Un thermomètre digital ou parler « I2C » à votre microcontroleur atmel –– deuxième partie

par Guido Socher (homepage)

L´auteur:

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

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

Résumé:

Dans cette seconde partie nous connecterons un afficheur et j'expliquerai comment fonctionne le logiciel.

Ceux qui découvrent cette série devraient d'abord lire la première partie (février 2005 – article 365) ; le même en français .

Les nouveautés

Dans l'article précédent / (traduction en français) nous avions déjà élaboré l'essentiel du matériel et des fonctionnalités afin de mesurer une température et en transmettre les données à un PC linux. Dans cet article nous ajouterons un affichage LCD et une interface très simple en gtk.

Ajouter ces deux choses est très facile à faire. Aussi, je consacrerai le reste de l'article à expliquer comment le logiciel I2C et le convertisseur analogique vers numérique fonctionnent.

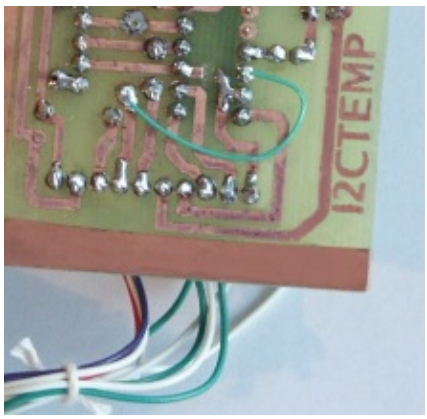
L'affichage LCD

Pour l'affichage LCD nous utiliserons un afficheur compatible HD44780 comme pour les articles précédents. Ces afficheurs sont très faciles à utiliser en combinaison avec des microcontrôleurs car vous pouvez leur envoyer des caractères ASCII.

Comme pour tous les articles de cette série vous pouvez vous procurer tous les matériaux nécessaires notamment l'afficheur LCD chez shop.tuxgraphics.org
J'utilise le même code du pilote LCD que dans les articles précédents. Les fichiers qui implémentent ce pilote LCD sont lcd.c, lcd.h et lcd_hw.h. Ils sont inclus dans le paquet que vous pouvez télécharger à la fin de cet article. L'interface pour ce code est vraiment facile à utiliser :

```
// call this once:
// initialize LCD display, cursor off
lcd_init(LCD_DISP_ON);

// to write some text we first clear
// the display:
lcd_clrscr();
lcd_puts("Hello");
// go to the second line:
lcd_gotoxy(0,1);
lcd_puts("LCD");
```



