Sistemas Embebidos
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Segundo Cuatrimestre de 2025

Laboratorio N° 2
E/S y Dispositivos Periféricos Integrados

Fecha de entrega: Lunes 15/09/2025

Entregables

Los resultados de este laboratorio se deberán comprimir y enviarse por mail respetando el siguiente formato **LaboratorioX-ApellidosComision.zip** para su evaluación. Los docentes repetirán la configuración de hardware necesario para hacer funcionar cada actividad, y verificarán que el código fuente respete el funcionamiento solicitado. En el momento de la fecha de entrega, los estudiantes deben enviar a la cátedra a la siguiente direccion **sistemas.embebidos@cs.uns.edu.ar**:

- Código fuente de cada una de las actividades.
- Un informe **conciso** (lo mínimo solicitado) y **completo** (lo que necesita la cátedra saber sobre el proyecto para compilarlo y cargarlo, y comentarios adicionales que sean *importantes* sobre la entrega).
  - identificar las tareas e ISRs del sistema pedido en el punto 2 de la actividad 2.
  - la inclusión de los puntos 3, 4 y 5 de la actividad 3. Añada toda decisión o limitación de diseño que considere relevante.

Actividad 1: Librería LyquidCrystal, LCD1602 Keypad Shield y PWM

1. Abrir el ejemplo **LCD1602KeyShieldTest** [1] provisto por la cátedra. Examinar el código del ejemplo y familiarizarse con el mismo. Prestar atención al manejo de los pulsadores realizados mediante un único canal de ADC, y al uso de la función `analogRead` [2]. ¿Qué ventaja presenta este esquema de conexión, en relación a conectar pulsadores de manera directa a los puertos de E/S?

2. Construir el proyecto, descargar la imagen al microcontrolador y probarlo.

3. Examinar la documentación de la función `analogWrite` [3] y determinar de qué manera es utilizada para controlar el brillo del backlight del display LCD. ¿Cómo es posible generar una señal analógica mediante un pin de salida digital?

4. Experimentar con diversos valores de brillo para el *backlight*,modificando el código para cada caso.

Actividad 2: Driver de teclado

Se desea modificar el ejemplo de la actividad 1 para que el sensado de los pulsadores se realice usando interrupciones y sin retardos por software. En lugar de realizar polling para los pulsadores, se desea que el ADC sea quien interrumpa al microprocesador cuando analice la conversión del valor del teclado, de manera que el CPU pueda continuar realizando otras tareas. Para ello se requiere implementar un driver no bloqueante mediante el uso de una planificación de *cola de funciones* para el teclado del LCD Keypad Shield siguiendo el código `fnqueue.cpp` brindado por la cátedra [4]. El driver debe implementar las ISRs y tareas necesarias para su funcionamiento, y además tener la siguiente interfaz

- `void key_down_callback(void (*handler)(),int tecla):` Asocia la función `handler` al evento de presionar la tecla `tecla`.
- `void key_up_callback(void (*handler)(), int tecla):` Asocia la función `handler` al evento de soltar la tecla `tecla`.

donde `tecla` puede tomar uno de los siguientes valores:

- `TECLA_UP = 0` *//botón up del LCD Keypad Shield*
- `TECLA_DOWN = 1` *//botón down del LCD Keypad Shield*
- `TECLA_LEFT = 2` *//botón left del LCD Keypad Shield*
- `TECLA_RIGHT = 3` *//botón right del LCD Keypad Shield*
- `TECLA_SELECT = 4` *//botón select del LCD Keypad Shield*

1. Analizar el tutorial [5] para utilizar interrupciones directamente mediante las librerías AVR, desde Arduino.
2. Identificar las tareas e ISRs del sistema. Especificar la lógica de cada tarea e ISR mediante diagramas de flujo.
3. Determinar en base a la información disponible en [6] y a los requerimientos del proyecto, el modo más conveniente para configurar el ADC. Justifique su elección.
4. Implementar el driver descripto anteriormente. Se puede seguir los pasos descriptos en [7]
5. Descargar la imagen modificada al microcontrolador y probar el sistema.

