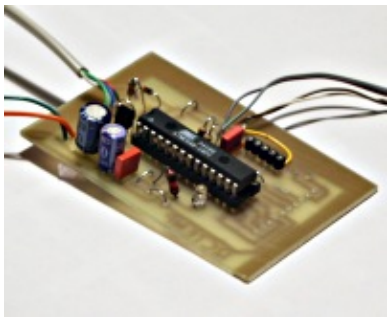
Un Termómetro Digital o Comunicarte en I2C Con Tu Microcontrolador Atmel

por Guido Socher (homepage)

Sobre el autor:

A Guido le gusta Linux porque es realmente un buen sistema para desarrollar tu propio hardware.

Traducido al español por:
Gabriela González
<tradugag(at)yahoo.com>



Resumen:

El microcontrolador Atmega8 de Atmel presenta muchos circuitos digitales y análogos de entrada/salida. Es el dispositivo ideal para desarrollar cualquier clase de equipo de medición.

En este artículo vemos cómo interconectar el microcontrolador a un ordenador Linux sobre una interfaz física RS232 sin el chip extra MAX232.

Introducción

Un prerequisito para este artículo es que tengas instalado el entorno de programación GCC AVR como se describió en mi artículo ("Programando el microcontrolador AVR con GCC, libc 1.0.4") "Programming the AVR microcontroller with GCC, libc 1.0.4". Si quieres evitar dificultades con la instalación, por supuesto, puedes usar el CD de progamación AVR de http://shop.tuxgraphics.org/

Cuando utilizas un dispositivo tan avanzado como un microcontrolador para medir señales análogas o digitales, por supuesto, quieres interfaces para evaluar los datos o enviar comandos al microcontrolador. En todos los artículos presentados aquí anteriormente siempre utilizamos comunicación rs232 con el UART (Receptor/Transmisor Universal Asíncrono)que se encuentra incluido en el microcontrolador. El problema es que esto requiere un chip MAX232 adicional y 4 condensadores extra. Atmel también sugiere la requisición de un oscilador de cristal externo para que la comunicación UART funcione de modo confiable. En todo caso, se trata de muchas partes extra... ¡y podemos evitarlo!

La cantidad de datos a transferir entre el ordenador y el microcontrolador es generalmente muy pequeña (sólo unos pocos bytes). La velocidad, por lo tanto, no nos preocupa. Esto hace que el bus/protocolo I2C sea apropiado para esta tarea.

I2C (pronunciación "eye−square−see" o "I cuadrado C" en español) es una interfaz de comunicación bidireccional de dos cables. Fue inventada por Philips y el nombre se encuentra protegido. Esta es la razón por la cual otros fabricantes utilizan otro nombre para el mismo protocolo. Atmel lo llama I2C "two wire interface" (TWI) o "interfaz de dos cables".

Muchos de ustedes podrían ya estar utilizando I2C en sus ordenadores sin saberlo. Todas las placas madres modernas tienen un bus I2C para leer temperaturas, la velocidad del ventilador, información acerca de la memoria disponible... toda clase de información sobre hardware. Este bus I2C lamentablemente no se encuentra en la parte externa del ordenador (no hay una conexión física). Por lo tanto, tendremos que inventar algo nuevo.

Pero primero veamos cómo la "interfaz de dos cables" (=TWI = un nombre alternativo para I2C) funciona.

Cómo Funciona I2C/TWI

La hoja de datos de Atmega8 (ver referencias) ya cuenta con una descripción muy detallada que comienza en la página 160. Por lo tanto yo sólo presentaré un resumen aquí. Después de este resumen, tú podrás

comprender la descripción en la hoja de datos.

En el bus I2C siempre tienes un dispositivo maestro y uno o varios dispositivos esclavo. El maestro es el dispositivo que inicia la comunicación y controla el reloj. Los dos cables de este bus se llaman SDA (línea de datos) y SCL (línea de reloj). Cada uno de los dispositivos del bus deben recibir energía en forma independiente (lo mismo que sucede con la comunicación tradicional rs232). Las dos líneas del bus se encuentran normalmente conectadas vía resistencias ascendentes 4.7K pullup a, logicamente, "High" ("Alto") (+5V por 5V ICs). Esto da una conexión eléctrica "or" ("o") entre todos los dispositivos. Un dispositivo simplemente extrae una línea a GND cuando quiere transmitir un 0 o lo deja en "Alto" cuando envía un 1.

