Extrait du PoBot

http://www.pobot.org

Débuter avec une carte Arduino

- Composants et technique - Les contrôleurs - L'univers Arduino -

Date de mise en ligne : samedi 11 octobre 2008

Description :

un petit guide pour apprendre à se servir d'une carte Arduino, qui permet de s'exercer à la programmation des micro-contrôleurs et que nous utilisons pour des essais, du prototypage rapide et de la formation.

PoBot

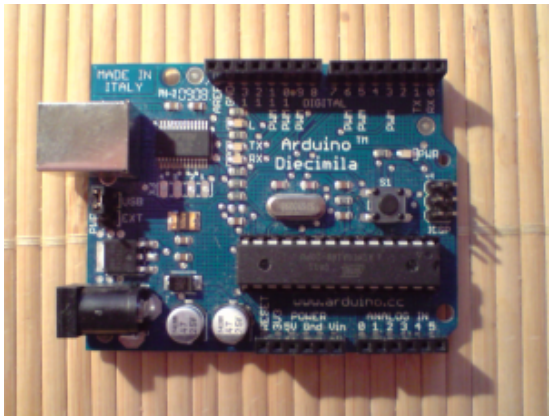
Sommaire

Il existe de nombreuses cartes à base de microcontrôleurs. Nous en avons décrites plusieurs sur ce site car chacune a des spécificités qui la distingue des autres.

La carte que nous allons présenter dans cet article est particulière car sa programmation est plus facile et convient bien pour les grands débutants. La carte présentée est la version Diecimilla de la famille Arduino. De nouvelles versions sont sorties (Uno, Duemilanove) mais le fonctionnement est identique.

> Arduino : c'est simple, c'est pratique et nous sommes là pour vous aider !

C'est un projet Open Source [http://www.arduino.cc] qui est en train de devenir un standard en électronique embarquée ludique.



Arduino Diecimilla
vue de dessus de la carte

Voici ses caractéristiques principales :

hardware

- une carte de 5.5 cm sur 7 cm d'une épaisseur de 1.5 cm
- avec un microcontrôleur Atmel AVR 8bits
- des connecteurs pour toutes les entrées et sorties, numériques et/ou analogiques
- des composants permettant l'utilisation du port USB en programmation et en communication
- un connecteur USB type B (standard carré)

- un connecteur d'alimentation (voir plus bas pour les détails)

software

- un environnement de programmation unique regroupant éditeur de code, compilation et debug
- un langage simplifié
- et des bibliothèques de code permettant d'étendre les fonctionnalités du micro-contrôleur.

Introduction

Tout d'abord, précisons qu'il y a déjà des ressources en français sur les cartes Arduino :

- la version française du site officiel [http://www.arduino.cc/fr/] (en cours de traduction)
- le document du Craslab d'Aix-en-Provence [http://www.craslab.org/arduino/LivretArduinoFr06.pdf]
- le site de Benoît Rousseau [http://pagesperso-orange.fr/rousseau-benoit/referencearduino/sommaire.xhtml]

Nous allons ici essayer de donner des éléments pour une prise en main étape par étape, qui sera complétée en fonction de vos retours d'expérience et de vos questions (forum en bas de l'article ou mail via le lien "contact" dans le menu de gauche).

Premiers pas

Le but de l'exercice est d'écrire un premier programme, le transférer dans la carte et de l'exécuter. Le cahier des charges est de faire s'allumer une lumière (LED) présente directement sur la carte.

Pour réaliser cet exercice pratique, il vous suffit d'avoir :
- une carte Arduino Diecimilla (ou une autre mais il faudra vous adapter)
- un câble USB
- un ordinateur

Une led simplement connectée à l'Arduino

C'est possible sur le port 13 : pas celle du microcontrôleur, celle de l'Arduino Diecimilla qui utilise des numéros continus 1-13 pour les entrées/sorties numériques et les 0 à 5 pour l'analogique pour ne pas complexifier avec la notion de ports 8 bits A, B, C et D.

