



Laboratorio N° 1

Introducción a las herramientas de desarrollo de Sistemas Embebidos

Fecha límite de presentación: Viernes 01/09/2017. 17:00 hs

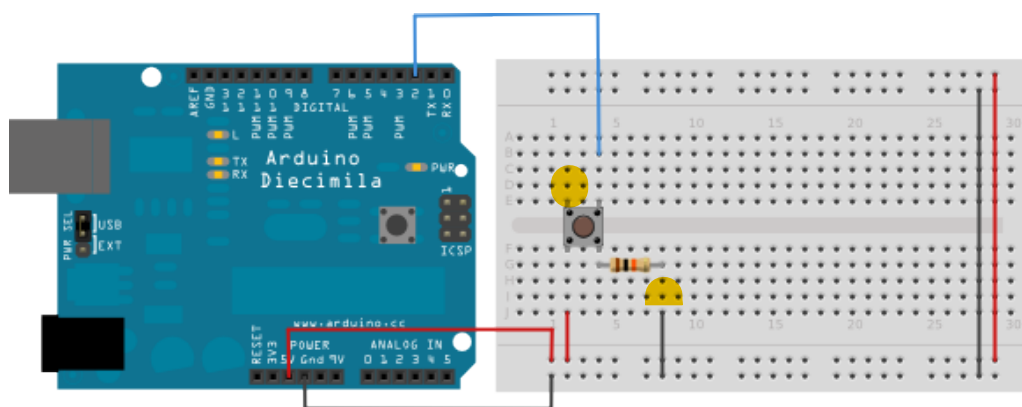
Fecha de evaluación: Lunes 04/09/2017

Objetivo: Introducir a los alumnos en el conjunto de herramientas de desarrollo de sistemas embebidos.

Desarrollo: El laboratorio deberá realizarse en comisiones de no más de 2 alumnos.

Actividad 1: 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. Descargar la imagen al microcontrolador y probar el programa.

Actividad 2: Debouncing

1. Examinar el código del ejemplo “Flanco” provisto por la cátedra, y determinar cómo debería comportarse. Probar el programa.
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 software 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 debouncing por software (inciso 3). Considere tanto ventajas y desventajas, como ejemplos de aplicación donde resulte conveniente cada una.

Actividad 3: Temporizado y retardos por software

1. Crear un proyecto nuevo en Atmel Studio.
2. Escribir una función `toggleLed` que permita cambiar el estado de un led (pasar de on a off y viceversa).
3. Determinar a qué pin del microcontrolador ATmega328P está asociado cada uno de los pines digitales 0 a 13 y los analógicos A0 a A5 de la placa Arduino.
4. 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*, *titilando a 1 Hz*, *titilando a 2 Hz*, *medio segundo encendido y un segundo apagado*, *apagado*. Implemente el programa mediante un único ciclo sin interrupciones en el cual se controle el estado del pulsador, luego se actualice el modo de operación del led y en función de dicho modo, se realice el control de encendido y apagado de acuerdo a lo especificado. Luego del reset, el sistema deberá iniciar su ejecución en el estado *encendido*.
5. Construir el proyecto. Simularlo y depurarlo como se hizo en la actividad anterior.
6. Analizar las ventajas y desventajas del esquema de polling utilizado (en relación a procesar la entrada mediante interrupciones).
7. Analizar las ventajas y desventajas de utilizar retardos por software (en relación a utilizar timers para la implementación de retardos por hardware).

Actividad 4: Librerías de Arduino. Librería Serial

1. Abrir el ejemplo “ButtonArduino” en Atmel Studio provisto por la cátedra. Compararlo con el ejemplo “Button” de la Actividad 1. ¿Qué realizan ambos ejemplos? Analice.
2. Examinar el código fuente de la librería Arduino.
 - a) Arduino inicia la ejecución en la función `main`. ¿Dónde se encuentra declarada esta función en las librerías de Arduino? Analice el flujo de control de dicha función y determine qué tareas llevan a cabo cada una de las funciones invocadas desde ella.

- b) Determinar dónde están definidas las constantes OUTPUT, INPUT, HIGH y LOW y qué valores representan.
 - c) Determinar dónde y cómo se implementa la correspondencia entre la numeración de los pines de Arduino y los pines del ATmega328P.
3. 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.
 4. 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 `libcoreArduinoUno161.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 `libcoreArduinoUno161.a` provista por la cátedra al Library Search Path.
 5. Estudiar la documentación de la librería “Serial” de Arduino [4]. En particular, analizar los ejemplos “Digital Read Serial” [3] y “Digital Write Serial” [5]. 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.
 6. Modificar el programa desarrollado en la Actividad 3 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.
 7. Construir el proyecto, descargar la imagen al microcontrolador y probarlo.
 8. 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. <http://arduino.cc/en/Main/arduinoBoardUno>.
- [2] Atmel AVR ATmega48A/48PA/88A/88PA/168A/168PA/328/328P Data Sheet.
- [3] Digital Read Serial. <http://arduino.cc/en/Tutorial/DigitalReadSerial>.
- [4] Librería “Serial” para Arduino. <http://arduino.cc/en/Reference/Serial>.
- [5] Switch (case) Statement, used with serial input. <http://arduino.cc/en/Tutorial/SwitchCase2>.