UNER Facultad de Ingeniería

Trabajo Práctico IV

Fundamentos de Programación de Sistemas Embebidos
Especialización en Sistemas Embebidos
Cohorte 2018

Objetivos

- Conocer el rol de los diferentes archivos que constituyen una aplicación para un sistema embebido basado en microcontrolador.
- Utilizar las diferentes herramientas disponibles para la construcción de aplicaciones y su depuración.
- Adquirir destreza en el uso del entorno de desarrollo propuesto.
- Comenzar a utilizar EDU-CIAA NXP como sistema embebido de desarrollo.
- Incursionar en el desarrollo de aplicaciones para sistemas embebidos y su problemática asociada.

Herramientas requeridas

Para el desarrollo del presente trabajo práctico deberá contar con:

- Equipo PC utilizando Linux como sistema operativo.
- Paquete de compilación de GNU GCC para ARM incluyendo binutils-arm-none-eabi, gcc-arm-none-eabi y make.
- Open OCD
- Eclipse IDE for GNU ARM (versión recomendada Oxygen 3a)
- Placa de desarrollo EDU-CIAA NXP.
- Archivos provistos por el docente.

1. Mi primera aplicación en la CIAA

Para realizar esta actividad deberá contar con un proyecto de base proporcionado por el docente (ej01.tar.gz).

Actividades:

1. Cree una carpeta para realizar todas las actividades de este trabajo práctico. Dentro de ella descomprima el archivo `ej01.tar.gz`. Analice la estructura de su contenido e identifique:
   - Archivo conteniendo la rutina de inicialización (Reset) y vector de interrupciones.
   - *Linker scripts*. Identifique las direcciones de memoria asignadas para el programa (FLASH) y para el stack (RAM). Contraste estos valores con los indicados en la hoja de datos del LPC4337 (archivo UM10503.pdf en el directorio raíz del proyecto). ¿Cuánta memoria ha sigo asignada y cuánta memoria se encuentra libre?.
   - Ubicación del programa principal. Analice que hace el programa.
   - Contenido de la carpeta `lpc`.
   - Archivos makefiles existentes en el proyecto.
   - Localice la carpeta donde se encuentra almacenado el firmware resultante del proceso de compilación, desde allí invoque los siguientes comandos y describa el resultado obtenido:
     - `$arm-none-eabi-size firmware.elf`
     - `$arm-none-eabi-objcopy -O ihex firmware.elf firmware.hex`
     - `$arm-none-eabi-objdump -d firmware.elf`
     - `$arm-none-eabi-objdump -d firmware.elf >firmware.asm`

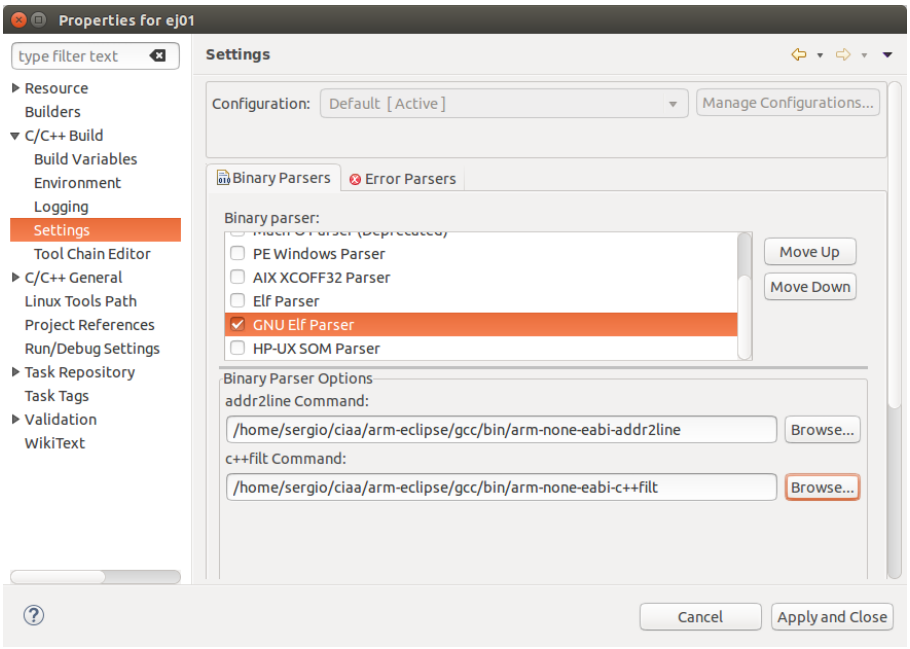

Figura 1: Configuración del analizador binario utilizando las herramientas de desarrollo para ARM Cortex-m4

