SENSOR DE TEMPERATURA

Martínez Franco

Diseño e implementación de un Sistema de Hardware y Software, basado en una arquitectura ARM, que toma valores de un sensor de temperatura y, mediante comunicación UART, envía los resultados en una PC que los presenta en una interfaz de usuario.

Índice

Objetivo

Este trabajo consiste en el diseño e implementación de un sistema encargado de censar la temperatura de una sala y mostrar, mediante una aplicación desarrollada en Python, los valores medidos y un gráfico con el histórico de temperaturas.

Para este trabajo, se decidió trabajar con un potenciómetro ya que éste nos permite mostrar la funcionalidad del sistema con mayor facilidad. La implementación de éste con un sensor de temperatura es trivial y no representa mayores complicaciones.

Implementación

El sistema consiste de un potenciómetro que entrega un valor de tensión variable entre 0V y 3,3V. Este valor es tomado por el módulo ADC del microcontrolador LPC1769, convirtiendo esta señal a un valor digital de 12 bits. El LPC cada un intervalo de tiempo definido, lee el valor entregado por el potenciómetro.

El microcontrolador comunicará este valor a una computadora por UART. Para ello, convierte este valor binario a caracteres, para poder enviarlo correctamente.

La PC recibe el mensaje con el valor medido. Un programa desarrollado en Python toma esta medición, y muestra un gráfico de temperatura histórico con los últimos valores medidos.

Siendo que los valores entregados a la PC varían entre 0 y 4095, el programa agrupa las mediciones en rango que representan una variación entre 0°C y 45°C.

Elementos Principales Utilizados

Placa de Desarrollo LPCXpresso 1769. Microprocesador ARM Cortex M3.

Es el encargado de tomar el valor del sensor de temperatura (potenciómetro en este caso), prepararlo para comunicarlo, y enviarlo a la PC.

Potenciómetro de 1 KOhm.

Hace las veces de sensor de temperatura.

Driver Max 232 para la conexión serie entre ARM – PC.

Necesario para la comunicación serie. Adapta los niveles de tensiones de la señal que entrega el circuito del microprocesador y de lo que necesita la PC para interpretar correctamente el valor.

LPC

El microcontrolador se programó para que periódicamente cense el valor del potenciómetro. Esto se llevó a cabo utilizando interrupciones del Timer0.

Al momento de obtener el valor, se utiliza el módulo ADC, obteniendo un valor digital de 12 bits que representa esta medición.

Utilizando librerías de C, se convierte este valor a una cadena de caracteres, ya que, al ser la comunicación UART, enviamos caracteres por el puerto (cada carácter es de 8 bits).

En la PC, un programa desarrollado en Python escucha el puerto. Cuando llega un valor, lo toma, lo transforma en un entero, y lo almacena en una cola FIFO de tamaño fijo.

Esta cola hace las veces de buffer. Aquí se almacenarán los datos que llegan, y será desde aquí donde el programa lee los datos para mostrarlos en un gráfico.

Conexión

La conexión entre LPC y PC es a través de un Driver MAX232, el cual se transforma los valores de tensión a niveles TTL, interpretables por la PC.

Se utilizó un cable Serial-USB y se instalaron drivers para emular el puerto COM, ya que las computadoras donde se probó el programa no contaban con puertos seriales.

Código

Para el desarrollo del código, se han utilizado al máximo las capacidades de las librerías que provee la suite de desarrollo, por lo que el código es muy compacto, modularizado y de fácil comprensión.

Código C del programa del microcontrolador:

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

//ADC
#include "freq.h"
#include "adc.h"
#include "debug.h"
#include "small_systick.h"

//UART
#include "type.h"
#include "uart.h"

//Utilidades relacionadas al manejo de cadenas y caracteres.
#include <string.h>
#include <stdlib.h>

```

```c
#include <cr_section_macros.h>


volatile uint32_t SysTickCount;           /* counts 10ms timeTicks */
uint8_t cont;
volatile uint16_t adcCount;

// InitSysTick() sets systick timer to 10 mS
#define InitSysTick(MHz) SysTick_Config(MHz * (1000000/100))

/*-------------------------------------------------------------------------
  SysTick_Handler
  Cuando Timer0 interrumpe, medimos un valor en el ADC, lo convertimos a
  caracteres y lo enviamos por UART
 *-------------------------------------------------------------------------*/
void SysTick_Handler(void)  //interrumpe cada 10 milisegundos
{
        char mensaje[8];

        SysTickCount++; /* provee delay */

        if (SysTickCount < 100) return; //100 = 1 segundo

        //Tomamos el valor leido por el sensor
        adcCount = ADC0Read(0);  //Se lee el puerto P0[23]

        //Convierte el valor medido a caracteres (un caracter por cada
        //digito decimal)
        itoa(adcCount, mensaje, 10);

        //Agrega un retorno de carro y nueva línea al mensaje
        strcat(mensaje, "\r\n");

        //Enviamos por UART el valor:
        UARTSend(3, (uint8_t *)mensaje , strlen(mensaje) );
        SysTickCount = 0;

        //Prende y apaga un LED para saber que el sistema está funcionando:
        //Sirve como control
        if (cont == 0){
                LPC_GPIO0->FIOCLR = (1 << 22); // P0[22] = 0
                cont = 1;
        }
        else {
                LPC_GPIO0->FIOSET = (1 << 22); // P0[22] = 1
                cont = 0;
        }
}


int main(void) {

    // TODO: insert code here
        adcCount = 1000;

        LPC_GPIO0->FIODIR |= (1 << 22); // P0[22] COMO SALIDA.


        ADCInit(ADC_CLK);  //Inicializa el ADC. Entrada ADC por el puerto P0[23].
        cont = 0;
        InitSysTick(100);  //Inicializa el Timer para que interrumpa cada 10 ms.

        //UART Setting
```

```
82          UARTInit(3, 9600);        /* baud rate setting */
83
84      while(1) {
85      }
86      return 0 ;
87  }
```

Mencionamos las librerías utilizadas, que se encuentran en el IDE del LPC en caso de que se necesite consultarlas. En el anexo se encuentra el código de las librerías incluidas en el IDE.

```
//ADC
"freq.h"
"adc.h"
"debug.h"
"small_systick.h"

//UART
"type.h"
"uart.h"

//Utilidades relacionadas al manejo de cadenas y caracteres.
<string.h>
<stdlib.h>
```