Fichiers

La conception du robot et la méthode pour le programmer impliquent toute une série de fichiers et de procédures d'installation. Cette section du site web donne les détails sur ces sujets. Rien de ce qui est présenté ici n'est nécessaire directement pour suivre le cours, mais bien des renseignements intéressants s'y trouvent pour ceux et celles qui désire aller plus loin avec leur robot. Les fichiers les plus importants se retrouvent aussi sur un site Github.

Conception du robot

Le robot est d'abord constitué d'un ensemble de pièces électroniques et mécaniques offertes sur le marché. Quelques pièces sont aussi fabriquées sur mesure. C'est le cas de la base du robot fait de plastique PVC de 6mm d'épaisseur. Un dessin en format DXF (format utilisé par Autocad et Qcad) donne les dimensions de la pièce. Les fichiers montrant les vues schématique et topologique (*layout*) des circuits imprimés du pont en H et de la carte mère pour le logiciel Eagle sont aussi fournis. Les fichiers «Gerber» pour la fabrication des circuits sont inclus également. Tous ces documents techniques sont sous licence GPL.

Modélisation en 3 dimensions du robot

M. Louis-Philippe Bourret a patiemment travaillé avec le logiciel Blender pour modéliser le robot en trois dimensions. Le rendu graphique est très joli. Un premier fichier montre le robot lui-même sans aucun éclairage ni animation alors qu'un second le montre dans le contexte d'une épreuve avec éclairage et animation. Nous remercions M. Bourret de sa collaboration.

Installation des outils AVR sous Linux

Pour programmer le robot, il faut une suite d'outils logiciels adaptés dont les plus importants sont le compilateur, l'assembleur et le logiciel de chargement par USB. Une librairie C, des utilitaires pour la manipulation des fichiers binaires et des documents d'utilisation complètent l'ensemble. Une manière pratique d'installer ces logiciels sur Fedora est d'utiliser l'utilitaire de gestion DNF avec la commande suivante:

```
% dnf list --all | grep -i avr
```

Ce qui permettra de lister ce qui est disponible. Un résultat semblable à ce qui suite devrait apparaître:

```
avr-binutils.x86_64  1:2.25-3.fc24    @fedora-everything
avr-gcc.x86_64       1:4.9.3-2.fc24   @fedora-everything
```

```
avr-gcc-c++.x86_64   1:4.9.3-2.fc24   @fedora-everything

avr-gdb.x86_64       7.1-13.fc24      @fedora-everything

avr-libc.noarch      1.8.0-12.fc24    @fedora-everything

avr-libc-doc.noarch  1.8.0-12.fc24    @fedora-everything

avrdude.x86_64       6.1-5.fc24       @fedora-everything

binutils-avr32-      2.26-8.fc24      @fedora-everything
linux-gnu.x86_64

texlive-             6.1.1-1.fc24     @fedora-everything
avremu.noarch

avra.x86_64          1.3.0-5.fc24     fedora

...                  ...              ...
```

Il suffira par la suite d'installer le tout avec une seule commande avec seulement les paquets d'intérêt:

```
% dnf install avr-binutils avr-gcc avr-gcc-c++ avr-gdb avr-libc
avr-libc-doc avrdude
```

Si vous utilisez une distribution de type Debian (comme Ubuntu), la commande sera différente, mais le résultat sera équivalent:

```
% sudo apt-get install avrdude avrdude-doc avr-libc binutils-avr
gcc-avr gdb-avr
```

Une autre méthode est de compiler et d'installer les outils à partir des fichiers sources. Il s'agit d'une méthode plus longue, plus complexe et qu'il faut exécuter méthodiquement.
Il faut télécharger les paquets binutils, gcc, avr-libc, la documentation avr-libc et avrdude. Il faudra aussi ajouter la librairie libusb et libusb-dev qui facilite l'accès au port USB sous Linux.
Une fois les paquets obtenus, et selon les numéros des versions les plus récentes, la liste des fichiers dans un répertoire de travail devrait ressembler à ce qui suit:

```
% ls
avrdude-5.3.1.tar.gz                avr-libc-user-manual-
1.4.5.tar.bz2
avr-libc-1.4.5.tar.bz2              binutils-2.14.tar.gz
avr-libc-user-manual-1.4.5.pdf.bz2  gcc-4.1.1.tar.bz2
```