2. Analice los archivos `makefile` y `config.mk` presentes en el directorio raíz del proyecto.

   a) Describa el contenido de cada uno y su función en el desarrollo del proyecto.

   b) Analice el sentido de la variable `BUILDTYPE`. ¿Cuál es su sentido?

   c) Agregue un objetivo llamado `help` en el `makefile` que indique (muestre por pantalla) la forma de invocar a `make`, los objetivos disponibles y su significado.

3. Desde un terminal construya el proyecto, genere la documentación del mismo, borre el contenido de la memoria de la EDU-CIAA NXP y programe la aplicación. Verifique su funcionamiento.

4. Utilizando el IDE de Eclipse cree un nuevo proyecto (*Make File Project with Existing Code*) y vincúlelo al proyecto existente (*Lenguaje C*, *Toolchain none*).

   Desde las propiedades del proyecto configure el analizador binario (*Binary Parser*) Figura 1.

   Configure también el *path* a los *headers* del compilador utilizado por el sistema (Figura 2) y establézcalo para todas las configuraciones (Figura 3).

   Finalmente compile la aplicación.

5. Inicie una sesión de depuración de la aplicación. Para esto:

   - Verifique que la variable `BUILDTYPE` del archivo `config.mk` tiene establecido el valor '`debug`'.
   - Cree una nueva configuración de depuración desde `Run`/`Debug configurations...` (Figuras 4 y 5) utilizando *GDB OpenOCD Debugging*.
   - Configure la pestaña *Debugger* estableciendo la ruta completa a OpenOCD (si no está seguro, verifique en consola su `path` utilizando el comando `which`).
     En el cuadro *Config Options* agregue el texto `-f openocd/lpc4337.cfg` (Figura 6), esta configuración será un parámetro de `openocd` al momento de ser invocado.

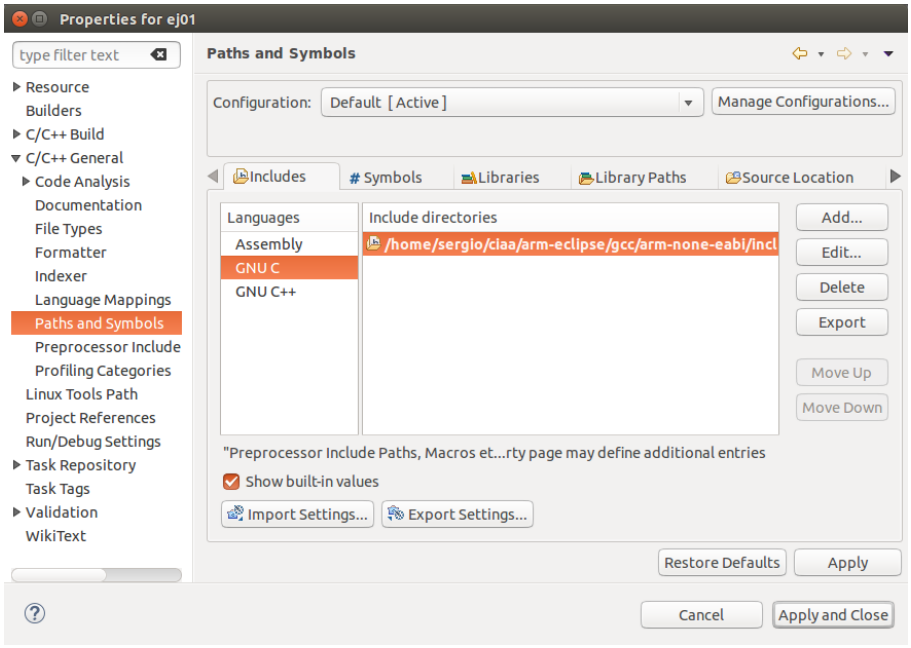

Figura 2: Configuración de las rutas a los headers del compilador utilizado

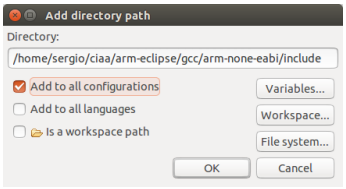

Figura 3: Utilizar los headers indicados en todas las configuraciones.

