Sistemas Embebidos
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Segundo Cuatrimestre de 2019

Laboratorio N° 1

Introducción a las herramientas de desarrollo de Sistemas Embebidos

Fecha de evaluación: Lunes 26/08/19
Objetivo: Introducir al conjunto de herramientas en el desarrollo de sistemas embebidos, manejo de temporizado por software y debouncing de un pulsador.
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.
- un pulsador.
- una resistencia de 10k (para conectar el pulsador).
- el led integrado en la placa Arduino Uno.
- con posterioridad se utilizará un capacitor de 1uF.

Actividad 1: Introducción al desarrollo en C embebido

1. Cree un proyecto seleccionando el compilador de GCC++ como herramienta de desarrollo de Atmel Studio.
2. Añadir el código ejemplo *begginer.c* provisto por la cátedra.
3. Construya el proyecto y depurelo/simulelo utilizando Atmel Studio [3] [4]

Actividad 2: Polling y copia de un puerto a otro

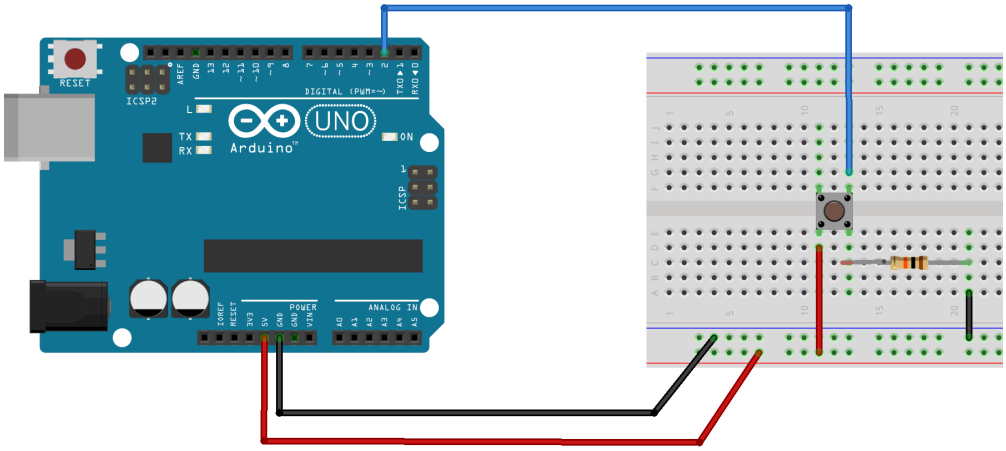
1. Abrir el entorno Atmel Studio y familiarizarse con las principales componentes y opciones, la placa Arduino Uno [1] y el microcontrolador ATmega328P [2] integrado.
2. Crear un nuevo proyecto y añadir el archivo *button.cpp* provisto por la cátedra.
3. Construir la imagen ejecutable del ejemplo.

4. Examinar los archivos producidos como resultado, asociándolos a las diferentes etapas de la generación de la imagen ejecutable del proyecto (ensamblado, linkeo, etc). Analice el proceso de construcción del software del toolchain de AVR en términos de las etapas antes mencionadas observando qué archivo se genera/utiliza en cada etapa.

5. Simular y depurar el proyecto utilizando el simulador integrado en el Atmel Studio. Para ello genere entradas que simulen el efecto del pulsador en el sistema de manera directa modificando el contenido del port adecuado mediante breakpoints.

6. Realizar el cableado del ejemplo "Button" en el protoboard y conectarlo a la placa Arduino tal como se muestra en la siguiente figura:

7. Analizar que sucede si quitamos la resistencia. Investigue a que se denomina "resistencia PULLUP" y "resistencia PULLDOWN".



8. Descargar la imagen al microcontrolador y probar el programa.

Actividad 3: Debouncing

1. Examinar el código del ejemplo *flanco.c* provisto por la cátedra, y determinar cómo debería comportarse.

2. Añadir el capacitor (una solución de debouncing por hardware) conectándolo en paralelo con el pulsador y volver a probar el funcionamiento del programa en el target.

3. Examinar el código del ejemplo "Debounce" provisto por la cátedra y analizar la estrategia de debouncing por sotfware utilizada. Probar el programa con el hardware del inciso 1.

