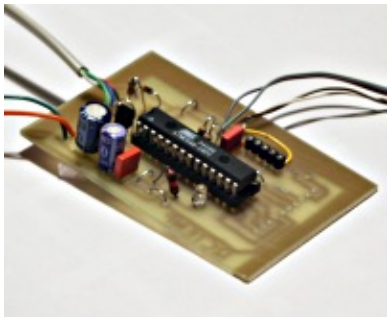
Un thermomètre numérique ou comment parler "I2C" à votre microcontrôleur Atmel

par Guido Socher (homepage)

L´auteur:

Guido aime Linux parce que c'est réellement un bon système pour développer votre propre matériel.

Traduit en Français par:
Jacques WLODARCZAK
<j.wlodarczak(at)tiscali.be>



Résumé:

Le microcontrôleur Atmega8 de chez Atmel possède de nombreuses lignes d'entrée/sortie. C'est le périphérique idéal pour développer n'importe quel équipement de mesure.

Dans cet article nous voyons comment interconnecter le microcontrôleur à un PC linux à travers une interface RS232 sans la puce MAX232.

Introduction

Un pre−requis pour cet article est que vous ayez l'environnement de programmation GCC AVR installé tel que décrit dans mon article "Programmer le microcontrôleur AVR avec GCC, libc 1.0.4". Si vous voulez éviter des problèmes d'installation vous pouvez évidemment utiliser le CD de programmation AVR de http://shop.tuxgraphics.org/

Quand vous utilisez ce genre d'appareil évolué comme une microcontrôleur, pour mesurer des signaux analogiques ou numériques, vous comptez évidemment pouvoir disposer d'interfaces pour recueillir les données ou envoyer des commandes au microcontrôleur. Dans tous les articles présentés ici dans le passé, nous utilisions toujours une communication RS232 avec l'UART qui est inclus dans le microcontrôleur. Le problème est que cela requérait en supplément une puce MAX232 et 4 condensateurs. Atmel suggère aussi qu'un oscillateur à quarz est nécessaire pour que la communication UART fonctionne de manière fiable. En tout cas ça fait beaucoup d'éléments supplémentaires ... et cela nous pouvons l'éviter !

La quantité de données à transférer entre le PC et le microcontrôleur est habituellement faible (juste quelques octets). Aussi la vitesse importe peu. Ce qui fait que le protocole/bus I2C est tout à fait indiqué pour remplir cette tâche.

I2C (prononcer à l'anglaise « eye−square−see ») est une interface de communication bidirectionnelle à deux conducteurs. Elle a été mise au point par Philips et ils ont protégé ce nom. C'est pourquoi d'autres fabricants utilisent un autre nom pour ce même protocole. Par exemple Atmel appelle I2C une « interface à deux conducteurs » (TWI : « two wire interface »).

Nombre d'entre vous ont sûrement déjà utilisé I2C sans le savoir. Toutes les cartes mères récentes contiennent un bus I2C pour lire les températures, les vitesses de ventilateur, des informations à propos de la mémoire disponible... bref toutes sortes d'informations sur le matériel. Ce bus I2C est malheureusement indisponible à l'extérieur du PC (il n'y pas de connecteur). C'est pourquoi nous devons en inventer un nouveau.

Mais voyons d'abord comment fonctionne « l'interface à deux conducteurs » (le TWI; rappelons qu'il s'agit d'une appellation alternative de I2C).

Comment fonctionne I2C/TWI.

Le cahier des spécifications de l'Atmega8 (voir les références) contient un descriptif très détaillé à partir de la page 160. Aussi, je vous en présenterai simplement un survol qui vous permettra de comprendre le descriptif dans le cahier des spécifications.

Sur le bus I2C vous avez toujours un dispositif maître et de nombreux dispositifs esclaves. Le maître est celui qui initie la communication et contrôle l'horloge. Les deux conducteurs de ce bus sont appelés SDA (ligne de données) et SCL (canal d'horloge). Chacun des dispositifs reliés au bus doit être alimenté séparément (tout comme pour une communication RS232 traditionnelle). Les deux conducteurs du bus sont normalement raccordés via une résistance de tirage au niveau logique « haut » de 4.7k (+5V pour les circuits imprimés à 5V). Cela donne une connection électrique en « OU » logique entre les dispositifs raccordés au bus. Un de ceux−ci amènera une ligne conductrice à la masse quand il voudra transmettre un 0, ou la laissera en position « haute » pour transmettre un 1. Le maître débute une communication en envoyant un motif appelé « état de départ » et alors s'adresse au dispositif avec lequel il veut communiquer. Chaque dispositif relié au bus possède une adresse unique sur 7 bits. Après cela le maître envoie un bit qui indique s'il veut lire ou écrire des données. L'esclave reconnaîtra alors qu'il a bien compris le maître en lui envoyant un « ack−bit » (bit d'acquittement). Ainsi nous avons maintenant localisé 9 bits de données sur le bus (soit 7 bits pour l'adressage + un bit de lecture + un « ack−bit »).

```
| départ | adresse 7-bit de l'esclave |  bit lecture de \
donnée | attente du bit de réception (ack-bit) | ... les données suivent ici
```