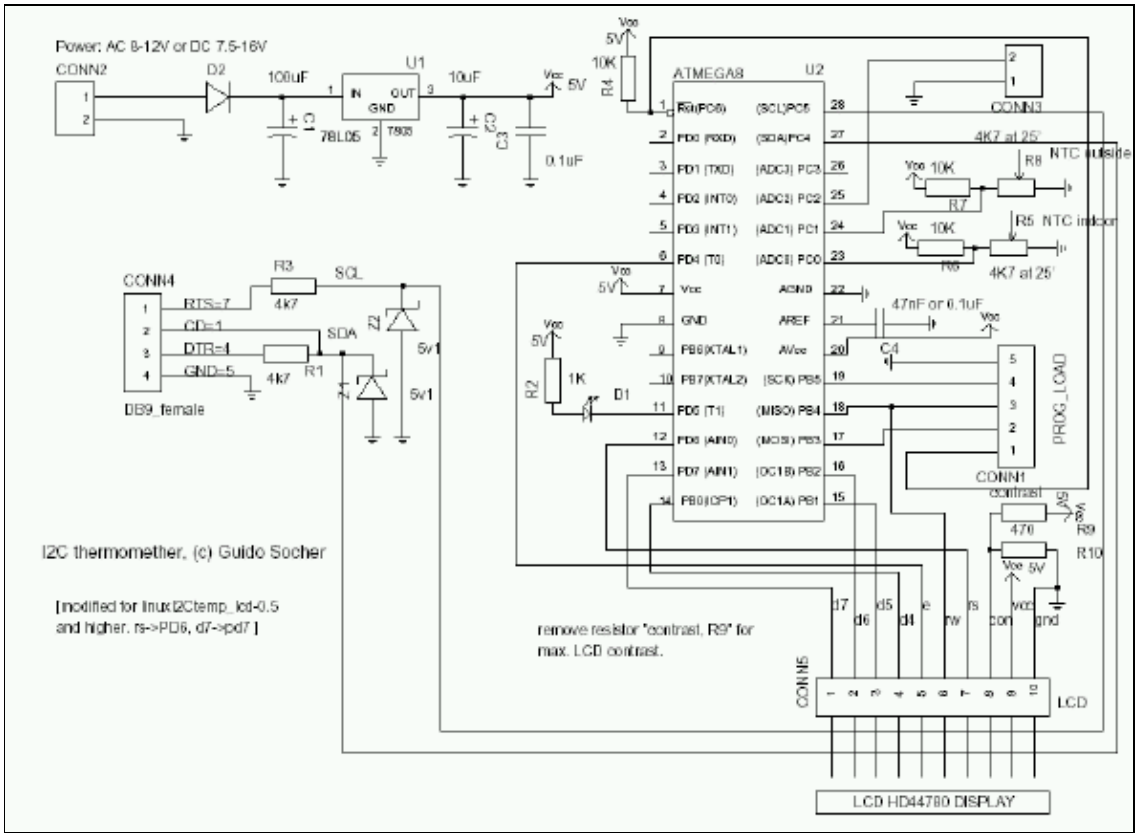
La manière dont fonctionnent ces afficheurs HD44780 est décrite sur LinuxFocus dans l'article de septembre 2002 Comprendre l'affichage LCD HD44780

Le logiciel est écrit de sorte qu'il fonctionne tant avec les afficheurs LCD 16x2 que 20x2.

Il y a aussi une actualisation du diagramme du circuit. J'ai découvert que certains afficheurs LCD ont une charge capacitive plus élevée et une résistance moins élevée que d'autres. Probablement parce que ils ont une meilleure protection ESD. Cette charge additionelle peut éventuellement produire des erreurs de bit durant la programmation « In circuit » quand l'afficheur LCD est connecté aux bornes SCK et MOSI.

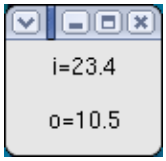
Dans un premier temps, j'ai essayé de connecter des résistances supplémentaires dans les lignes vers l'afficheur LCD. Chez moi ça marchait, mais certains, particulièrement parmi les utilisateurs de laptop, continuaient à connaître des problèmes

Pour éviter tous ces problèmes, j'ai actualisé le diagramme du circuit et ainsi les bornes D7 et RS de l'afficheur LCD sont maintenant connectées aux bornes PD7 et PD6. Et ce n'est pas un problème de changer ça, même si vous avez déjà fait la plaquette : ajoutez juste un petit fil sous la plaquette et coupez la connection vers PB3 avec une lame.



Une petite interface graphique

Pour ceux qui auraient souhaité une interface graphique sur leur bureau, j'en ai fait une très simple. qui consiste en deux étiquettes qui affichent la sortie sur deux lignes de la commande i2ctemp_linux (i2ctemp_linux est la commande qui lit les températeurs venant du circuit via l'i2c) :



Maintenant nous avons un chouette thermomètre avec nombre de possibilités :

- Vous pouvez lire la température sur place grâce à l'afficheur
- Vous pouvez avoir une petite interface graphique sur votre bureau
- Vous pouvez écrire des valeurs, à l'aide d'une tâche cron, dans un fichier, pour obtenir des statistiques à long terme

Je profite du reste de cet article pour expliquer quelque peu les méandres de ce logiciel.

Comment fonctionne une conversion de l'analogique vers du numérique

L'Atmega8 supporte deux modes. Un où il mesure en permanence les signaux analogiques et déclenche une interruption quand la mesure est prête. Le logiciel de l'application peut alors utiliser cette interruption pour copier aussitôt le résultat venant de deux registres vers une variable.