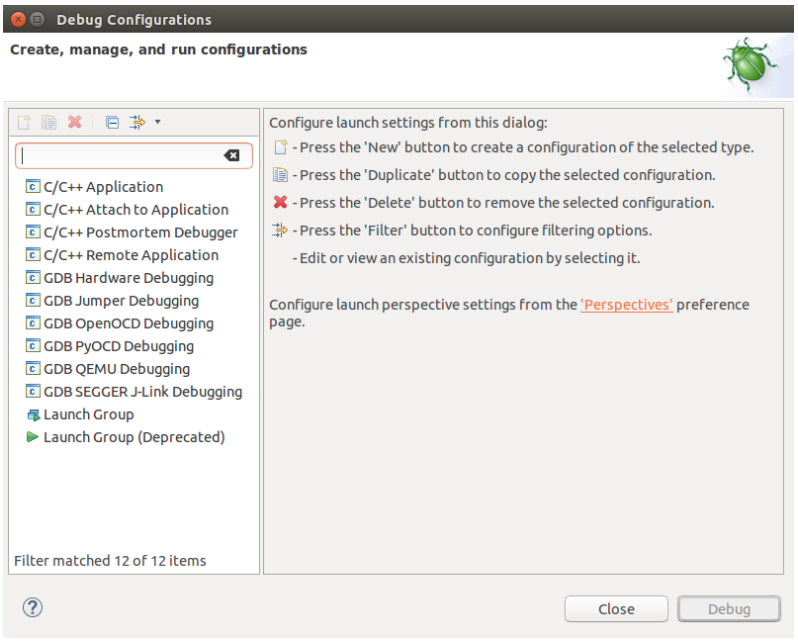

Figura 4: Creación de una nueva configuración de depuración.

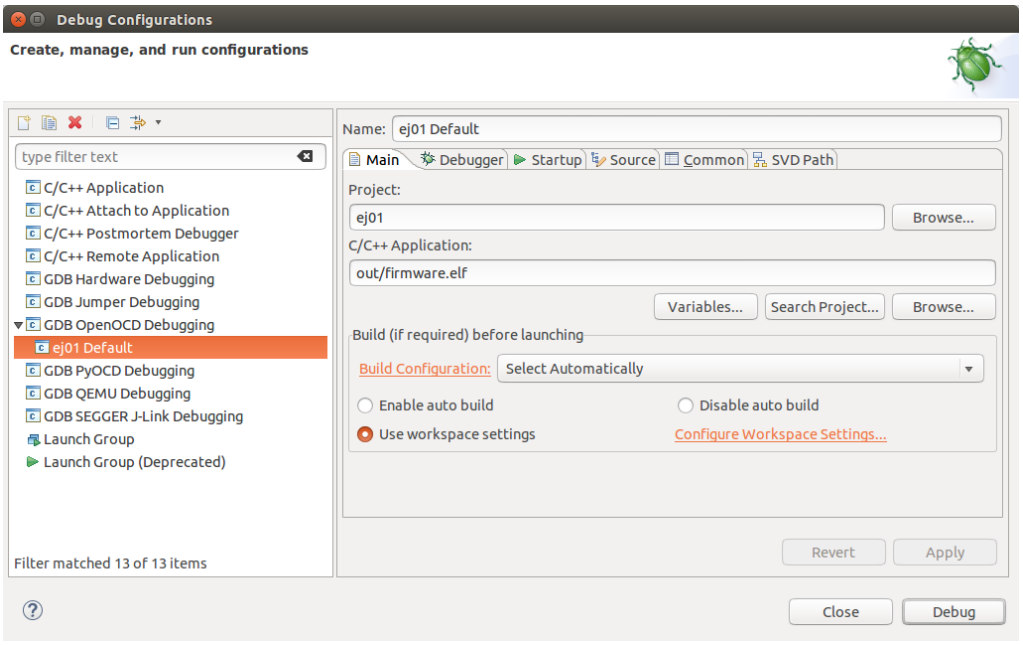

Figura 5: Configuración de la sesión de depuración. Verificar que la casilla aplicación contenga el nombre del binario.

