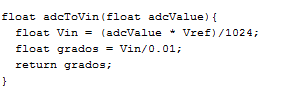
Actividad 1



El resultado de la conversión ADC (almacenada en los registros ADCL, ADCH) es la

expresión ADC = (Vin \* 1024) / Vref. En nuestro caso Vin es el voltaje enviado por el sensor

y Vref el voltaje de referencia (los 5 Volts de alimentación).

De esta expresión se despejó el voltaje de entrada Vin = (ADC \* Vref) / 1024 y luego se usó

la expresión de salida del sensor Temp = 0 [mV] + 10.0 [mV /C°] para finalmente llegar a la

conclusión de que la temperatura es Vin / 0.01 [V] = Vin / 10 [mV], ya que se aumentan 10

mV por °C.

Para obtener el cálculo de la temperatura promedio, el sistema almacena en un arreglo circular los últimos 100 valores tomados a lo largo de los últimos 15 segundos. Para esto se realiza un control de tiempo para que las inserciones se realicen cada 15 centécimas de segundo.

Tareas del sistema:

Loop():

Tarea principal del sistema, ejecuta las funciones encoladas.

Dispositivos asociados: Ninguno.

Eventos que incian su ejecución: finalización del método setup().

CleanDisplay():

Limpia el display LCD.

Dispositivos asociados: Display LCD.

Eventos que inician su ejecución: finalización de impresión del mensaje inicial.

AdcToTemp():

En base al valor leído del ADC, lo convierte a un valor actual de temperatura en grados centígrados, almacena las mediciones en el arreglo y registra cambios en los valores de temperatura mínima y máxima.

Dispositivos asociados: Conversor ADC, Timer 2.

Eventos que inician su ejecución: llamado a la función de callback cuando se procesa una conversión.

AdcToVin():

Convierte el valor leído del ADC en un valor de temperatura en grados centígrados.

Dispositivos asociados: Conversor ADC.

Eventos que inician su ejecución: llamado a la función de callback cuando se procesa una conversión.

AgregarMedicion():

Inserta una nueva medición en el arreglo de mediciones, del cuál se extrae el promedio de temperaturas.

Dispositivos asociados: Conversor ADC.

Eventos que inician su ejecución: llamado a la función de callback cuando se procesa una conversión. Cumplimiento de periodo de tiempo.

ObtenerPromedio():

Realiza el cálculo del promedio de los valores almacenados en el arreglo de mediciones.

Dispositivos asociados: Ninguno.

Eventos que inician su ejecución: Cambio al modo de mostrar promedio de temperaturas.

ImprimirInicio():

Imprime en el display LCD el mensaje inicial del sistema.

Dispositivos asociados: Display LCD.

Eventos que inician su ejecución: Inicio del sistema.

MostrarEstado():

Imprime en el display LCD información del estado actual del sistema.

Dispositivos asociados: Display LCD, pulsadores.

Eventos que incian su ejecución: cambio de modos por medio de los pulsadores.

ImprimirBrillo():

Imprime en el display LCD el nivel de brillo del mismo.

Dispositivos asociados: Display LCD, pulsadores.

Eventos que incian su ejecución: Cambio al modo de mostrar nivel de brillo del display.

ProcesarTimer():

Método invocado por la rutina de interrupción del timer2. Realiza el conteo de tiempo en centésimas de segundo.

Dispositivos asociados: Timer 2.

Eventos que inician su ejecución: Interrupción de la rutina ISR del timer.

ISR(TIMER2\_COMPA\_vect):

Rutina de interrupción del timer 2.

Dispositivos asociados: Timer 2.

Eventos que inician su ejecución: Interrupción del timer.

ModoActual():

Imprime en el display LCD el valor asociado al modo en que se encuentra el sistema.

Dispositivos asociados: Display LCD, pulsadores.

Eventos que inician su ejecución: cambio de modos por medio de los pulsadores.

Actividad 2

La comunicación entre el firmware en Arduino y el host se llevará a cabo a través del

puerto serie COM donde se encuentre conectado el arduino, mediante la escritura y la lectura asincrónica de los datos.

Las temperaturas serán enviadas desde Arduino al host todo el tiempo, cada medio segundo para evitar la saturación del buffer, dentro del ciclo de ejecución del método ModoActual.El envío se coordinará para obtener una precisa recepción de los datos enviando un indicador de temperatura, previo al envío de la temperatura en sí.

