une étape de calcul pour associer ou traiter des données, que le numéro 'physique' de ces entrées ou sorties sur la carte. Une "variable" n'est donc pas exclusivement un paramètre variant dans le programme.

*Pour composer un programme ex-nihilo, il est nécessaire de définir toutes les **composantes** d'entrée et de sortie qui vont affecter le **montage matériel** et les **calculs à effectuer.** Chaque entrée et chaque sortie sera une variable.*

Exemple:
Si on a **un** capteur (une cellule photo-électrique qui capte les variations de lumière par exemple) branché à une entrée de l'Arduino, et **un** actionneur (une LED) branché à une sortie l'Arduino, et si on veut que **la valeur** de luminosité change **la valeur** de l'intervalle de clignotement de la LED, alors on a 2 variables "stables" "pour la définition du matériel" et, en théorie, "2 variables pour les calculs " à déclarer en début de programme. En théorie seulement, car on va se servir directement de la valeur issue du capteur pour définir la valeur de l'intervalle de temps de durée d'extinction et durée d'allumage. On a donc besoin que de trois variables en tout. On va leur donner des noms arbitraires mais précis afin de bien les reconnaître dans le programme.



Le montage, ici réalisé sur une plaque d'expérimentation spécifique que l'on peut fixer sur l'Arduino

Matériel:
-cellule photo-electrique
-LED
-2 résistances

on verra ce montage en détail plus loin dans le livret

Au début de notre programme:

(Syntaxe en marron, paramètres utilisateur en vert)

Le code ci-dessous *déclare* (et dénomme arbitrairement) la variable capteur1, puis lui affecte (par exemple) le numéro de l'entrée analogique numéro 0. (L'Arduino possède 6 entrées analogiques numérotées de 0 à 5) :

```
int capteur1 = 0; // déclaration de la variable identifiant le portanalogique 0 de la carte
```

La ligne ci-dessous *déclare* (et dénomme arbitrairement) la variable LED1, puis lui affecte (par exemple) le numéro de la sortie numérique numéro 13 . (L'Arduino possède 13 entrées ou sorties numériques numérotées de 1 à 13) :

```
int LED1 = 13; // déclaration de la variable identifiant le autre port numérique 13 de la carte
```

La ligne suivante *déclare* (et dénomme arbitrairement) la variable qui correspond à la valeur de luminosité envoyée par le capteur. De plus, sa première valeur à l'allumage de la carte sera (arbitrairement) 0 :

```
int lum1 = 0; // déclaration de la variable identifiant la valeur de la luminosité du capteur 1
```


Nos trois variables sont maintenant déclarées et définies, passons à la configuration des entrées-sorties de la carte.

2/ Configuration logicielle du matériel

Rappel des premières lignes du programme: *(Syntaxe en marron, paramètres utilisateur en vert)*

```
int capteur1 = 0; // variable identifiant le port ana. 0 de la carte
int LED1 = 13; // variable identifiant le port num. 13 de la carte
int lum1 = 0; // variable identifiant la valeur de la luminosité du capteur 1
```

2/ Configuration du matériel (entrées et sorties)

Pour ce montage, il n'y a qu'une broche à configurer: la broche numérique sur laquelle on va brancher la LED (car elle pourrait être aussi bien configurée en sortie ou en entrée).

Ici, on va configurer cette broche numérique en sortie, car la LED est un actionneur. La broche d'entrée analogique pour le capteur n'est pas à configurer, car la carte Arduino possède 6 entrées analogiques qui ne font que cela.

après le void setup(), qui précise qu'on est à l'étape de configuration, on définit donc l'état de la broche 13 :

```
void setup()
{
  pinMode(LED1, OUTPUT);      // configure la broche 13 comme une sortie
}
```

et on ferme la phase de configuration par une accolade (touche clavier alt -parenthèse sur clavier français)

On peut maintenant passer à la boucle, c'est à dire le coeur du programme, qui définit les actions à effectuer avec ces variables.