Et ensuite ?

Ensuite nous pouvons recevoir ou transmettre des données. Une donnée est toujours un multiple de 8 bits (un octet) et doit être acquittée (acknowledged) par un « ack−bit ». Autrement dit, nous relèverons toujours des suites de 9 bits sur le bus. Quand la communication est terminée alors le maître doit transmettre un « état de fin ». Aussi le maître doit−il savoir combien de données parviendront quand il les lira en provenance de l'esclave. Ce n'est cependant pas un problème du fait que vous pouvez transmettre cette information à l'intérieur du protocole d'utilisateur. Par exemple nous utiliserons l'octet 0 à la fin d'un mot pour indiquer qu'il n'y plus de donnée.

Les données sur le conducteur SDA est valide tant que le SCL est à 1. Comme ceci :

```
SDA H -\        /---\     /---\           /---\
    L   \-----/     \---/     \--------/     \------....

SCL H ----\     /-\       /-\     /-\    /-\    /-\
    L      \---/   \-----/   \---/   \--/   \--/   \-....


  | START |      1 |       1 |     0 |    1 |    0 |
```

Une des choses les plus intéressantes concernant ce protocole est que vous n'avez pas besoin d'un signal d'horloge précis et synchrone. Le protocole continue à fonctionner même s'il y a quelques parasites sur le signal d'horloge.

Précisément cette propriété rend possible l'implantation du protocole I2C dans une application en espace utilisateur sans avoir besoin d'un pilote du noyau ou d'un matériel spécial (comme un UART). Chouette non ?

La démarche

Comme dit auparavant nous ne pouvons utiliser le bus I2C interne du PC mais nous pouvons utiliser n'importe quelle interface externe où nous pourrons envoyer et recevoir des bits de données. Nous utiliserons simplement l'interface RS232 de notre PC. Autrement dit notre interface de communication est encore la même que dans les articles précédents mais nous épargnons le matériel MAX232, des capacités, etc.

La partie la plus rude est évidemment d'implanter le protocole I2C en partant de rien. Cela me prit 5 semaines pour le connaître et le déboguer mais maintenant c'est fait et vous n'avez qu'à le copier :−). J'espère que lorsque vous l'utiliserez, vous vous souviendrez de ce que ce code représente comme effort.

Comme application exemplaire, nous construirons un thermomètre. Vous pouvez évidemment mesurer n'importe quoi d'autre ou juste utiliser comme interrupteur d'une lumière. A vous de voir.

Dans un second article, nous ajouterons un affichage LCD local. En d'autres mots, vous aurez un thermomètre où vous lirez la température directement sur l'affichage et/ou sur votre PC linux. L'affichage viendra dans un second article afin de ne pas alourdir celui−ci.

Les NTC sont petit, bon marché et suffisamment précis

La sonde de température

Il est possible d'avoir des sondes de température calibrées d'origine (certaines parlent I2C ;−) mais elle sont plutôt chères. Les NTC sont meilleur marché et presque aussi bons, même sans calibration. Si vous les calibrez quelque peu, vous pouvez parvenir à une précision au−delà de la décimale.

Un problème posé par les NTC est qu'ils ne sont pas linéaires. Néanmoins ce n'est qu'une question de physique des semi−conducteurs pour trouver la bonne formule qui corrige la courbe non linéaire. Le microcontrôleur étant un petit ordinateur, les opérations mathématiques ne sont pas un problème. Les NTC sont des résistances dépendantes de la température. La valeur R du NTC pour une température donnée est :

$$R = R_N \cdot e^{B(\frac{1}{T} - \frac{1}{T_N})}$$

where

R_N = Value of NTC at T_N

$T_N = 25\ ^{o}/_{K} + 273K$

B = see datasheet of NTC

$T = T_C + 273K$

thus T_C *can be written as*

$$T_C = \frac{1}{\frac{1}{B}\ln(\frac{R}{R_N}) + \frac{1}{T_N + 273}} - 273$$

T ou Tc représente la valeur de la température que nous recherchons. Rn est la valeur de résistance du NTC à 25°c. Vous pouvez vous fournir des NTC à 4k7, 10k .... aussi Rn reprend cette valeur.