El maestro comienza una comunicación enviando un patrón llamado "start condition" ("condición de inicio") y luego se dirige al dispositivo con el que quiere hablar. Cada dispositivo del bus tiene una dirección 7 bit única. Luego de esto, el maestro envía un bit que indica si quiere leer o escribir datos. El esclavo ahora reconocerá que ha comprendido al maestro enviando un bit de reconocimiento. En otras palabras, ahora hemos visto 9 bits de datos en el bus (7 bits dirigidos + bit de lectura + bit de reconocimiento):

```
| start | 7-bit slave adr | read_data bit | wait for ack | ... data comes here
```

¿Qué Sigue Ahora?

Ahora podemos recibir o transmitir datos. Los datos son siempre un múltiplo de 8 bits (1 byte) y deben ser reconocidos por un bit de reconocimiento. En otras palabras, siempre veremos paquetes de 9−bit en el bus. Cuando la comunicación ha finalizado, el maestro debe transmitir una "stop condition" ("condición de finalización"). En otras palabras, el maestro debe saber cúantos datos vendrán cuando lea datos de un esclavo. No obstante, esto no es un problema dado que puedes transmitir esta información dentro del protocolo del usuario. Usaremos, por ejemplo, el byte cero al final de una cadena para indicar que no hay más datos.

Los datos en el cable SDA son válidos mientras que el SCL es 1. Resulta así:

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

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


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

Una de las mayores ventajas de este protocolo es que no necesitas una señal de reloj precisa y síncrona. El protocolo aún funciona cuando hay un pequeño soplido de un magnetrón en la señal del reloj.

Exactamente esta propiedad es la que hace posible implementar el protocolo I2C en una aplicación de espacio de un usuario sin necesidad de un controlador del núcleo o hardware especial (como un UART). Fantástico, ¿no?

El Plano

Como se comentó anteriormente, no podemos utilizar el bus I2C interno del ordenador pero podemos usar cualquier interfaz externa donde podamos enviar y recibir datos bits individuales. Simplemente utilizaremos la interfaz de hardware RS232 de nuestro ordenador. En otras palabras, nuestra interfaz de comunicación sigue siendo la misma de artículos anteriores pero ahorramos el hardware MAX232, condensadores, etc...

La parte difícil es, por supuesto, implementar el protocolo I2C del scratch. Me llevó 5 semanas aprenderlo y

depurarlo pero ahora ya está hecho y tú simplemente puedes copiarlo :–). Espero que recuerdes el valor de este código cuando lo uses.

Como ejemplo de aplicación vamos a construir un termómetro. Puedes, por supuesto, medir algo más o simplemente prender y apagar luces. Depende de tí.

En un segundo artículo, vamos a agregar una pantalla local LCD. En otras palabras, tendrás un termómetro en donde podrás leer la temperatura directamente de la pantalla y/o leerla con tu ordenador Linux. La pantalla la veremos en un segundo artículo a fin de no sobrecargar éste.

NTCs son pequeños, económicos y lo suficientemente apropiados

El Sensor de la Temperatura

Ya es posible conseguir sensores calibrados de temperatura (algunos ya se comunican con I2C ;–) pero son bastante caros. Los NTCs son más baratos y casi tan buenos aún sin calibración individual. Si los calibras un poco, entonces es posible alcanzar la precisión detrás del punto decimal.

Un problema con los NTCs es que son no lineales. Sin embargo, esto es solamente un tema de física semiconductora encontrar la fórmula correcta para corregir la curva no lineal. El microcontrolador es una pequeña computadora por lo cual las operaciones matemáticas no son un problema. Los NTCs son resistencias que dependen de la temperatura. El valor R del NTC a una temperatura dada es:

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

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 o Tc es el valor de temperatura que estamos buscando. Rn es el valor de resistencia del NTC a 25'C. Puedes comprar NTCs 4k7, 10K, ... entonces Rn tendrá este valor.