3/ Programmation des interactions

Rappel des premières lignes du programme: *(Syntaxe en marron, paramètres utilisateur en vert)*

```
int capteur1 = 0; // variable identifiant un port ana. 0 de la carte
int LED1 = 13; // variable identifiant le port num. 13 de la carte
int lum1 = 0; // variable identifiant la valeur de la luminosité du capteur 1

void setup()
{
  pinMode(LED1, OUTPUT);      // configure la broche 13 comme une sortie
}
```

3/ Programmation de l'interaction

- On indique maintenant qu'on crée une boucle avec void loop() {
- Puis on effectue la première opération: lire la valeur du capteur = lire la variable lum1 identifiant la valeur de luminosité

```
lum1 = analogRead( capteur1);
```

- On peut maintenant allumer la LED

```
digitalWrite(LED1, HIGH);
```

- On patiente un certain temps: en fonction de la valeur de la variable luminosité lum1

```
delay(lum1);
```

- On peut maintenant éteindre la LED

```
digitalWrite(LED1, LOW);
```

- On patiente un certain temps: en fonction de la valeur de la variable luminosité lum1

```
delay(lum1);
```

- On peut maintenant boucler, avec une accolade, c'est-à-dire faire remonter automatiquement au début de la boucle pour lire la nouvelle valeur du capteur et ainsi de suite...jusqu'à ce qu'on éteigne l'Arduino.

```
}
```

Notre programme est terminé, terminons les commentaires si cela n'est pas déja fait

(Syntaxe en marron, paramètres utilisateur en vert)

```
/* Ce programme fait clignoter une LED branchée sur la broche 13
 * avec une vitesse de clignotement proportionnelle à l'éclairage ambiant
 * capté par une cellule photo-électrique.
 * JNM, 2006, Centre de Ressources Art Sensitif.
 */



int capteur1 = 0; // variable identifiant un port ana. 0 de la carte
int LED1 = 13; // variable identifiant le port num. 13 de la carte
int lum1 = 0; // variable identifiant la valeur de la luminosité du capteur 1

void setup()
{
  pinMode(LED1, OUTPUT);      // configure la broche 13 comme une sortie
}

void loop()
{
lum1 = analogRead( capteur1); // lire la donnée capteur
digitalWrite(LED1, HIGH);      // allumer la LED 1
delay(lum1);                  // attendre pendant la valeur donnée par le capteur en millisecondes
digitalWrite(LED1, LOW);       // éteindre la LED 1
delay(lum1);                  // attendre pendant la même valeur
}
```

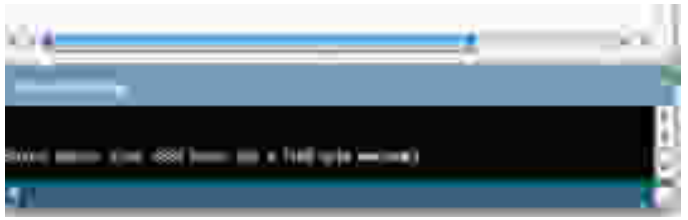
-Vérifions maintenant qu'un point-virgule finit bien chaque ligne de code, que les espaces soient bien placés
-Testons le programme sur le logiciel, avant de le transférer sur la carte:

5/ Test et téléchargement

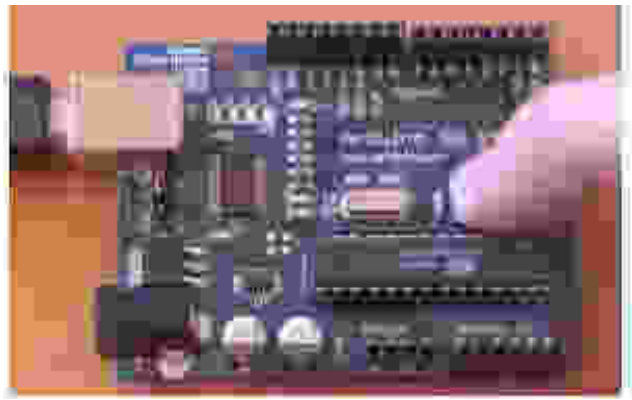
```
/* Ce programme fait clignoter une LED branchée sur la broche 13
 * avec une vitesse de clignotement proportionnelle à l'éclairage ambiant
 * capté par une cellule photo-électrique.
 * JNM, Centre de Ressources Art Sensitif.
 */

int capteur1 = 0; // variable identifiant un port ana. 0 de la
carte
int LED1 = 13; // variable identifiant le port num. 13 de la
carte
int lumT = 0; // variable identifiant la valeur de la luminosité
du capteur 1

void setup()
{
  pinMode(LED1, OUTPUT);      // configure la broche 13 comme une
sortie
}

void loop()
{
lum1 = analogRead( capteur1); // lire la donnée capteur
digitalWrite(LED1, HIGH);       // allumer la LED 1
delay(lum1);                    // attendre pendant la valeur donnée
par le capteur en millisecondes
digitalWrite(LED1, LOW);        // éteindre la LED 1
delay(lum1);                    // attendre pendant la même valeur
}
```