Esto es:

- Se enviará el valor 200 para indicar que se está por enviar la temperatura actual, y seguido de esto se enviará dicha temperatura.

- Se enviará el valor 201 para indicar que se está por enviar la temperatura máxima, y seguido de esto se enviará dicha temperatura.

- Se enviará el valor 202 para indicar que se está por enviar la temperatura mínima, y seguido de esto se enviará dicha temperatura.

- Se enviará el valor 203 para indicar que se está por enviar la temperatura promedio,y seguido de esto se enviará dicha temperatura.

De la misma manera e inmediatamente seguido al envío de las temperaturas, se realizará

el envío del modo actual de la placa.

- Se enviará el valor 204 para indicar que se está por enviar el modo actual de la placa, y seguido de esto se enviará dicho modo.

Del lado del host se realizarán lecturas constantes del puerto serie dentro del método Draw, una para cada uno de los envíos antes mencionados, generando de esta manera una obtención de los datos correcta y precisa.

Luego de realizarse estas 10 escrituras y lecturas en el puerto serie (5 para el envío y recepción de los indicadores previos y 5 para cada uno de los datos), desde el host se verificará si se ha presionado alguno de los 4 botones presentes en la interfaz gráfica, correspondientes a cada una de las temperaturas. De ser así, se enviará hacia Arduino el valor correspondiente al botón que se haya presionado mediante una única escritura en el puerto serie (la escritura del valor del botón). Del lado de Arduino pues, se preguntará dentro de ModoActual al puerto serie si hay algún valor para leer, y de encontrarse uno, este será leído y luego será asignado al estadoActual del sistema para una correcta visualización en el display LCD.

Respecto a arduino, no hubo que realizar demasiadas modificaciones a la aplicación de la actividad 1, simplemente agregar las variables y el código necesario para permitir la comunicación con el host.

Con este propósito se utilizó la librería Serial deArduino. De ella usamos las siguientes

funciones:

• begin(): Configura la velocidad de transferencia de datos en el puerto serie (9600 baudios es el valor más común).

• available(): Permite verificar si hay elementos (Bytes) sin leer en el buffer del puerto serie. Este buffer puede almacenar hasta 64 bytes (en nuestro caso equivale a 64 datos y/o mensajes de codificación).

• write(): Permite escribir datos en el puerto serie en formade bytes o series de bytes.

• read(): Permite leer datos que entran por el puerto serie.

Hubo que agregar algunas variables para mantener los valores actualizados de los datos que son enviados/recibidos. Estos son almacenados en bytes, de esta manera pueden enviarse y recibirse de manera sencilla por el puerto serie.

En el método modoActual() de la aplicación se agregó el código necesario para que se envíen las

temperaturas y el modo a través del puerto serie siguiendo el protocolo definido previamente. Luego de los 10 envíos, se realiza una lectura en el puerto serie con la cual se obtiene el último botón presionado desde la aplicación encargada de la interfaz gráfica.

Respecto a Processing, hubo que importar la librería processing.serial para permitir la comunicación a través del puerto serie con la aplicación hecha en Arduino. Se definió una variable myPortde tipo Serial, para utilizar las funciones de comunicacióncon el puerto serie que ofrece la librería. También se definieron variables para almacenar los valores que son recibidos desde la aplicación de Arduino a través del puerto serie COM que se encuentre conectado el arduino.

Se definió una variable boton de tipo byte, que almacenará un valor indicador del botón presionado en la interfaz gráfica (en caso de que haya sido presionado alguno).

Las lecturas y escrituras en el puerto serie se manejaron de manera similar a como se hizo

previamente: se utilizaron las funciones available(), read() y write() de la misma forma que se las usó en el lenguaje Arduino. En el método setup()se creó el objeto de tipo Serial myPort, indicando mediante los parámetros de su constructor el puerto COM que se utilizaría y la velocidad de transferencia (los cuales deben coincidir con los de la otra aplicación).

Se definieron 4 variables del tipo ScrollingFcnPlot para poder graficar las temperaturas del sensor. Se manejaron de la misma manera que la función del ejemplo: se crea la función en el setup y luego se actualizan los valores y se dibujan en el bucle de la aplicación, mediante los métodos updateValue()y displayIntoRect().