Il y a une résistance de 1k sur cette patte, car une résistance est déjà présente

Et c'est tout !

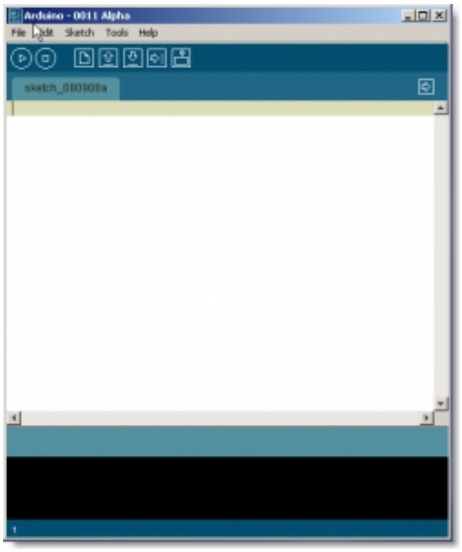
Installation du logiciel

Téléchargez la dernière version du logiciel Arduino disponible sur http://www.arduino.cc/en/Main/Software. Cette carte fonctionne sur Windows, Macintosh et même Linux en suivant leurs conseils d'installation.

Le programme fait près de 55 Mo (version 0011 Windows) car il contient tous les outils qui permettront de fonctionner sans aucun autre programme !

Selon votre machine, procédez à son installation. Par exemple sous Windows, il suffit de dézipper dans un répertoire de votre disque dur, facile non ?

Dans le répertoire d'installation, lancez le programme arduino (ex : arduino.exe sous Windows). Cette application est basée sur les mêmes principes de simplicité qu'une application de programmation Java appelée "Processing" et qui permet en deux clics de compiler et de lancer le programme développé.



Fenêtre principale Arduino

Ainsi, vous constaterez que l'application est composée uniquement d'une fenêtre d'édition de texte, d'un menu et d'une barre de boutons.

Cette barre de boutons permet de compiler ("Verify" est un résumé), d'arrêter la compilation (car dans le cas de certains programmes complexes, cela peut s'avérer long), de transférer le programme dans la carte Arduino et d'ouvrir une fenêtre spéciale de communication entre l'ordinateur et la carte (aussi appelée "console série"), et bien sûr d'ouvrir, de sauver et de créer des fichiers.

Connexion

Une fois faite l'acquisition de cette carte, vous pouvez la déballer et commencer à vous en servir en connectant la carte via un cable USB (prise A / prise B). L'alimentation est fournie par l'ordinateur et la carte est reconnue comme un

nouveau port série (COMx où x est un nombre supérieur ou égal à 1, en fonction des équipements du même type qui ont déjà été connecté à votre ordinateur).

Le code

```
int ledPin = 13;                // la led est présente sur la carte Arduino Diecimilla connectée à la sortie numérique 13
// initialisation
void setup() {
  // patte en sortie
  pinMode(ledPin, OUTPUT);
}
// boucle sans fin
void loop() {
  // on allume la led
  digitalWrite(ledPin, HIGH);
  // on attends une seconde
  delay(1000);
  // on éteint la led
  digitalWrite(ledPin, LOW);
  // on attends une seconde
  delay(1000);
  // et on recommence (ne jamais oublier la dernière attente)
}
```

Premier exercice : dialoguer avec la carte

Le but est d'envoyer des commandes (un caractère) à la carte Arduino. On va donc utiliser une communication série, en utilisant les fonctions "Serial" du langage Arduino.