El Circuito

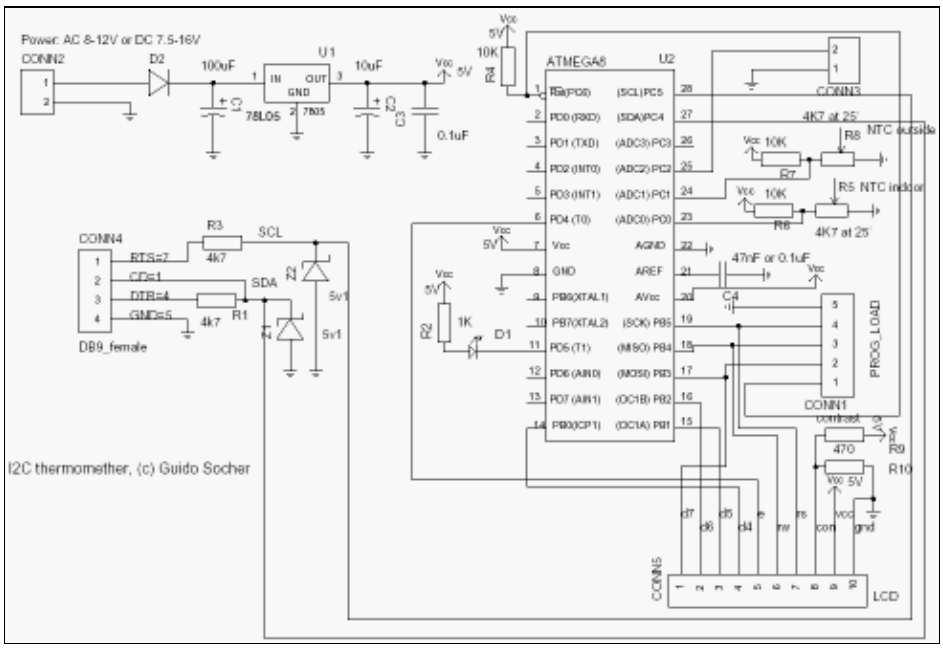


Diagrama del circuito. Haz click en el diagrama para obtener una vista detallada.

Ahora tenemos todo lo que necesitamos para construir un termómetro digital. Agregamos dos sensores NTC, uno para la temperatura interior y otro para la exterior. Si quieres, puedes agregar más (conn3, pin PC2 por ejemplo, es gratuito). En el diagrama de circuito yo ya agrego los cables necesarios para la conexión de una pantalla LCD porque no quiero que construyas un circuito nuevo completo para el próximo artículo.

También hay conectado un LED (diodo emisor de luz). No cuesta mucho es realmente útil para una depuración básica. Yo lo usé, por ejemplo, para depurar la máquina de estado I2C cuando desarrollé la comunicación I2C entre el ordenador y el microcontrolador. Durante la operación normal, simplemente podemos dejarlo parpadeando para indicar que las mediciones son tomadas.

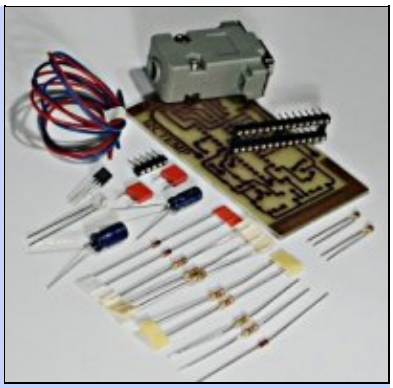
De otro modo el circuito es directo. El análogo al adaptador digital en el microcontrolador se utiliza para medir el voltage en el NTC el que entonces será convertido en un valor de temperatura.

El Atmega8 tiene dos opciones en lo que se utiliza como referencia de voltage para el análogo al adaptador digital. Puede utilizar tanto el 5V (AVcc) como una referencia interna 2.56V. Para las temperaturas internas no necesitaremos un rango de temperatura que sea tan grande como para el sensor externo. +10'C a +40'C normalmente debería ser suficiente. Por lo tanto podemos usar la referencia 2.56V cuando midamos el sensor interno. Esto proporciona una alta precisión ya que los 1024 valores digitales posibles sólo se expanden sobre 0−2.56V con lo que obtenemos una resolución de 2.5mV (¡más precisa que en la mayoría de los voltímetros digitales!).