Vérifier

S'il n'y a pas d'erreurs on verra s'afficher: "done compiling", suivi de la taille du programme.



On peut sauver le fichier sur l'ordinateur, puis appuyer sur le bouton de ré-initialisation de la carte, ci-dessous.



Ré-initialiser

Télécharger

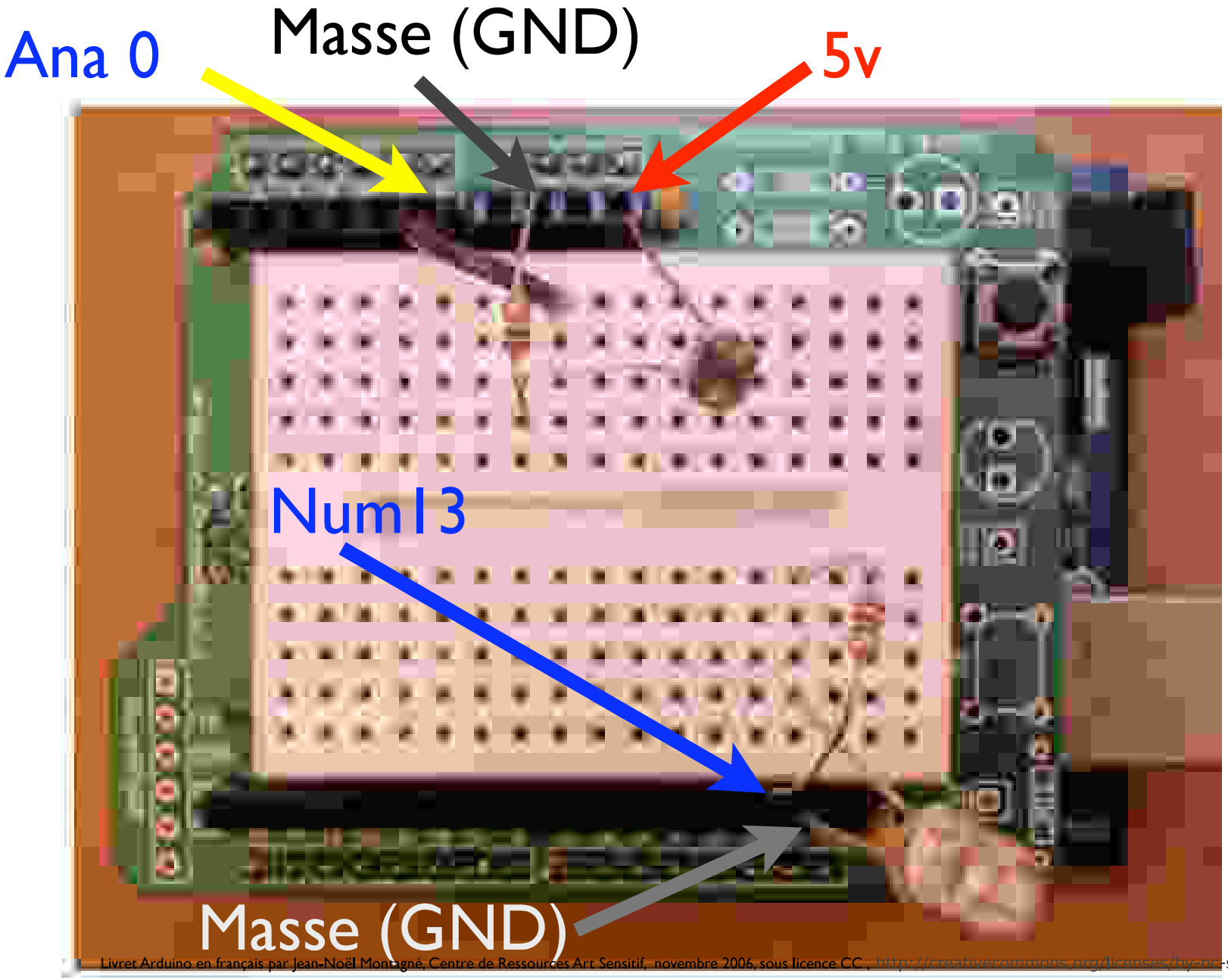
Et enfin, télécharger le programme sur l'Arduino: attention, vous avez 5 secondes après l'appui sur le bouton de ré-initialisation pour cliquer sur le bouton "Upload " !