L'autre mode est appelé mode « par à coups » (single shot). Ici une seule conversion est effectuée, Le mode par à coups est en plus assez rapide. En tenant compte du temps d'initialisation préléminaire des registres et de l'extraction vous pouvez obtenir 100 conversions par seconde. C'est plus que suffisamment rapide pour nous. Aussi utiliserons−nous ce mode.

Sur notre Atmega8 nous pouvons utiliser les bornes d'entrées analogiques ADC0 jusque ADC3. En plus de cela il y a les bornes AGND (masse analogique connectée à la masse normale), AREF (la tension de référence) et AVCC (connecté au +5V).

Durant la conversion analogique vers numérique, le signal analogique est comparé avec l'AREF. Un signal analogique égal à l'AREF correspond à une valeur numérique de 1023. l'AREF peut être n'importe quelle référence externe entre 0 et 5V. Sans l'utilisation d'une référence externe vous pouvez encore opérer une conversion précise par l'usage ou d'une référence interne (2,56V) ou d'AVCC. Ce qui est utilisé est déterminé dans le logiciel par les bits REFS0 et REFS1 dans le registre ADMUX.

Le convertisseur analogique vers numérique peut convertir une des lignes d'entrée ADC0−ADC3 à la fois. Avant de lancer la conversion vous devez placer des bits dans le registre ADMUX pour indiquer à la puce quel canal utiliser.

Une simple conversion analogique vers numérique ressemble donc à ceci :

```
volatile static int analog_result;
volatile static unsigned char analog_busy;

analog_busy=1; // busy mark the ADC function
channel=0; // measure ADC0
// use internal 2.56V ref
outp((1<<REFS1)|(1<<REFS0)|(channel & 0x07),ADMUX);
outp((1<<ADEN)|(1<<ADIE)|(1<<ADIF)|(1<<ADPS2),ADCSR);
sbi(ADCSR,ADSC); // start conversion
```

Maintenant le microcontrôleur effectuera la conversion analogique vers numérique et appelera la fonction SIGNAL(SIG_ADC) une fois prêt. Dans cette fonction nous pouvons copier le résultat vers une variable. Comme programmeur vous devez veiller à lire les 8 bits de faible poids d'abord car le microcontrôleur a un mécanisme de verrouillage pour simuler une lecture « atomique ».

```
SIGNAL(SIG_ADC) {
        unsigned char adlow,adhigh;
        adlow=inp(ADCL); /* read low first, two lines. Do not combine
                            the two lines into one C statement */
        adhigh=inp(ADCH);
        analog_result=(adhigh<<8)|(adlow & 0xFF);
        analog_busy=0;
}
```

Après cela nous avons le résultat de la conversion « analogique vers numérique » disponible sous forme de nombre dans la variable « analog_result ». Ceci peut être utilisé ailleurs dans le programme. Très facilement.

Comme pour toute interruption vous devez appeler sei(); pour les autoriser globalement. Ce qui devrait ête fait quelque part dans le programme principal (ce n'est pas montré ci−dessus).
Il y a quelques « drapeaux » que je vais vous expliquer brièvement :

- ADEN: activer le convertisseur analogique vers numérique, à activer avant de positionner ADSC.
- ADIE : autoriser l'interruption ADC (=autoriser l'appel de SIGNAL(SIG_ADC))
- ADIF : ADC Interrupt Flag (doit être positionné sur 1 avant la conversion)
- ADPS : ADC clock pre−scaler bits : doivent être fixés de telle sorte que la fréquence d'horloge divisé par le facteur « pre−scale » soit une valeur entre 50 et 200 KHz. Le facteur de division est égal à 2^ADPS (2 élevé à la puissance de la valeur des bits ADPS). Les valeurs ci−dessus (ADPS2=1, ADPS1=0, ADPS0=0 = decimal 4 −> 2^4 = 16 −> division factor = 16) conviennent pour une fréquence d'horlogue de 1MHz.

L'Atmega8 offrent de nombreuses possibilités de sélection de la tension de référence. Ce dernier est comparé à notre tension d'entrée analogique. C'est la tension qui correspond à une valeur numérique de 1023.