Ecrivez le code suivant :

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  while (Serial.available())  {
    char key = Serial.read();
    switch (key)  {
    default:
      Serial.print("commande ");
      Serial.print(key);
      Serial.println(" non reconnue.");
      break;
    }
  }
}
```

Compilez (CTRL+R pour aller vite) et chargez dans la carte. L'environnement Arduino propose une console "terminal série" (dernier bouton de la ligne, "Serial Monitor"), cliquez sur l'icone, sélectionnez la bonne vitesse (9600 dans l'exemple ci-dessus) et saisissez des caractères au clavier dans le champ de saisie à droite du sélecteur de vitesse, puis faites "entrer" au clavier ou cliquez sur le bouton "send" : la carte réagit. Il suffit ensuite d'écrire des cas différentes selon ce que vous voulez associer comme commande :

```
switch (key)  {
  case 'l':
    analogOutput(200);
    break;
  case 'r':
    analogOutput(5);
    break;
  default:
    Serial.print("commande ");
    Serial.print(key);
    Serial.println(" non reconnue.");
    break;
}
```

Attention, n'oubliez pas l'instruction "break" qui sert à sortir du "switch", sinon vous enchaîneriez les différents cas (c'est parfois utile quand on veut écrire une séquence d'actions et pouvoir y entrer à n'importe quelle étape).

Second exercice : piloter un servomoteur

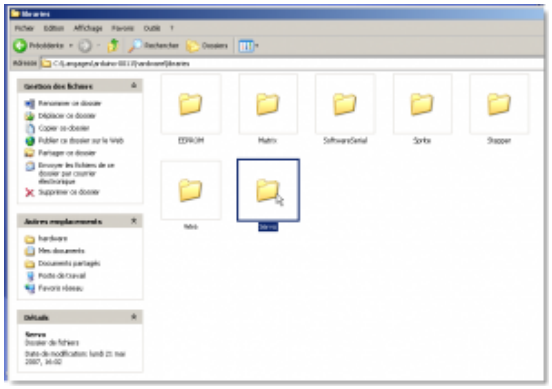
On va maintenant utiliser la carte Arduino pour piloter la rotation d'un servomoteur. On a déjà présenté sur ce site en détail le fonctionnement d'un servomoteur et comment le faire tourner. Avec la carte Arduino, on va se simplifier la vie en utilisant une bibliothèque existante, c'est-à-dire un code qui exécute les commandes de contrôle en offrant une liste de fonctions simples qui réduisent grandement notre propre code.