Et voilà ! les deux petites LEDs TX RX sur la carte clignotent pendant le chargement, puis, quelques secondes plus tard, le programme se met en route.....jusqu'à ce qu'on éteigne la carte...

6/ Montage des composants

Le montage du tutoriel : Ce programme fait clignoter une LED branchée sur la broche 13
avec une vitesse de clignotement proportionnelle à l'éclairage ambiant capté par une cellule photo-électrique.

Matériel :une LED, une cellule photoélectrique, du cable, deux résistances (à appairer en fonction de la cellule et de la LED)...



Syntaxe du langage Arduino

Voir la page d'accueil du document références (menu "Aide" du logiciel) , qui renvoie vers les explications de chaque commande de la syntaxe Arduino et dont voici la table des matières. Chaque instruction est suivie de sa traduction, entre-parenthèses et en noir.

Commandes de structure du programme

Structure générale

- void setup() (configuration-préparation)
- void loop() (exécution)

Contrôle et conditions

- if (si...)
- if...else (si...alors...)
- for (pour...)
- switch case (dans le cas où...)
- while (pendant que ...)

Opérations de comparaison

- == (équivalent à)
- != (différent de)
- < (inférieur à)
- > (supérieur à)
- <= (inférieur ou égal à)
- >= (supérieur ou égal à)

Operations booléènnes

- && (et)
- || (ou)
- ! (et pas)

Autres commandes

- // (commentaire simple ligne)
- /* */ (commentaire multi-lignes)
- #define (donner une valeur à un nom)

Variables

Variables

- char (variable 'caractère')
- int (variable 'nombre entier')
- long (variable 'nombre entier de tès grande taille')
- string (variable 'chaine de caractères')
- array (tableau de variables)

Niveaux logiques des connecteurs numériques

- HIGH (état 1)
- LOW (état 0)
- INPUT (configuré en entrée)
- OUTPUT (configuré en sortie)

Fonctions

Entrées-sorties numériques

- pinMode(broche, état) (configuration des broches)
- digitalWrite(broche, état) (écrire un état sur une broche num.)
- digitalRead(broche) (lire un état sur une broche num.)
- unsigned long pulseIn(broche, état) (lire une impulsion sur une broche num.)

Entrées analogiques

- int analogRead(broche) (lire la valeur d'une broche ana.)
- analogWrite(broche, valeur) (PWM : écrire une valeur analogique sur les broches 9, 10 ou 11)

Gestion du temps

- unsigned long millis() (temps de fonctionnement du programme)
- delay(ms) (attente, en millisecondes)
- delayMicroseconds(us) (attente, en microsecondes)

Suite de la page d'accueil du document références (menu "Aide" du logiciel) , qui renvoie vers les explications de chaque commande de la syntaxe Arduino et dont voici la table des matières. Chaque instruction est suivie de sa traduction, entre-parenthèses et en noir.