4. Comparar el comportamiento del ejemplo "Button" con el del ejemplo "Flanco" realizado en el inciso 1. ¿Cuándo es necesaria la aplicación de una estrategia de debouncing? Analizar.

5. Analizar comparativamente el debouncing por hardware (inciso 2) y el deboucing por software (inciso 3). Considere tanto ventajas y desventajas, como ejemplos de aplicación donde resulte conveniente cada una.

Actividad 4: Temporizado y retardos por software

1. Crear un proyecto nuevo en Atmel Studio.
2. Escriba una rutina que permita cambiar el estado de un led (pasar de on a off y visceversa).
3. Utilizando la función del punto 2 y la función `_delay_ms`, desarrollar un programa que permita seleccionar el modo de operación del led conectado al pin 13 del Arduino. Puede basarse en el ejemplo Blink provisto por la cátedra. El led podrá variar entre los siguientes modos de operación secuencialmente: *encendido*, *titilando a 0.5 Hz* (1 segundo apagado y 1 segundo encendio), *titilando a 1 Hz* (0.5 segundo apagado y 0.5 segundos encendido), *titilando a 2 Hz* (0.25 segundos apagado y 0.25 segundo encendido), *medio segundo encendido y un segundo apagado*, por último *apagado*.
4. Construir el proyecto. Simularlo y depurarlo como se hizo en la actividad anterior.
5. Analizar las ventajas y desventajas del esquema de polling utilizado (en relación a procesar la entrada mediante interrupciones).
6. Analizar las ventajas y desventajas de utilizar retardos por software (en relación a utilizar timers para la implementación de retardos por hardware).

Actividad 5: Librerías de Arduino. Librería Serial

1. Abrir el ejemplo *buttonArduino.cpp* en Atmel Studio provisto por la cátedra. Compararlo con el ejemplo "Button" de la Actividad 2. ¿Qué realizan ambos ejemplos? Analice.
2. Explique qué tareas realizan las funciones pinMode, digitalRead y digitalWrite. ¿Cómo se logra el mismo resultado en código AVR? Compare ambas alternativas en términos de performance, facilidad de uso, etc.
3. Configurar el entorno para poder compilar el ejemplo y descargarlo al microcontrolador teniendo en cuenta lo siguiente para todas las configuraciones:
   - AVR/GNU C++ Compiler → Directories:
     - `c:/arduino/hardware/arduino/avr/cores/arduino`
     - `c:/arduino/hardware/arduino/avr/variants/standard`

     Para usar las librerías externas de Arduino hay que agregar entradas adicionales para cada una, por ejemplo: `c:/arduino/libraries/LiquidCrystal/`
   - AVR/GNU Linker → Libraries:
     - Agregar la librería libcoreArduinoUno.a para que linkee con ella. Si se usan algunas operaciones matemáticas puede hacer falta agregar la librería libm.a.
     - Agregar el directorio donde se encuentra la librería libcoreArduinoUno.a provista por la cátedra al Library Search Path.
4. Estudiar la documentación de la librería "Serial" de Arduino [5]. En particular, analizar los ejemplos "Digital Read Serial"- [6] y "Digital Write Serial" [7]. Prestar especial atención al manejo de la librería Serial para lectura y escritura. Analizar qué recursos de hardware son utilizados por la librería y cómo influye ello en el diseño del software embebido.

5. Modificar el programa desarrollado en la Actividad 4 para que cambie el modo de operación del led según un número leído desde entrada Serie: (1) *encendido*, (2) *titilando a 0.5 Hz*, (3) *titilando a 1 Hz*, (3) *titilando a 2 Hz*, (4) *medio segundo encendido y un segundo apagado*, (5) *apagado*, y muestre por la salida Serie la frecuencia a la que titila.

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

7. Abrir el monitor serial en el host (el que sea que usemos) y asegurarse de que la tasa de transferencia (bps) coincida con la indicada en la inicialización de la librería Serial en el firmware descargado.

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] https://www.microchip.com/mplab/avr-support/atmel-studio-7.

[4] http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-Studio-7-User-Guide.pdf.

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

[6] Digital Read Serial. http://arduino.cc/en/Tutorial/DigitalReadSerial.

[7] Switch (case) Statement, used with serial input. http://arduino.cc/en/Tutorial/SwitchCase2.