SISTEMAS EMBEBIDOS
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Segundo Cuatrimestre de 2025

Laboratorio N° 1
Introduccion a las herramientas de desarrollo de Sistemas Embebidos

Fecha de entrega: Viernes 29/08/25

Entregables

Los resultados de las actividades de este laboratorio deben entregarse a la cátedra 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.

Los estudiantes deberán enviar a **sistemas.embebidos@cs.uns.edu.ar** el laboratorio resuelto respetando el siguiente formato LaboratorioX-ApellidoComision.zip:

- 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).
    - la descripción del proceso de construcción de la imagen ejecutable pedido en el punto **??** de la actividad 2.
    - los análisis comparativos de los puntos 4 y 5 de la actividad 3.
    - un diagrama de flujo que ilustre la lógica del programa implementado en el punto 3 de la actividad 4
    - el analisis pedido en el punto 6 de la actividad 4

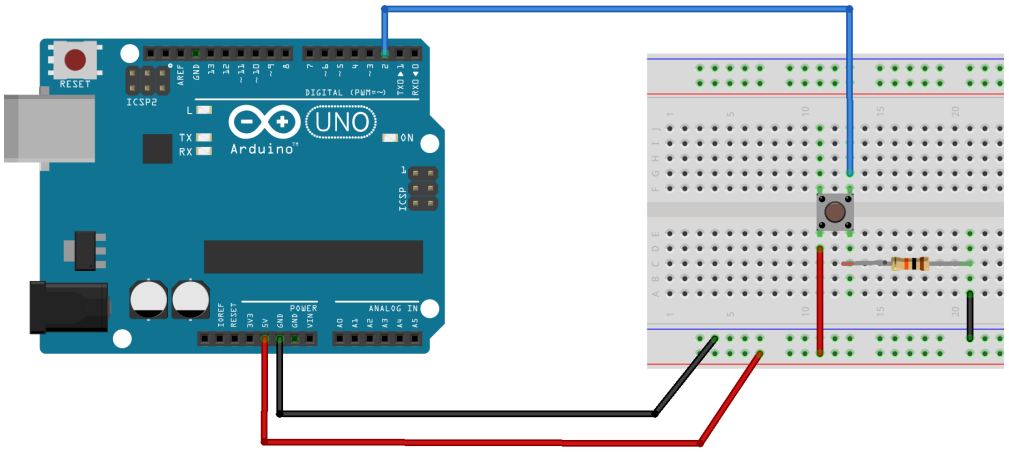
Actividad 1: Introducción al desarrollo en C embebido

1. Cree un proyecto en PlatformIO utilizando el framework Arduino (o baremetal) y configure el compilador GCC/C++ como herramienta de construcción.
2. Añadir el código ejemplo `blink.c` provisto por la cátedra.
3. Construya el proyecto y depurelo/simulelo utilizando PlatformIO [1]

Actividad 2: Polling y copia de un puerto a otro

1. Abrir el entorno de PlatformIO. Crear un nuevo entorno y añadir el archivo `button.cpp` provisto por la cátedra.
2. Construir la imagen ejecutable del ejemplo.

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



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

5. Analizar que sucede si quitamos la resistencia. Responda en el informe a que se denomina "resistencia PULLUP" y "resistencia PULLDOWN".

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? Responda en el informe.

5. Analizar comparativamente y reflejar en el informe el *debouncing* por hardware (inciso 2) y el *debouncing* por software (inciso 3). Ventajas y desventajas. Considere ventajas y desventajas, y ejemplos de aplicación donde resulte conveniente cada estrategia.

Actividad 4: Temporizado y retardos por software

1. Crear un nuevo entorno en PlatformIO.

2. Escriba una rutina que permita cambiar el estado de un LED (pasar de on a off y viceversa).

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 de 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 PlatformIO 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 desempeño, facilidad de uso, etc.

3. Estudiar la documentación de la librería "Serial" de Arduino [2]. En particular, analizar los ejemplos "Digital Read Serial" [3] y "Digital Write Serial" [4]. 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.

4. 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.

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

6. 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] PlatformIO. https://docs.platformio.org/en/latest/.

[2] Librería "Serial"para Arduino. http://arduino.cc/en/Reference/Serial.

[3] Digital Read Serial. https://docs.arduino.cc/built-in-examples/basics/DigitalReadSerial/.

[4] Digital Write Serial. https://www.arduino.cc/en/Serial/Write.