Nombres aléatoires

- randomSeed(seed) (aléatoire 'piloté')
- long random(max) (aléatoire à partir de telle valeur)
- long random(min, max) (aléatoire entre deux valeurs)

Communications série entre Arduino et autres machines ou ordinateur

- Serial.begin(speed) (configuration de la vitesse de communication Série)
- Serial.available() (donne combien de caractères disponibles dans la zone tampon Série)
- Serial.read() (lit les données Série)
- Serial.print(data) (envoie des données Série)
- Serial.println(data) (envoie des données Série suivies de caractères spécifiques)

Librairies additionnelles

Matrix (gestion d'une matrice de LEDs par contrôleur MAX7219)

- matrix(data, load, clock) (Pour configurer des matrices de LEDs)
- matrix.write(x, y, value) (Pour envoyer des données aux matrices de LEDs)
- matrix.write(x, y, sprite) (Pour envoyer des données aux matrices de LEDs)
- matrix.clear() (efface l'écran de LEDs).
- matrix.setBrightness(value) (règle la brillance de l'écran)

Sprite (gestion de formes sur matrice de LEDs)

- Sprite(largeur, hauteur, colonne1, colonne2...) (Starts the LCD library.)
- sprite.width() (Returns the width in pixels of the sprite.)
- sprite.height() (Returns the height in pixles of the sprite.)
- sprite.write(x, y, valeur) (Writes data to an x, y position of the sprite.)
- sprite.read(x, y) (Returns the data stored on the x, y position of the sprite.)

SimpleMessageSystem (librairie pour communiquer avec tous les logiciels à messages sous forme de caractères ASCII, comme PD ou Max), documentée ici http://www.arduino.cc/playground/Code/SimpleMessageSystem

Wire, une librairie pour connecter l'Arduino dans un réseau de capteurs en communication à deux fils, comme dans la carte Wiring

PDuino, un programme interne pour controler des fonctions avancées avec Pure Data http://at.or.at/hans/pd/objects.html

Vous savez maintenant programmer des opérations simples, mais comment aborder la partie électronique ?

Faire des montages électroniques simples est à la portée de tous et les ressources sur l'électronique interactive (physical computing) sont multiples sur le web, il y a cependant des notions de base à avoir pour se lancer dans la réalisation de ses propres montages, même si on ne fait que copier un montage sans le comprendre :

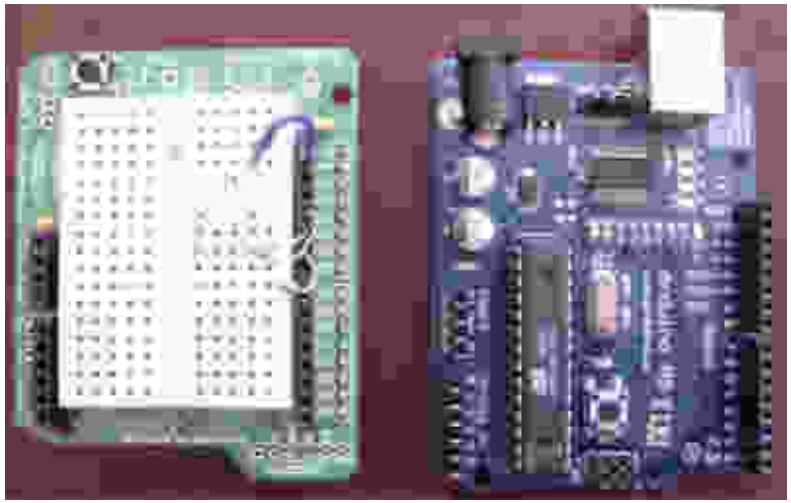
- il est toujours utile de savoir reconnaître les composants
- il est toujours utile de savoir déterminer la valeur d'un composant (le code visuel des couleurs, les sigles et les abréviations)
- il est (souvent) utile de connaître la fonction d'un composant
- il est (parfois) utile de savoir lire un schéma, c'est-à-dire reconnaître les symboles des composants, et la raison des cablages.