Bibliothèque

On utilise la bibliothèque Servo documentée (en anglais) sur le site Arduino : http://www.arduino.cc/playground/Co... [http://www.arduino.cc/playground/ComponentLib/Servo]

Bibliothèque "servo"
pour piloter des servomoteurs avec une carte Arduino

Sur les anciennes versions il faut télécharger le zip et extraire le répertoire Servo dans votre répertoire d'installation du logiciel Arduino. Maintenant c'est désormais intégré en standard.



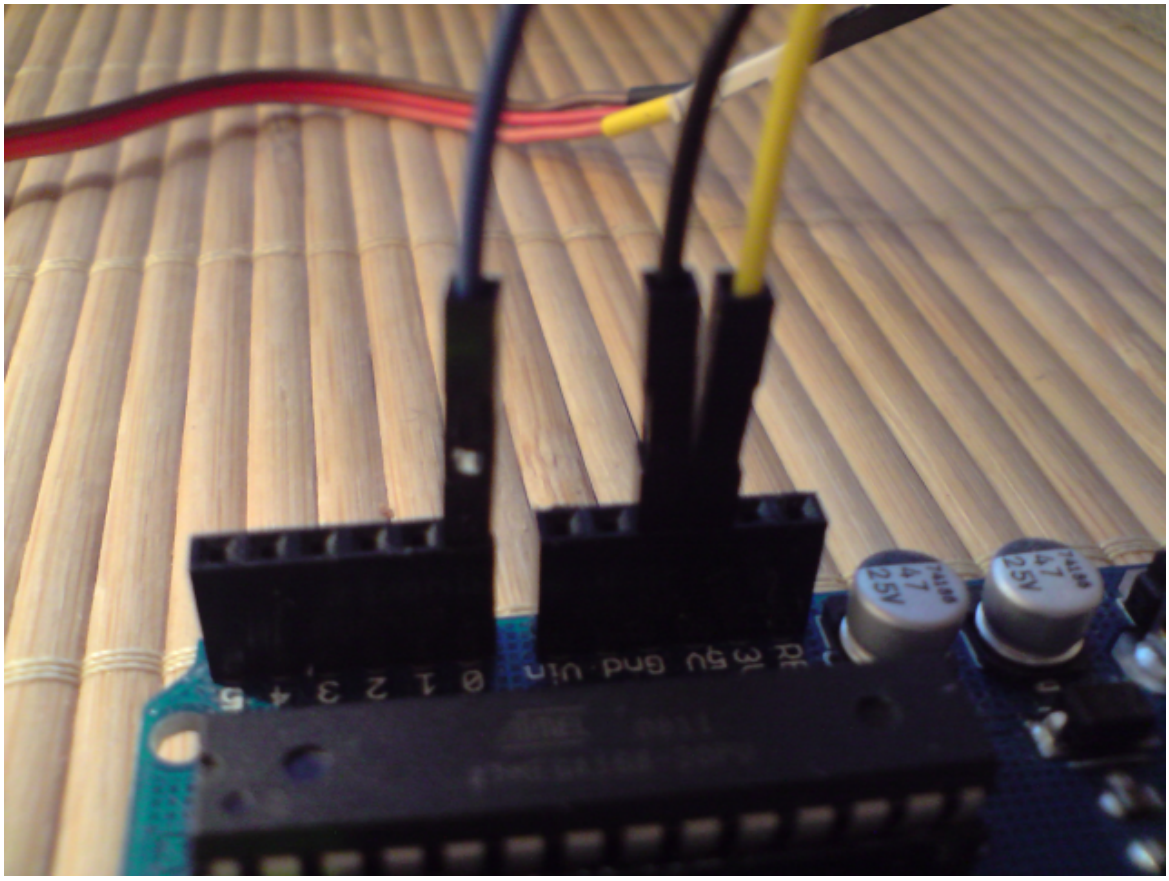
Installation de la bibliothèque
Notez bien le chemin vers le répertoire d'installation du logiciel Arduino.

Montage

Rien de plus simple, on va relier ensemble le câble de connexion du servo à la carte Arduino. Les connecteurs des servomoteurs étant différents selon les constructeurs, voici la liste de correspondance.

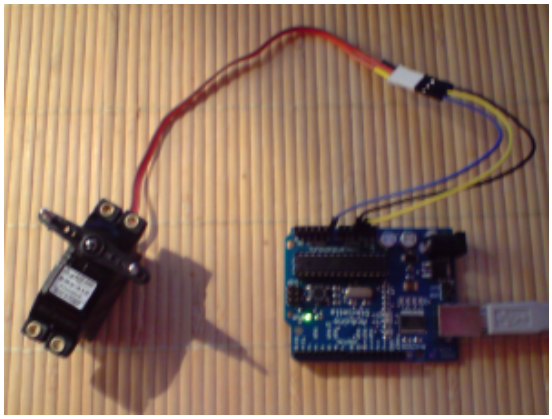
Rôle	Masse	Tension	Signal
Arduino	GND	5V	DIGITAL 9
type 1	noir	rouge	blanc
type 2	marron	rouge	orange

On va connecter selon le choix indiqué ci-dessus le signal de commande du servomoteur à la patte "DIGITAL 9" de la carte Arduino.



Le connecteur d'un servomoteur étant généralement femelle, comme les connecteurs d'E/S [1] de l'Arduino, j'utilise des câbles mâle/mâle bien pratiques que l'on trouve un peu partout sur le Net (voir nos fournisseurs). Vous pouvez aussi utiliser un simple fil de cuivre dénudé aux extrémités.

Le montage Arduino + servo terminé est simple comme vous pouvez le constater :



Connexion d'un servomoteur sur la carte Arduino

Programmation

Voici le code pour la carte Arduino :