Le circuit

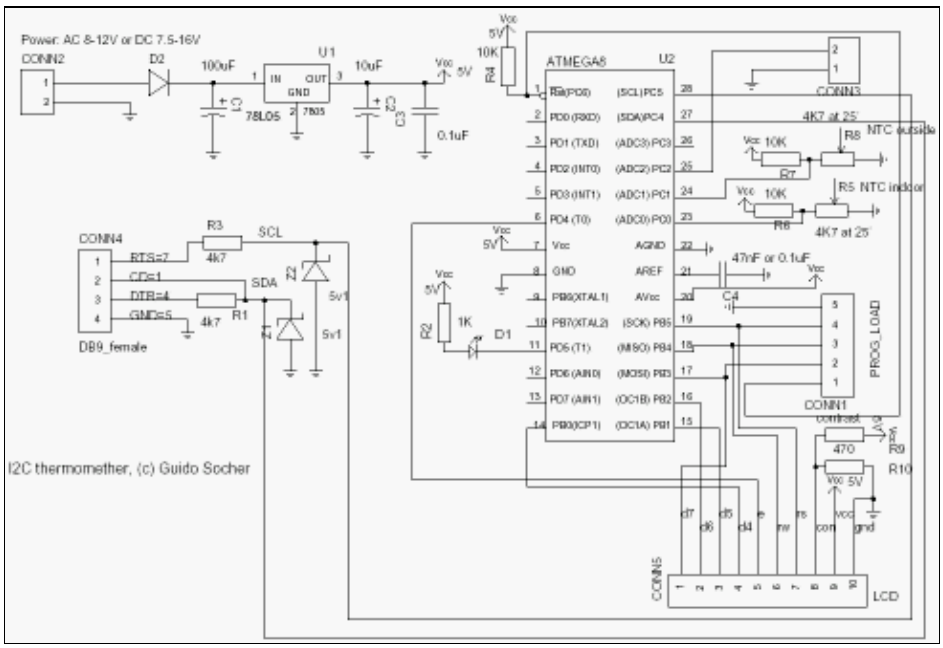


Diagramme du circuit. Cliquez sur le diagramme pour une vue plus détaillée

Maintenant, nous avons tout ce qu'il faut pour construire notre thermomètre numérique. Nous reprenons deux sondes NTC, une pour l'intérieur, l'autre pour l'extérieur. Vous pouvez en ajouter d'autres si vous voulez (conn3, la borne PC2 par exemple est libre). Dans le diagramme du circuit, j'y ai déjà repris les conducteurs nécessaires pour le raccordement à un affichage LCD car je n'ai pas envie de reconcevoir un nouveau circuit

complet pour le prochain article.

Il y a aussi une LED connectée. Ça coûte pas cher et c'est vraiment utile pour le débogage de base. Je l'ai utilisé par exemple pour déboguer l'état I2C de l'engin quand je programmais la communication I2C entre le PC et le microcontrôleur. Durant le fonctionnement courant, nous pouvons le laisser clignoter pour indiquer que des mesures sont en cours.

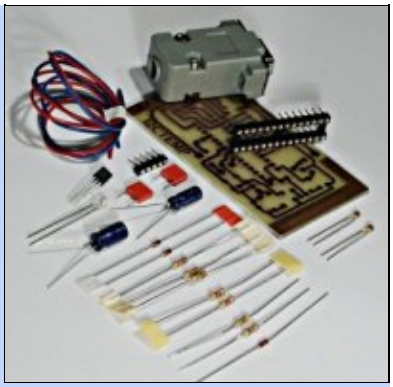
Pour le reste, le circuit est suffisamment explicite. Le convertisseur analogique−numérique dans le micro−contrôleur est utilisé pour mesurer la tension sur le NTC qui sera converti en valeurs de température.