Il faut aussi prendre quelques précautions:

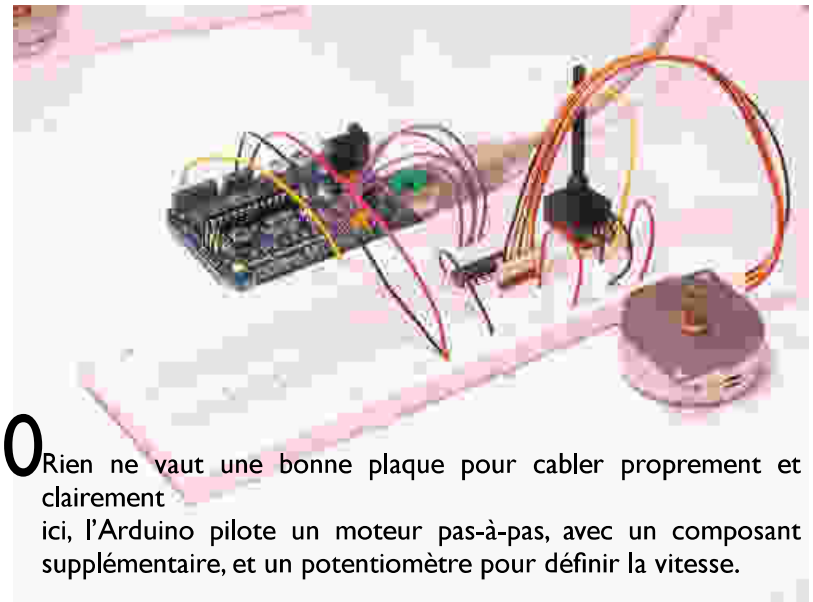
- Eviter de faire des court-circuits
- Utiliser les sources électriques recommandées
- Certains composants électroniques ont un sens = ils sont polarisés. Exemple: la LED, certains condensateurs, les diodes..
- Certains composants ne peuvent pas fonctionner seuls , comme la LED, qui a besoin d'une résistance appropriée pour limiter le courant.
- Certains composants se ressemblent mais n'ont pas du tout la même fonction: toujours bien regarder leur signalétique.
- **Ne pas manipuler de 240V ou 110 V sans connaissances appropriées.**
- **Préferer faire les essais et les montages avec une alim externe plutôt que celle de l'USB, ce qui évitera de griller votre port USB de l'ordinateur (très courant)**

Les ressources du web, et notamment de Wikipedia vous aideront sur les concepts les plus difficiles.

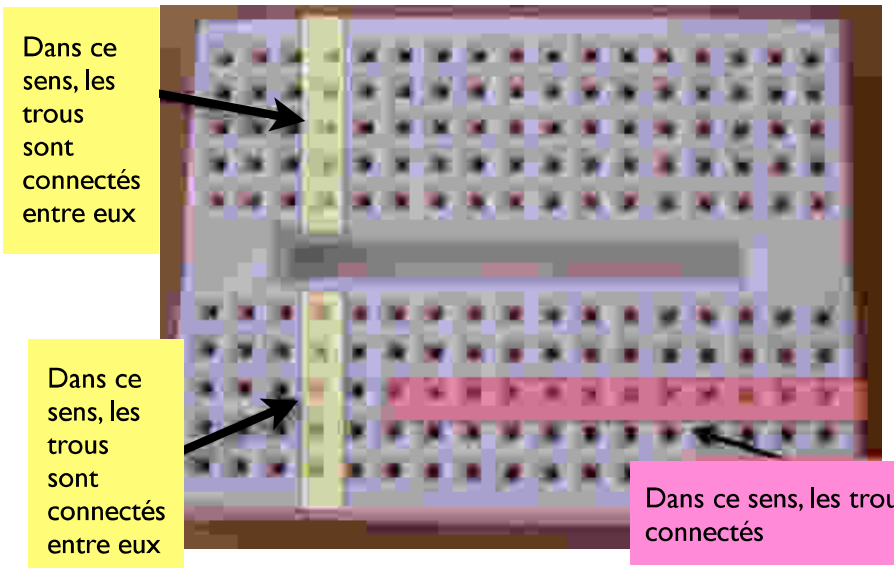
Une plaque d'expérimentation (breadbord en anglais) permet de cabler de nombreux composants sans faire de soudure, et en gardant un montage entièrement démontable. Le projet Arduino propose une plaque adaptée à l'Arduino, dotée de connecteurs reproduisant exactement le plan d'implantation de la carte. On peut aussi se la bricoler soi-même avec quelques composants comme montré sur le site http://todbot.com/blog/2006/07/11/arduino-breadboard-shield/