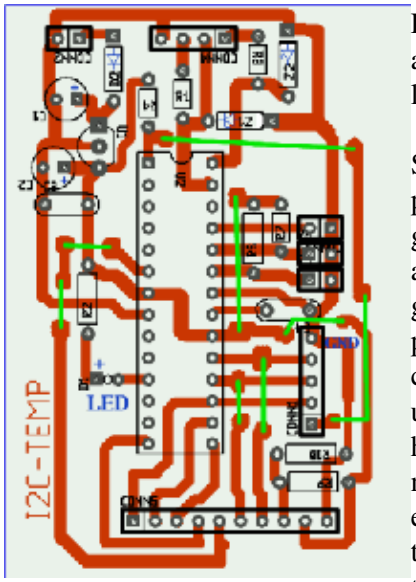
El CD−pin en RS232 es una línea de entrada y se encuentra conectada a SDA en el bus I2C. Lo utilizamos para leer datos del microcontrolador. DTR y RTS son líneas de salida. Cuando el ordenador coloca datos bits en la línea SDA entonces simplemente alterna DTR. El maestro I2C (aquí el ordenador Linux) controla la línea SCL (reloj). En otras palabras, la línea reloj es una línea de salida en el rs232.

El 78L05 se utiliza para generar un suministro estable de energía y referencia de voltage. Puedes utilizar casi cualquier tipo de suministro de energía AC o DC entre 7.5V y 12V. 9V es una buena elección.

Construyendo la Placa



tuxgraphics.org vende todas las piezas que se necesitan para este artículo junto con una placa grabada apropiada.



Por supuesto, puedes reutilizar la placa de prototipo que usamos en el artículo anterior. Simplemente reconecta el LED al pin 11 y agrega todas las cosas nuevas.

Si deseas tener un circuito que luzca bien entonces sería mejor usar una placa nueva. Dado que el circuito es mucho más complicado, conviene grabar apropiadamente una placa de circuito impreso. Después de leer el artículo de Linuxfocus de Iznogood sobre gEDA yo también decidí usar gEDA en lugar de Eagle. gschem, la herramienta de dibujo esquemática para gEDA es muy buena. No tiene una biblioteca de símbolos tan grande como Eagle y tuve que crear el símbolo para Atmega8 pero es muy fácil de usar y es tan buena como Eagle. Un poco problemática es pcb, la herramienta para dibujar PCBs. Cuando has utilizado Eagle, primero notarás que es imposible desconectar las partes de las bandas elásticas. Para estar seguro de que la banda elástica correcta está conectada al pin correcto, tienes que ejecutar Connects−>Optimize rats−nest de una vez. Primero tendrás que completar el diagrama de circuito y luego construir la placa. La anotación entre ambos es solamente manual.

Yo usé la capa pintada de naranja para dibujar. De algún modo, las otras capas no generan ninguna salida al imprimir. El problema es que la capa pintada de de color naranja ya se encuentra del lado de la placa en donde están las partes. Si escribes texto en esta capa, tendrá que ser duplicado cuando lo imprimas en la placa física. Por lo tanto hice el diseño básico con pcb y todo el resto con gimp.

Gracias a shop.tuxgraphics.org no tendrás que trabajar con químicos peligrosos ni recorrer la ciudad para encontrar los componentes correctos. Ellos venden todas las partes necesarias para este artículo. De este modo, sólo tendrás que concentrarte en la parte divertida y ensamblar este circuito con éxito.

Colocando Todo Junto

Cuando ensamblas el circuito, presta atención a las piezas en las que la polaridad es importante: Electrocondensadores, diodo, diodos Z, 78L05, LED y el microcontrolador.