La compilation de chaque paquet doit se faire dans l'ordre qui suit et en utilisant les commandes suivantes. L'installation doit se faire en étant «root» et on assume ici que les fichiers seront installés dans /usr/local/AVR:

```
% gtar zxvf binutils-2.14.tar.gz
% cd binutils-2.14
% ./configure --prefix=/usr/local/AVR --target=avr
% make
% make install
% PATH="$PATH:/usr/local/AVR/bin"
% cd ..

% gtar jxvf gcc-4.1.1.tar.bz2
% cd gcc-4.1.1
% ./configure --prefix=/usr/local/AVR \
        --target=avr --enable-languages="c,c++" \
        --disable-nls --disable-libssp
% make
% make install
% cd ..

% gtar jxvf avr-libc-1.4.5.tar.bz2
% cd cd avr-libc-1.4.5
% unset CC
% ./configure --build=`./config.guess` --host=avr --prefix=/usr/
local/AVR
% make
% make install
% cd ..

% bunzip2 avr-libc-user-manual-1.4.5.pdf.bz2
% cp avr-libc-user-manual-1.4.5.pdf /usr/local/AVR
% gtar jxvf avr-libc-user-manual-1.4.5.tar.bz2

% cp -r avr-libc-user-manual-1.4.5 /usr/local/AVR/

% gtar zxvf avrdude-5.3.1.tar.gz
% cd avrdude-5.3.1/etc/security/console.perms.d
% ./configure --prefix=/usr/local/AVR
% make
% make install
% cd ..
```

Configuration du port USB pour l'utilisation de la carte mère sur Fedora

Jean-Marc Chevalier, technicien au département de génie informatique et de génie logiciel, a mis au point une configuration pour le port USB pour les systèmes Linux utilisant l'interface udev. Deux fichiers de deux lignes chacun doivent être installés par l'usager “root” pour y arriver.

Le premier fichier doit être placé dans /etc/udev/rules.d/usbdev_atmel.rules et il doit avoir exactement les deux lignes suivantes (il ne faut par inscrire «Ligne1:» et «Ligne2:», car ces deux identificateurs sont placés uniquement pour bien marquer les débuts de lignes):

```
Ligne1:ATTR{idVendor}=="16c0", ATTR{idProduct}=="05dc",
SYMLINK+="vendor16c0-05dc", RUN+="/sbin/pam_console_apply"
Ligne2:SUBSYSTEM=="usb", KERNEL=="usbdev*", ATTR{idVendor}=="16c0",
ATTR{idProduct}=="05dc", RUN+="/sbin/pam_console_apply"
```

De plus, il faut ajouter un fichier /etc/security/console.perms.d/99-atmel.perms (ou avec tout autre numéro comme identificateur de début de fichier) ayant les deux lignes qui suivent comme contenu:

```
<vendor16c0-05dc>=/dev/vendor16c0-05dc
<console> 0600 <vendor16c0-05dc> 0660 root.root
```

Il faudra faire un «*logout*» avant de refaire un «*login*» pour activer les bonnes permissions. Par la suite, en branchant la carte dans un port USB, la commande «*dmesg*» devrait révéler que la connexion est bien établie. Une autre façon est de voir s'il y a bien une entrée dans /proc/bus/usb/devices pour la carte en regardant les numéros de *idVendor* et *idProduct* qui devraient correspondre à ceux de la carte mère. La commande «*lsusb*» peut donner aussi cette information.

À noter que cette configuration fait en sorte que la permission est accordée à n'importe quel appareil qui se branche sur le bus USB. L'usager ayant une session ouverte sur le poste de travail obtient les droits de lecture et d'écriture vers l'appareil en question

Nous remercions également Jean-Marc Chevalier pour sa contribution au développement du cours INF1900 et pour tout le travail qu'il fait sur les postes du laboratoire utilisé pour le cours.