L'Atmega8 présente deux options qui sont utilisées comme tension de référence pour le analogique−numérique. Il peut utiliser soit les 5V (AVcc) soit une référence interne de 2,56V. Pour la température intérieure, nous n'avons pas besoin d'une étendue de valeurs aussi large que pour le capteur extérieur. De +10°c à +40°c devrait normalement suffire. Aussi pouvons−nous utiliser la référence à 2,5V quand il s'agit de mesurer la sonde d'intérieur. Cela procure une très grande précision puisque les 1024 valeurs numériques possibles sont étalées sur seulement une portée de 0 à 2,56V ce qui nous donne une résolution de 2,5 mV (encore plus précis que sur la plupart des voltmètres numériques!).

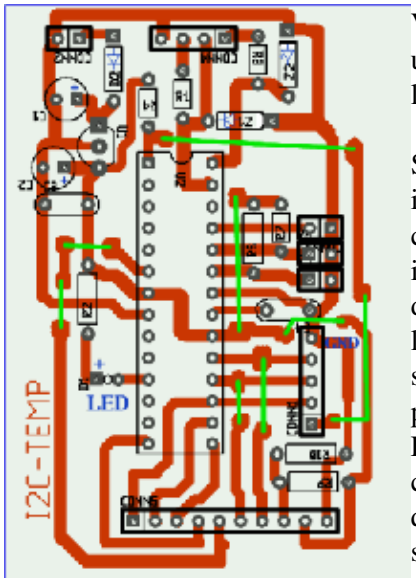
La borne CD sur la RS232 est une ligne d'entrée et est connectée au SDA du bus I2C. Nous l'utilisons pour lire les données venant du micro−contrôleur. DTR et RTS sont des lignes de sorties. Quand le PC envoie des bits de données sur le SDA alors il enclenche le DTR. Le maître I2C (ici le PC linux) contrôle la ligne SCL (horloge). Autrement dit la ligne d'horloge est une ligne de sortie sur la RS232.

Le 78L05 est utilisé pour générer une alimentation stable et une tension de référence. Vous pouvez utilisez quasi n'importe quel type d'alimentation AC ou DC entre 7,5V et 12V. 9V est un bon choix.

Fabrication du circuit imprimé



tuxgraphics.org vend tous les éléments requis dans cet article rassemblés avec un circuit imprimé de bonne facture.



Vous pouvez évidemment réutiliser le circuit imprimé prototype que nous utilisions dans notre article précédent. Il faut juste à reconnecter la LED sur la borne 11 et ajouter tout ce qui est nouveau.

Si vous voulez un circuit imprimé de belle apparence alors il est plus indiqué d'utiliser une nouvelle plaquette. Vu que le circuit est plus compliqué, il est évident qu'il faut graver proprement la plaquette du circuit imprimé. Après la lecture de l'article de Iznogood sur linuxfocus à propos de gEDA, j'ai décidé d'utiliser également gEDA au lieu de Eagle. gschem, l'outil de dessin de schémas, est très bon. Il n'a pas une librairie de symboles aussi étendue que chez Eagle et de ce fait j'ai dû créer le symbole pour l'Atmega8, mais c'est très facile d'utilisation et aussi bon que sous Eagle. Quelque peu plus problématique est pcb, l'outil pour dessiner les circuits imprimés. Quand vous venez d'Eagle, vous remarquez tout d'abord qu'il est possible de déconnecter les composants des « pistes ». Afin d'être sûr que les bonnes « pistes » soient connectées aux bonnes broches, vous devez lancer Connects–>Optimize rats–nest à l'occasion. Vous devriez d'abord compléter le diagramme du circuit et alors fabriquer le circuit imprimé. L'annotation entre les deux est seulement manuelle.

J'ai utilisé le calque de couleur orange pour le dessin. D'une manière ou d'une autre, les autres calques ne génèreront aucune sortie à l'impression. Le problème est que le calque coloré en orange est sur la face de la plaquette où se situent les éléments. Si vous écrivez le texte dans ce calque alors il sera en miroir quand vous l'imprimerez sur la plaquette matérielle. C'est pourquoi j'ai fait la maquette de base avec pcb et tout le reste avec gimp.

Grâce à shop.tuxgraphics.org, vous n'aurez pas à vous lancer dans des manipulations chimiques hasardeuses et courir la ville pour trouver les bons composants. Ils vendent tous les éléments requis par cet article. De cette manière, vous pourrez vous concentrer sur la partie la plus agréable et assembler ce circuit avec succès.

L'assemblage

Quand vous monterez le circuit soyez attentif aux éléments pour lesquels la polarité compte : les condensateurs chimiques, les diodes, les zeners, le 78L05, la LED et le microcontrôleur.