Antes de colocar el microcontroladore en el casquillo, debes verificar la parte de suministro de energía. Si esta no funciona, tú no sólo obtendrás lecturas de temperatura incorrectas sino que también podrías llegar a destruir el microcontrolador. Por lo tanto, conecta energía exterior (por ejemplo, una batería 9V) y verifica con un voltímetro que tienes exactamente 5V en el pin del casquillo del microcontrolador. Como siguiente paso, conecta el circuito al puerto rs232 de tu ordenador Linux y ejecuta el programa i2c_rs232_pintest con varias combinaciones de señales.

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

Este programa establece los niveles de voltage en los pins RTS (usado como SCL, opción −c) y DTR (usado como SDA, opción −d) del puerto rs232. El puerto rs232 tiene niveles de voltage de aproximadamente +/− 10V. Detrás del diodo Z deberías medir unicamente −0.7 para un cero lógico y +4−5V para un uno lógico.

Inserta el microcontrolador solamente después de que tu circuito haya pasado las pruebas descriptas anteriormente.

Utilizando la Comunicación I2C

Descarga (ver referencias) el archivo linuxI2Ctemp tar.gz y desempaquétalo. La comunicación I2C está implementada en 2 archivos:

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

Yo le he dado a atmega8 la dirección esclava "3". Para enviar la cadena "hello" ("hola") a atmega8 ejecutarías las siguientes funciones C:

```
address_slave(3,0); // tell the slave that we will send something
i2c_tx_string("hello");
i2cstop(); // release the i2c bus

on the microcontroller side you would receive this "hello" string with
i2c_get_received_data(rec_buf);
```

Muy fácil. Leer datos del microcontrolador es similar. Mira el archivo i2ctemp_avr_main.c para ver cómo funciona cuando las lecturas de temperaturas estén hechas.

¿Cúan cálido es?

Para compilar y cargar el código para el microcontrolador, ejecuta los siguientes comandos del paquete de directorio linuxI2Ctemp.

```
make
make load
```

Compila los dos programas i2c_rs232_pintest y i2ctemp_linux

```
make i2c_rs232_pintest
make i2ctemp_linux
```

... o simplemente usa las versiones precompiladas en el "recipiente" subdirectorio.

Para leer las temperaturas simplemente ejecuta:

```
i2ctemp_linux
```

... e imprimirá las temperaturas internas y externas. Para hacer que estos datos se encuentren disponibles en un sitio web, sugiero no ejecutar directamente i2ctemp_linux desde el servidor web porque la comunicación i2c es muy lenta. En cambio, ejecútalo desde un cron job y escribe desde allí a un archivo html. Un ejemplo de secuencia de comandos se incluye en el archivo README del paquete linuxI2Ctemp.

Conclusión

El protocolo I2C requiere muy poco hardware extra y es óptimo para transmitir o recibir pequeñas cantidades de datos. Esto es exactamente lo que necesitamos cuando queremos comunicarnos con nuestro propio hardware de controlador. ¡Es realmente una muy buena solución!

En este artículo he puesto mucho énfasis en la parte de hardware. Si te gusta este artículo entonces escribiré un segundo en el que describo cómo funciona el software. Especialmente veremos cómo hacer la conversión

análoga a digital y cómo funciona la implementación del protocolo I2C. En el próximo artículo también podemos agregar una pantalla LCD y conversión entre Farenheit y Celsius.

Referencias

- **Página de descargas** para este artículo: el software linuxI2Ctemp, diagramas, actualizaciones de software
- Cómo programar el atmega8 con gcc: November2004 article 352
- Hoja de Datos para Atmega8: ver http://www.atmel.com/ y seleccionar products−>Microcontrollers −>AVR−8 bit RISC−>Documentation−>datasheets (local copy, pdf, 2479982 bytes)
- The tuxgraphics shop. Realmente una gran tienda en línea :−): shop.tuxgraphics.org
  Aquí puedes obtener el CD de programación Linux AVR, todas las piezas para este artículo, pantallas LCD y microcontroladores.

Contactar con el equipo de LinuFocus

http://www.LinuxFocus.org

Información sobre la traducción:
en −−> −− : Guido Socher (homepage)
en −−> es: Gabriela González <tradugag(at)yahoo.com>

2005−03−01, generated by lfparser_pdf version 2.51