REFS0=0, REFS1=0	utilise AREF externe, Vref interne off
REFS0=0, REFS1=1	AVCC avec condensateur externe optionel sur la broche AREF
REFS0=1, REFS1=1	Tension de Référence Interne de 2.56V avec condensateur externe (optionel) sur la broche AREF

Une capacité optionnelle sur la borne AREF peut être utilisée pour supprimer le bruit (noise) et stabiliser la tension AREF.

Comment fonctionne la communication I2C : du côté de l'Atmega8

J'ai déjà expliqué dans la partie 1 (February2005 article365) comment le protocole I2C fonctionne. Jetons maitenant un coup d'oeil sur le logiciel. L'Atmega8 possède un support matériel pour une communication I2C. Aussi vous n'aurez pas besoin d'implémenter ce protocole. Par contre vous devrez implémenter un mécanisme d'état (state machine) qui dira à l'Atmega8 quoi faire ensuite. Voici un exemple :

On reçoit un paquet I2C avec l'adresse de notre propre esclave. L'Atmega8 fera alors appel à la fonction SIGNAL(SIG_2WIRE_SERIAL) avec le code d'état 0x60 (pour d'autres évévements nous aurions eu d'autres codes).

−−> Nous devons maintenant fixer un nombre en registre pour dire à l'Atmega8 quoi faire ensuite. Dans ce cas nous lui dirons : recevoir la partie « données » et en accuser réception.

Quand les données sont reçues nous serons appelés par le code de statut 0x80.

−−> Maintenant nous lisons les octets de données et disons à l'Atmega8 de réceptionner les prochains le cas

échéant.

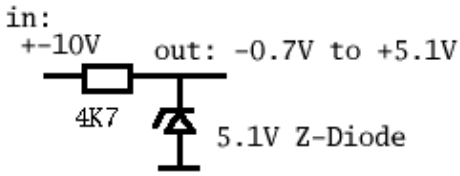
Quand la communication est terminée nous avons un code de statut 0xA0 (stop condition) et disons à notre application qu'un message complet à été reçu.

L'entièreté du mécanisme d'état pour le mode esclave I2C et tous les états possibles, sont expliqués dans le livre de spécifications page 183 (voir le lien dans la section « références » à la fin de l'article).

La transmission des données est tout à fait similaire, jetez un oeil sur le code !

Comment fonctionne la communication I2C : du côté de Linux

D'abord un mot à propos du matériel. Même si I2c est un bus nous n'utiliserons qu'une connection poste à poste entre un esclave et le PC Linux comme maître I2C. Nous pouvons ainsi épargner la résistance de pull−up du fait que l'esclave peut faire chuter la ligne sans créer de court circuit. Nous mettrons simplement une résistance de 4.7 K dans ce circuit.



La tension doit être ajustée : c'est effectué grâce à une diode Zener qui limite les tensions négatives à −0.7V et les positifs à maximum +5.1.

Après une étude approfondie des entrailles de l'Atmeag8, j'en vins finalement à la conclusion que la protection interne de l'Atmeag8 lors de l'entrée est probablement suffisante car les courants qui traversent la résistance de 4.7 K sont très faibles. Nous n'aurons pas besoins d'une diode Zener. Ceci dit, ça ne peut faire de tort d'en avoir une.

Le logiciel I2C de Linux implémente de manière basique une pile I2C complète. Je souhaite utiliser un petit utilitaire de ligne de commande qui ne nécessite pas de bibliothèque spéciale ou un module du noyau. Cela fonctionnera par lui−même.

Si vous fouillez dans le fichier i2c_m.c (voir le téléchargement) vous pouvez constater que chaque message I2C est construit bit par bit.

Pour générer les « bits » nous devons brancher les bornes physiques sur l'interface rs232. C'est fait avec les appels ioctl :

```
// set RTS pin:
int arg=TIOCM_RTS;
ioctl(fd, TIOCMBIS, &arg);
```

... ou pour produire un zéro :

```
// clear RTS pin:
int arg=TIOCM_RTS;
ioctl(fd, TIOCMBIC, &arg);
```

Si vous voulez porter cette pile vers un autre OS, vous n'avez qu'à changer ces lignes, le reste est du pur C.