Configuration du port USB pour l'utilisation de la carte mère sur Ubuntu

Les efforts de Jean-Marc ont été poursuivis par Philippe Proulx pour obtenir la configuration correcte sur Ubuntu. Il suffit d'éditer le fichier /etc/udev/rules.d/50-avr.rules pour que sa seule et unique ligne ressemble à ceci (en tant qu'usager “root”, bien entendu) :

```
SUBSYSTEM=="usb", ATTR{idVendor}=="16c0", ATTR{idProduct}=="05dc",
OWNER="votre_utilisateur", MODE="0666", SYMLINK+="inf1900-board"
```

Prenez soin de remplacer, dans cette dernière ligne, *votre_utilisateur* par votre nom d'utilisateur (vous pouvez l'obtenir grâce à la commande *whoami*). Par la suite, il suffit de redémarrer le udev avec la commande suivante:

```
% /etc/init.d/udev restart
```

Après avoir redémarré la carte mère connectée à l'ordinateur, il est toujours possible de voir si l'installation et la configuration fonctionnent bien avec la commande suivante (en tant qu'usager normal) :

```
% ls -la /dev/ | grep inf1900-board | wc -l
```

La commande suivante donne toujours aussi beaucoup d'information sous Linux:

```
% dmesg
```

La commande suivante peut aussi être très utile pour lister les périphériques USB connectés:

```
% lsusb
```

Merci à Philippe pour sa contribution.

Solution sous Windows

Pour y arriver sous Windows, l'installation de la solution est assez différente, car il faut installer en tout premier lieu les outils de compilation regroupés sous le nom de WinAVR. Par la suite, il faudra installer un pilote de périphérique (*driver*) spécifique pour Windows. Il faudra suivre ces deux sites pour arriver à une installation complète.

Et sur Mac ?

Aller à http://www.obdev.at/products/crosspack/index-de.html pour installer les outils. Par la suite, il est possible d'utiliser Xcode pour programmer. On recommande d'utiliser la compilation en ligne de commande comme on le fait sur Linux et de ne pas chercher à intégrer la compilation dans Xcode, car la configuration peut devenir rapidement compliquée et changée souvent selon les versions utilisées.

Le micrologiciel (firmware) sur la carte mère

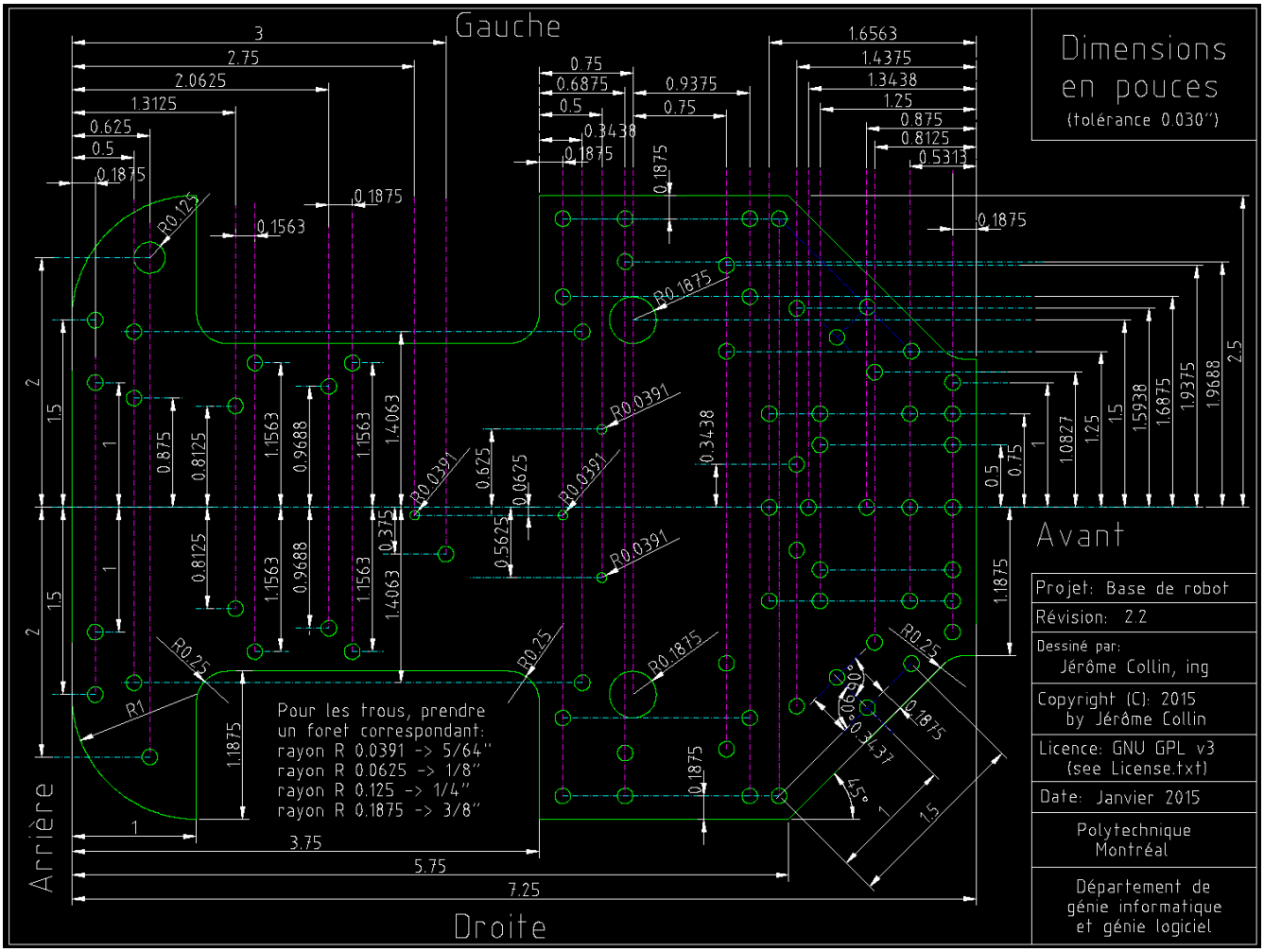
Le micrologiciel usbaspPoly est un programme qui doit être chargé sur le microcontrôleur ATMega8 sur la carte mère. Ce code très complexe gère les échanges par port USB avec le PC. Le micrologiciel a été développé par le groupe Objective Developpement (partie vusb) et par Thomas Fischl (partie USBasp), mais a été modifié par Matthew Khouzam, ancien étudiant en maîtrise à l'École Polytechnique de Montréal en génie informatique. Matthew a écrit un rapport qui accompagne le code. Une lecture du rapport, du code et du document "USB in a nutshell" peut permettre a une personne motivée de comprendre les détails du protocole USB sur la carte mère. Matthew Khouzam est un des plus grands contributeurs au projet de robot en INF1900. Une licence particulière accompagne ce logiciel.

