SISTEMAS EMBEBIDOS
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Segundo Cuatrimestre de 2019

Laboratorio N° 3
Conversor Analógico Digital

Fecha de evaluación: Lunes 23/09/19
Objetivo: Introducir a los alumnos en el muestro y adquisición de datos utilizando el conversor analógico/digital presente en el microncontrolador ATmega328P.
Además, se utilizará la unidad USART para establecer transmisión de datos hacia el host y realizar una aplicación que permita visualizar los datos del sistema.
Desarrollo: El laboratorio deberá realizarse en comisiones de no más de 2 alumnos. Al finalizar la evaluación, se deberá comprimir y enviarse por mail respetando el siguiente formato: *LaboratorioX-ApellidosComision.zip*.

Descripción del hardware

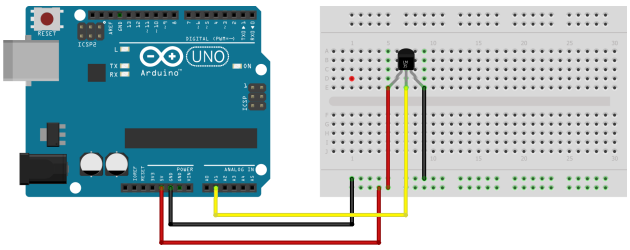
El Hardware a utilizar se compone de:

- Una placa Arduino Uno [1] con un microcontrolador ATmega328P [2].
- Protoboard y cables para realizar el conexionado.
- Una resistencia entre 75 y 120 ohms para la conexión del sensor (opcional).
- Un capacitor de 1uF para la conexión del sensor (opcional).
- Un sensor de temperatura lm35 conectado en modo de sensado básico, con un damper R-C, tal y como se muestra en la Figura 4 de la hoja de datos (ver [3]).

Actividad 1: Sensor digital de temperatura

Utilizando el protoboard, el sensor LM35, capacitor, resistencia de 100 ohms y placa Arduino se desea implementar un sensor digital de temperatura como se indica en la figura 1

Figura 1: Circuito Arduino y sensor LM35



1. Implementar un *driver* de ADC, que provea la siguiente función de inicialización

   - `bool adc_init(adc_cfg *cfg)`: Función que inicializa el *driver* de ADC y acepta como parámetro una estructura de tipo `adc_cfg`.
     La estructura `adc_cfg` tiene que tener un campo que selecciona el canal a configurar, y una función de callback que será invocada cuando se haya completado la conversión de un valor. Se puede añadir cualquier otro campo que se considere necesario.
     Esta función inicializará el ADC del ATMega328P en el modo de operación más conveniente para la aplicación (ver [2]) y asociará el canal seleccionado con su callback cada vez que sea invocada, retornando 1 si la configuración resultase exitosa, o 0 en caso de producirse un fallo en la inicialización o configuración.

   El *driver* debe ser capaz de recibir más de un llamado a la función `adc_init`, con distinta selección de canal, y ser capaz de asociar a cada canal su función *callback*, invocándola cuando una conversión de ese canal esté completa.

   El *driver* debe estar escrito en archivos independientes al programa principal, con la implementación escrita en un archivo .c/.cpp, y su interfaz pública declarada en un archivo *header* (.h).

   Para obtener el resultado de la conversión del ADC, el *driver* debe valerse de la interrupción provista por el periférico.

2. A partir del voltaje obtenido del sensor, se deberá poder convertir dicha cantidad a un valor en grados centígrados mediante el *driver* de ADC.

   Dicho valor de temperatura será presentado utilizando el display LCD. Además del valor de temperatura actual, el sistema deberá llevar registro de las temperaturas máxima y mínima registradas desde el inicio de la ejecución, y de la temperatura promedio. Utilizando el *driver* de TECLADO del laboratorio anterior se deberá poder seleccionar qué temperatura se visualiza (actual, máxima, mínima o promedio). Para el cálculo del promedio, se espera que el sistema conserve, en un arreglo circular, 100 muestras de temperaturas tomadas a lo largo de los últimos 15 segundos.