```
#include <Servo.h> #define INCR  1  // on déclare le servomoteur comme une variable typée Servo servo1;  // l'angle actuel du servomoteur int angle = 0; // le sens actuel du servomoteur int sens = 0;  /**  * Initialisation du programme  */ void setup() {   servo1.attach(9); // le servomoteur est sur une patte PWM }  /**  * Boucle infinie qui va gérer la rotation et faire alterner le sens de rotation   */ void loop() {     // faire tourner le servomoteur     if (sens == 0)     {       angle += INCR;     }     else {       angle -= INCR;     }     // si on est arrivé à une extremité     if (angle == 0)     {       sens = 0;     }     if (angle == 180) {       sens = 1;     }     delay(10);     servo1.write(angle); }
```

(note : ne tenez pas compte de la dernière ligne, c'est un bug de la mise en couleur du code)

Compilez (touche triangulaire à gauche de la barre de menu) et chargez dans la carte (avant dernière icone de la barre de menu).

Si vous avez cette erreur à la compilation, c'est que vous avez oublié d'installer la bibliothèque "Servo" (voir ci-dessus) :

> 19 : error : Servo.h : No such file or directory In function 'void setup()' : In function 'void loop()' :

Et avec un peu d'entrainement, voilà ce que vous pourrez programmer :

Pour aller plus loin

On va maintenant chercher à débrancher la carte du PC pour que le programme que vous avez développé s'exécute dans l'environnement cible (robot, objet intelligent, ...). Il va donc falloir lui fournir une alimentation.

Caractéristiques d'alimentation

Note : ceci est une traduction en français des informations fournies sur le site officiel concernant la carte Arduino Diecimilla avec quelques compléments pour les débutants.

La carte Arduino Diecimila peut être alimentée via le câble USB (voir premiers pas) ou par une alimentation 'externe' :

- un adaptateur secteur qui convertit le 220V alternatif qui sort du mur en tension continue, aussi connu comme bloc d'alim standard tel qu'on en trouve avec une imprimante, un disque dur externe, une perceuse sans fil, etc...)

- une pile ou des accumulateurs rechargables, qui sont une source d'alimentation bien pratique pour un système mobile (robot, gadget portable).

La source à utiliser doit être sélectionnée sur la carte en utilisant un jumper à deux positions : un cavalier (jumper) à cheval sur les deux pattes du haut, i.e. vers le connecteur USB type B métallique ou sur les deux du bas vers le connecteur 'jack' femelle noir en plastique.

La tension continue en provenance d'une source "externe" peut être connectée sur ce connecteur jack (le pôle + est au centre) ou pour les batteries (piles, accus), directement sur les pattes (connecteur femelle) GND pour la masse (0 volts) et Vin (voltage in) pour la tension nominale. Un régulateur à faible chute de tension (low dropout ou LDO) assure la conversion en 5 volts.

Cette tension externe peut être comprise entre 6 et 20 volts (au passage, 6 volts pour en fournir 5, c'est ce qu'on appel une faible chute de tension). Si la tension d'entrée descend en dessous de 7 volts, la tension de sortie qui alimente la puce et toute l'électronique "logique" (en gros, toute la carte) va descendre en dessous de 5V et le comportement de la carte sera instable. A l'inverse, si vous fournissez une trop grosse alimentation (supérieure à 12V), le régulateur de tension va chauffer et peut endommager la carte.

La tension d'alimentation externe recommandée se situe donc entre 7 volts et 12 volts. Respectez ces valeurs

Rappel sur le connecteur de puissance :

- VIN : la tension d'entrée qui alimente la carte Arduino quand on utilise une source dite "externe" (par opposition à l'alimentation par câble USB ou toute source 5V régulée). Cette patte du connecteur femelle est totalement équivalente à la tension du connecteur jack femelle en plastique noire, vous pouvez utiliser indépendamment l'un ou l'autre. Attention, pas les deux en même temps ! (c'est à dire pas une pile sur Vin et une alimentation d'un bloc d'alim sur le jack).