Le chargement du micrologiciel sur une carte mère doit se faire en suivant à la lettre une suite d'opérations précises.

Programmes développés pour le cours

Une fois que la carte mère peut communiquer avec le PC par le port USB, il faut tout de même avoir un programme qui peut aller lire et écrire les données sur ce port. Lorsqu'on programme la carte en utilisant la commande "`make install`", l'utilitaire de chargement avrdude est appelé par une des règles du Makefile. Ce programme se charge d'ouvrir un canal de communication vers le port USB pour envoyer les données vers la carte. Par contre, lorsqu'on désire rediriger les données émises par RS232 vers le câble USB, il faut utiliser le programme serieViaUSB pour y arriver. Tout comme avrdude, serieViaUSB ouvre un canal de communication vers le port USB pour permettre les échanges de données. Il n'y a aucune licence associée à ce programme.

Le petit programme progmem utilisé surtout pour le travail pratique 9 est en réalité un compilateur écrit avec l'analyseur lexical lex (plus précisément sa variante GNU appelée flex) et l'analyseur syntaxique YACC (plus précisément sa variante GNU appelée bison). En bref, ces deux outils sont très utiles pour écrire des compilateurs. La compilation est étudiée en détail dans le cours LOG3210. Le code du programme progmem peut être utilisé librement et n'est distribué sous aucune forme particulière de licence.


Gauche
Avant
Droite
Arrière
Dimensions en pouces
(tolérance 0.030")
Pour les trous, prendre un foret correspondant:
rayon R 0.0391 -> 5/64"
rayon R 0.0625 -> 1/8"
rayon R 0.125 -> 1/4"
rayon R 0.1875 -> 3/8"
Projet: Base de robot
Révision: 2.2
Dessiné par: Jérôme Collin, ing
Copyright (C): 2015 by Jérôme Collin
Licence: GNU GPL v3 (see License.txt)
Date: Janvier 2015
Polytechnique Montréal
Département de génie informatique et génie logiciel