Avant d'insérer le microcontrôleur sur son support, vous devriez vérifier l'alimentation. Si elle est défectueuse, non seulement vous aurez des mesures de températures incorrectes mais vous risquez aussi de détruire le microcontrôleur. C'est pourquoi connectez l'alimentation externe (par exemple une pile de 9V) et vérifiez avec un voltmètre que vous avez exactement 5V sur la broche du support du microcontrôleur. A l'étape suivante vous connectez le circuit au port RS232 de votre PC linux et lancez le programme i2c_rs232_pintest avec diverses combinaisons de signaux.

```
i2c_rs232_pintest -d 1 -c 1
i2c_rs232_pintest -d 0 -c 1
i2c_rs232_pintest -d 1 -c 0
```

Ce programme met les bornes RTS aux valeurs de tension prévues (utilisé comme SCL, option –c) et DTRn (utilisé comme SDA, option –d) du port RS232. Le port RS232 a un niveau de voltage d'environ 10V. Derrière la diode zener, vous devriez mesurer seulement –0,7 pour une valeur binaire de 0 et + 4–5 V pour une valeur binaire de 1.

N'insérez le microcontrôleur qu'après que votre circuit ait passé tous ces tests.

Utiliser la communication I2C

Téléchargez (voyez les références) le fichier linuxI2Ctemp tar.gz et décompactez–le. La communication I2C est implantée dans deux fichiers :

```
i2c_avr.c -- the i2c statemachine for the atmega8
i2c_m.c   -- the complete i2c protocol on the linux side
```

J'ai donné à l'Atmega8 l'adresse d'esclave 3. Pour envoyer la chaîne « hello » à l'Atmega8, vous exécuterez les fonctions C suivantes :

```
address_slave(3,0); // renseigne l'esclave que nous allons envoyer quelquechose
i2c_tx_string("hello");
i2cstop(); // se dégager du bus I2C

du côté du microcontrôleur vous devriez recevoir le mot "hello" avec
i2c_get_received_data(rec_buf);
```

Très facile. Lire des données venant du microcontrôleur est similaire. Jeter un oeil sur le fichier i2ctemp_avr_main.c pour voir comment ça fonctionne quand les lectures de température sont effectuées.

Quelle température fait–il ?

Pour compiler et charger le code pour le microcontrôleur, lancez les commandes suivantes à partir du répertoire du paquet linuxI2Ctemp.

```
make
make load
```

Compilez les deux programmes i2c_rs232_pintest et i2ctemp_linux

```
make i2c_rs232_pintest
make i2ctemp_linux
```

ou simplement utilisez les versions précompilées se trouvant dans le répertoire « bin ».

Pour lire la températures lancez simplement:

```
i2ctemp_linux
```

... et cela affichera les températures d'intérieur et d'extérieur. Pour faire en sorte que ces données soient disponibles sur un site web, je suggère de ne pas lancer directement i2ctemp_linux du serveur car la communication I2C est très lente. Il est préférable de le lancer à partir d'une tâche cron et, de là, de l'écrire dans un fichier html. Un exemple de script est inclus dans le fichier README du paquet linuxI2Ctemp.

Conclusion

Le protocole I2C requiert très peu de matériel supplémentaire et est optimisé pour la transmission ou la réception de faibles quantités de données. C'est exactement ce qu'il nous faut si nous souhaitons communiquer avec notre microcontrôleur. C'est vraiment une très élégante solution !

Dans cet article, je me suis focalisé essentiellement sur la partie matérielle. Si vous aimez cet article alors j'en écrirai aussi un second dans lequel je décrirai comment le sofware fonctionne. Spécialement, comment faire la

conversion analogique−numérique et comment l'implantation du protocole I2C fonctionne. Pour ce prochain article nous pouvons aussi ajouter un affichage LCD et une conversion entre Farenheit et Celsius.

References

- **Page de téléchargement** pour cet article : the linuxI2Ctemp software, diagrams, software updates
- Comment programmer l'Atmega8 avec gcc : November2004 article 352
- Cahier de spécifications de l'Atmega8 : aller sur http://www.atmel.com/ et sélectionner products−>Microcontrollers −>AVR−8 bit RISC−>Documentation−>datasheets (local copy, pdf, 2479982 bytes)
- le magasin tuxgraphics. Vraiment un grand magasin en ligne :−): shop.tuxgraphics.org
  Sur ce site vous pourrez acquérir le CD de programmation AVR sous linux, touts les composants pour cet article, les affichages LCD et les microcontrôleurs.

2005−04−28, generated by lfparser_pdf version 2.51