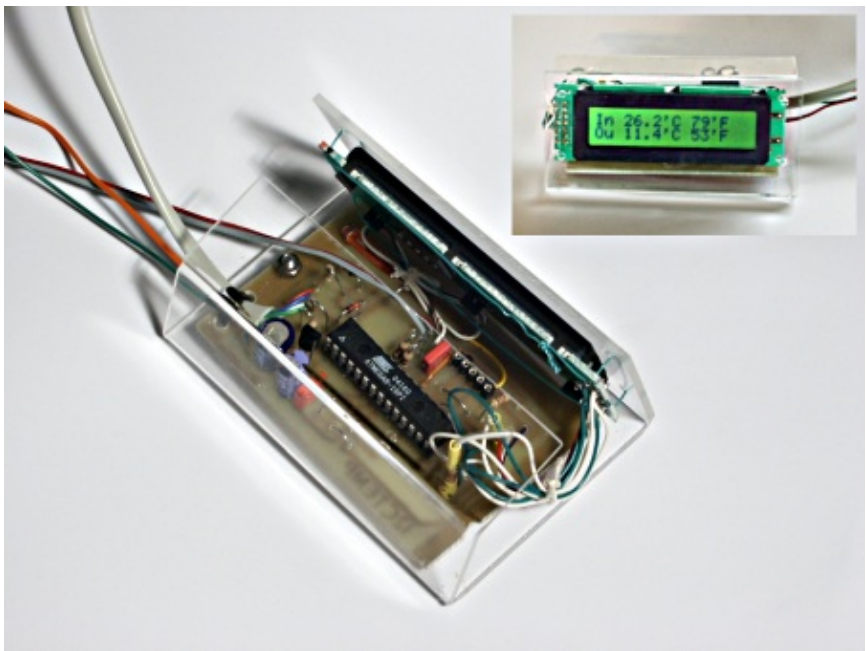
USB vers RS232

Pour les laptops qui n'ont plus à ce jour d'interface rs232 pour pouvez simplement utiliser l'adaptateur USB/rs232. J'utilise un adaptateur sans marque qui contient une puce Prolific 2303. L'adaptateur en question apparaît comme suit dans le fichier /proc/bus/usb/devices : Vendor=067b ProdID=2303 Rev= 2.02. Voyez aussi "Use your ATEN UC−232A USB adapter with Linux (Linuxfocus, November 2001, article 223)".

Conclusion

J'utilise maintenant le thermomètre depuis deux mois et je l'apprécie vraiment car on peut lire directement sur l'afficheur et vous avez la possibilité de stocker les données sur votre PC. Vous pouvez les lire, dessiner des graphiques et faire des statistiques. Chouette !

Le capteur extérieur doit être protégé de la pluie (et du soleil). Vous pouvez essayer de l'envelopper dans du plastique mais je ne le recommande pas. Quelque soit la manière dont l'aurez attaché, la pluie finira par y pénétrer et stagner. Le NTC est très robuste et il importe peu qu'il soit un tantinet humide pourvu qu'il puisse sécher. Utiliser un tube de comprimés retourné que vous laissez ouvert à la base. De cette façon l'eau pourra sortir.



Vous pouvez de nouveau commander toutes les pièces (LCD display, PCB, microcontrôleur, ...) au magasin en ligne tuxgraphics : shop.tuxgraphics.org.
Bon amusement !

Références

- **Télécharger** la page de cet article (en anglais) : the linuxI2Ctemp_lcd software, diagrams, software updates
- Comment programmer l'atmega8 avec gcc: November2004 article 352
- Spécifictions de l'Atmega8: sur http://www.atmel.com/ et sélectionner products−>Microcontrollers −>AVR−8 bit RISC−>Documentation−>datasheets
  (local copy, pdf, 2479982 bytes)

Site Web maintenu par l´équipe d´édition LinuxFocus © Guido Socher "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Translation information: en −−> −− : Guido Socher (homepage) en −−> fr: Jacques WLODARCZAK <j.wlodarczak(at)tiscali.be>

2005−09−15, generated by lfparser_pdf version 2.51