Actividad 3: Contadores, Timers, PWM y E/S Digital

Se desea implementar un contador automático con dimmer. El sistema cuenta con el display LCD de 16×2 del LCD Keypad Shield, donde se visualizarán los tiempos medidos (en formato M.SS.d,i.e el dígito más significativo para los minutos, los dos siguiente para los segundos y el menos significativo para las décimas). Los minutos, segundos y las décimas de segundo deberan estar separados por el punto decimal. En todo momento deberá mostrarse el valor del conteo y el modo de operación actual. Además el sistema hará uso de 3 de los 5 pulsadores del shield (Select, Up y Down). En el sistema se puede variar la magnitud del paso del contador, el tiempo (en segundos)

que transcurre entre cambios y el brillo del display. Para la implementación, debe utilizarse el *driver* de teclado implementado en la actividad anterior. El sistema cuenta con los siguientes modos de operación:

- *Modo de Conteo Ascendente (MCA): El cronómetro cuenta en sentido ascendente el tiempo transcurrido. El display muestra de manera continuada dicho tiempo*
- *Modo Pausa (MP): El cronometro se pausa y el display muestra el tiempo transcurrido desde el último reseteo hasta que se pausa.*
- *Modo Visor Tiempos (MVT): El cronómetro muestra hasta 10 tiempos previamente guardados.*
- *Modo de ajuste de dimmer (MAD): En este modo, es posible ajustar la intensidad luminosa del backlight del display LCD (el cual se mantendrá mostrando el tiempo pausado, el modo y el brillo actual) usando los pulsadores Up y Down (20 %, 40 %, 60 %, 80 % o 100 % del brillo).*

El sistema hace uso de tres pulsadores con la siguiente funcionalidad:

- *UP:* en MCA, pausa el conteo temporal. En MP, continúa el conteo temporal ascendente. En MVT, muestra el siguiente tiempo guardado. En MAD, incrementa el nivel de brillo del display.
- *DOWN:* en MCA, guarda el tiempo actual. En MP, guarda el tiempo actual y vuelve el contador a 0. En MVT, muestra el anterior tiempo guardado. En MAD, disminuye el nivel de brillo del display
- *SELECT:* en MP, pasa a MVT si se mantiene presionado menos de 3 segundos, y pasa a MAD si se mantiene presionado más de 3 segundos. En MVT, pasa a MP. En MAD, vuelve a MP. En MCA, no tiene efecto

El sistema inicialmente muestra por el display datos del laboratorio, materia, cuatrimestre y comisión desarrolladora (utilizar scrolling [3] si el texto es demasiado largo). Al presionar cualquiera de las 3 teclas, se inicia en MP con el conteo de tiempo en cero y con un nivel de brillo del display del 80 %. Mediante el pulsador Up y Select se puede alternar entre los diferentes modos. Para pasar a MAD,en MP se deberá mantener pulsado Select durante más de 3 segundos. El sistema retornará automáticamente de MAD a MP al no registrarse ninguna intervención del usuario sobre los pulsadores durante más de 5 segundos.
En base a la descripción del sistema dada previamente:

1. Analizar el tutorial [8] para utilizar timers directamente mediante las librerías AVR, desde Arduino.
2. Identificar las tareas e ISRs del sistema e indicar, para cada una, los dispositivos asociados (display, pulsadores, timers,etc).
3. Especificar la lógica de cada tarea e ISR identificada mediante diagramas de flujo. Determinar los eventos que iniciarán la ejecución de cada una de las tareas identificadas.
4. Representar mediante un diagrama de transición de estados, los modos de operación del sistema y los eventos que ocasionan los cambios entre dichos modos.
5. Determinar las dependencias (sincronización y comunicación) entre tareas así como los datos compartidos entre ellas.

6. Integrar el sistema, construir y depurar el proyecto.

7. Descargar la imagen al microcontrolador y probar el sistema.

Referencias

[1] Ejemplo "LCD1602KeyShieldTest" disponible en la pagina web de la materia.

[2] Analog Read. https://docs.arduino.cc/built-in-examples/basics/AnalogReadSerial/.

[3] Analog Write. https://docs.arduino.cc/language-reference/en/functions/analog-io/analogWrite/.

[4] David E. simon. *An Embedded Software Primer. Cap.5, sec. 3* . 1999.

[5] Timer Interrupt in Atmega16-AVR.

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

[7] Michael Barr. *Programming Embedded Systems in C and C++*. O'Reilly, 1999.

[8] Elliot Williams. *Make: AVR Programming: Learning to Write Software for Hardware*. Maker Media, Inc, 2014.

[9] Newbie's Guide to AVR interrupts. https://www.avrfreaks.net/forum/tut-newbies-guide-avr-interrupts.