- 5V : la tension régulée qui alimente le microcontrôleur et l'ensemble des autres composants. Soit c'est la sortie du régulateur low-dropout si vous utilisez l'alim externe, soit c'est le même 5V que votre port USB. Ici aussi, pas de blague en connectant une deuxième source de 5V s'il y en a déjà une que vous auriez oublié (le câble USB avec le connecteur (cavalier, jumper) en position USB.

- 3V3 : une autre tension régulée, cette fois-ci à 3.3 volts car c'est une tension qu'on retrouve de plus en plus en électronique (puces de communication sans fil, certains capteurs, certains circuits intégrés). D'ailleurs pas de régulateur low-drop, c'est la puce FTDI qui gère les communications séries/UART/USB sur la carte Arduino qui sort cette tension adaptée. Le courant maximum sur ce port est de 50 mA.

- GND : la masse, car GrouND en anglais.

Intensité

Après avoir parlé des tensions, parlons de courant : l'intensité en sortie des connecteurs de la carte Arduino est limitée par les possibilités du microcontrôleur. Soit, pour l'ATmega168, une intensité maximale fournie à chaque patte d'entrée sortie de 40 mA. Lire la suite pour les limites.

En effet, voici des compléments intéressants suite à quelques questions d'un de nos visiteurs :

> Quel ampérage il me faut pour l'alimentation ... je suppose que c'est 1 ampère ...

Je vous vois déjà faire le calcul : une vingtaine de pattes à 40 mA, ça fait dans les 800mA.

Et bien non, car il ne faut pas oublier que pour cette famille de microcontrôleurs Atmel AVR (et peut être valable pour les autres familles), chaque port est limité à un courant total de 200 mA, et le microcontrôleur a une limite globale de 400 mA. De plus, un port USB standard (celui du PC où vous brancherez l'Arduino) fournit au maximum 500 mA (voire beaucoup moins pour un ordinateur portable).

Donc une source d'alimentation qui fournit un maximum de 500 mA suffit.

Mais bien sûr qui peut le plus peut le moins, et si le bloc d'alim qu'on vous propose fait moins de 10 euros, alors 1A iront très bien.

> Comment est redistribuée l'électricité ? La carte Arduino s'occupe-t-elle elle même de de redistribuer le courant ? Dans le sens où je doit calculer tout ce que je vais mettre dessus ou pas ? LED et servomoteurs

Et bien tout dépend du composant. Dans le cas des servomoteurs, la patte de sortie de l'Arduino va juste piloter (donner les consignes de rotation) à l'électronique interne du servo, tandis que les pattes masse et tension du moteur sont reliées aux pattes du connecteur de puissance, donc avec autant de courant disponible que votre alimentation peut en fournir (500mA dans le cas de l'adaptateur secteur discuté ci-dessus, mais dix fois plus si c'est une pile ou un accu rechargeable). Donc aucun calcul nécessaire.

Dans le cas de la led, c'est différent. Vous n'avez pas à calculer le courant disponible pour le limiter, mais vous devez réaliser une baisse de tension car la LED n'a pas besoin de 5 volts et accepte autant d'intensité que vous lui en donnez, jusqu'à éclater.

Il va donc falloir appliquer une petite loi d'ohm :

```
U=R*I
```

où U est la tension aux bornes de la résistance et I l'intensité la traversant. Prenons une diode LED qui a une chute de tension à ses bornes de 1.8 volts pour un courant de 10 mA. La carte Arduino fonctionne en 5 volts, donc

```
R = (5 - 1.8) / 0.01 = 320
```

soit 330 car les valeurs sont normalisées (c'est le même exemple que dans l'article d'Eric sur les premiers pas avec un microcontrôleur sans ta mère).

Vérifions la puissance dissipée (une résistance chauffe) :

```
P = U * I = 3.4*0.010 = 34 mW
```

largement inférieur au 1/4 de Watt des résistances carbones usuelles.

---

[1] entrées/sorties ou I/O en anglais