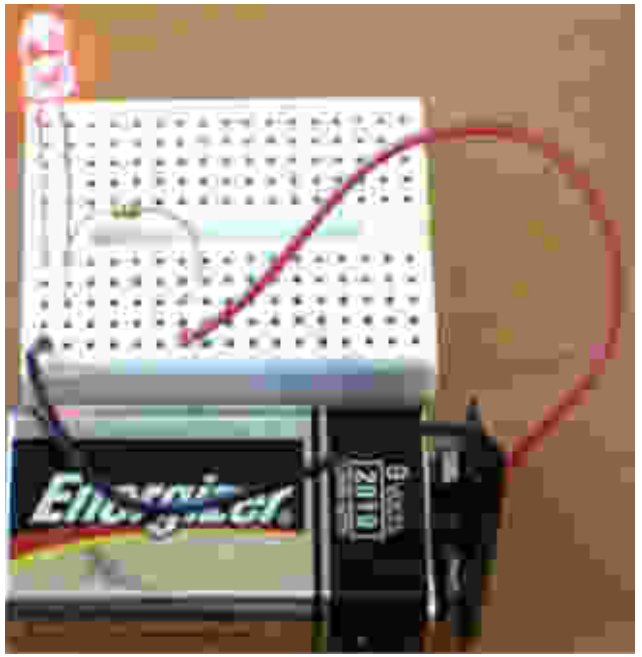


La plaque d'expérimentation Arduino "officielle": ici démontée, car elle vient normalement couvrir l'Arduino



Rien ne vaut une bonne plaque pour cabler proprement et clairement
ici, l'Arduino pilote un moteur pas-à-pas, avec un composant supplémentaire, et un potentiomètre pour définir la vitesse.





Une pile, une LED, sa résistance , voilà un bon début pour commencer sur une plaque.

Si les fils souples de la pile ne veulent pas rentrer, on peut les étamer avec un peu de soudure.

Electronique interactive
Reconnaitre les composants/1

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

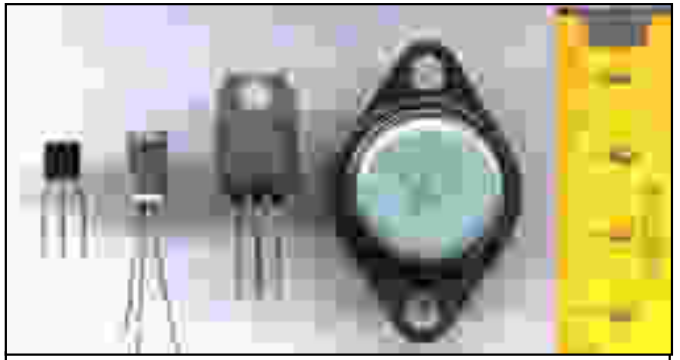
Les condensateurs peuvent stocker un peu de courant si on les charge, mais comme un tonneau percé, ils renvoient ce courant instantanément si ils sont branchés à un organe consommateur de courant.

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

non polarisé polarisé

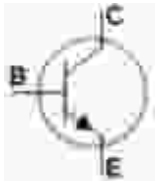
Symbole

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

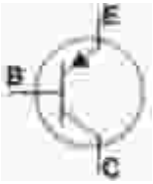


Le transistor

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.



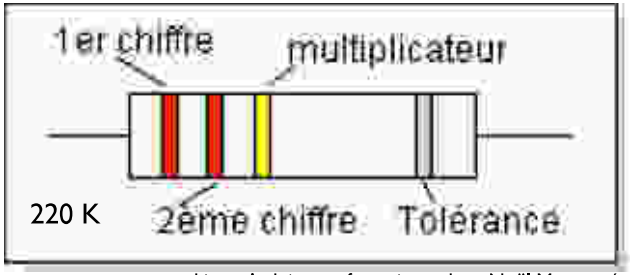
Symboles



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%

Electronique interactive

Reconnaitre les composants /2

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.

ERROR: undefinedresource
OFFENDING COMMAND: findresource

STACK:

/40
/CSA
/40