   Para el sistema tenga en cuenta las siguientes consideracion de diseño:

   - En base al valor de voltaje entregado por el sensor y al voltaje de referencia utilizado por el ADC, determine la función para convertir cada valor digitalizado en el valor de temperatura correspondiente en C. Incluya los cálculos realizados.
   - Estime la frecuencia de muestreo necesaria para obtener los valores utilizados en el cálculo de la temperatura promedio.
   - Identifique las tareas del sistema e indique para cada una los dispositivos asociados (display, pulsadores, ADC, etc)
   - Determine los eventos que iniciarán la ejecución de cada una de las tareas identificadas.
   - Determine las dependencias (sincronización y comunicación) entre tareas así como los datos compartidos entre ellas.

Actividad 2: Interface de control sensor digital de temperatura

1. Se desea incorporar al Sensor Digital de Temperatura realizado en la actividad anterior una aplicación en el *host* que:

- Permita graficar las temperaturas obtenidas (actual, máxima, mínima, y promedio) durante el último minuto de ejecución del sistema
- Muestre los valores numéricos de las cuatro temperaturas obtenidas (actual, máxima, mínima y promedio).
- Provea al usuario de botones que le permitan cambiar el modo de operación del sensor (para seleccionar qué temperatura se visualiza por el display LCD del mismo), permitiendo ejercer el control del sensor desde el host.

Para ello se requiere modificar el sistema implementado en el laboratorio anterior, añadiéndole la capacidad de comunicarse con el host de manera asincrónica mediante la UART del ATmega328P , apelando a la librería Arduino Serial.

Adicionalmente, se requiere la implementación de una aplicación en el host que realice las tareas indicadas anteriormente. Para simplificar esta parte del proyecto, se provee un ejemplo realizado en el lenguaje Processing [7].

El ejemplo dispone de la mayor parte de la funcionalidad para la interface de control ya implementada, y se espera que los alumnos lo modifiquen adaptando la funcionalidad existente e incorporando la funcionalidad faltante.

Si alguna comisión considera utilizar otra plataforma de desarrollo para la aplicación host, podrá implementar esta parte del proyecto siempre que la plataforma elegida ofrezca la funcionalidad gráfica y permita realizar comunicación través de los puertos serie de la PC.

En base a la descripción dada previamente

a) Estudie y familiarícese con la librería Serial [4] de Arduino.

b) Estudie y familiarícese con la librería Serial [5] de Processing.

c) Descargue y abra el ejemplo *"SE_Lab_HostTempApp"* [6] de Processing disponible en la página web de la materia.
Examine el código del ejemplo y familiarícese con el funcionamiento del mismo.

d) Describa las modificaciones a realizar en la aplicación desarrollada en la actividad anterior para incorporar la capacidad de realizar comunicación bidireccional en serie asincrónica. Añada la descripción de las tareas incorporadas para tal fin. Indique qué eventos desencadenan la ejecución de dichas tareas y qué cambios, si existen, deben producirse en la estructura de control del sensor como consecuencia de su incorporación (por ejemplo, manejo de mensajes entrantes, manejo de mensajes salientes, etc).

e) Describa brevemente las modificaciones a realizar sobre el ejemplo para adaptarlo a los requerimientos del laboratorio.

f) Implemente las modificaciones planteadas precedentemente.

Referencias

[1] Arduino Uno WebSite. `https://arduino.cc/en/Main/arduinoBoardUno`.

[2] Atmel AVR ATmega48A/48PA/88A/88PA/168A/168PA/328/328P Data Sheet.

[3] LM35 Precision Centigrade Temperature Sensors Data Sheet.

[4] Libreria "Serial" para Arduino. http://arduino.cc/en/Reference/Serial.

[5] Biblioteca “Serial” de Processing. http://processing.org/reference/libraries/serial/index.html.

[6] Ejemplo “SE_Lab_HostTempApp” disponible en la pagina web de la materia.