En el cuadro *GDB Client Setup* (Figura 7) establezca la ruta completa al comando `arm-none-eabi-gdb` (deberá encontrarse en la carpeta de aplicaciones de GCC para ARM). Conecte la EDU-CIAA NXP al equipo PC. Finalice la configuración e inicie la depuración (*Apply* and *Debug*). Realice el seguimiento paso a paso por la aplicación.

6. Modifique el código de la aplicación principal agregando un contador, de modo que su valor se incremente por cada iteración y, según si su valor es par o nó, se encienda o apague el led. Verifique el correcto funcionamiento de la aplicación.

7. Inicie una sesión de depuración con la nueva aplicación desarrollada. En ella realice las siguientes operaciones:

   - Determine la dirección de memoria del contador utilizado y contrástela con las direcciones de memoria definidas en el *script de linker*.
   - Desde la vista de registros del microcontrolador identifique el valor que tiene el registro `sp`. Verifique que su valor sea adecuado según la configuración utilizada. ¿A qué conclusión puede llegar a partir del valor observado respecto al consumo de memoria?
   - Utilizando el monitor de memoria observe la dirección donde se almacenó la aplicación (FLASH). Analice el valor de los 4 primeros bytes. ¿Este valor es consistente con la configuración utilizada?

8. Utilizando una EDU-CIAA NXP programada con la aplicación del ejercicio anterior y conectada a su equipo PC realice las siguientes operaciones:

   - Desde un terminal acceda al directorio raíz del proyecto y desde allí ejecute openocd (`openocd -f openocd/lpc4337.cfg`) de modo de iniciar una sesión de depuración.
   - Desde otra terminal o desde una nueva pestaña en la misma terminal establezca una comunicación a OpenOCD utilizando el comando telnet (`telnet localhost 4444`)

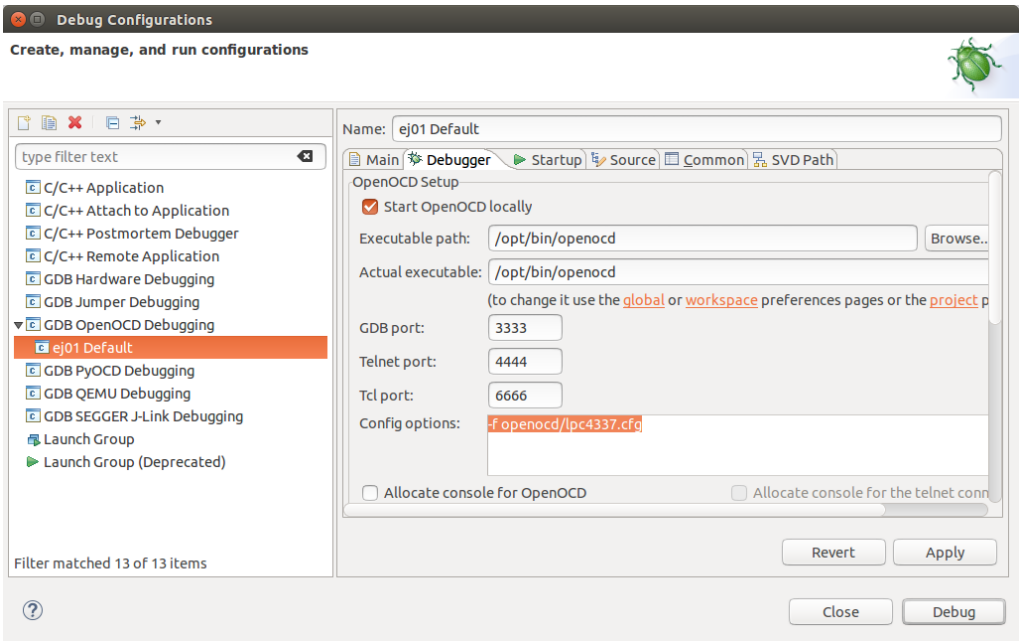

Figura 6: Configuración de OpenOCD

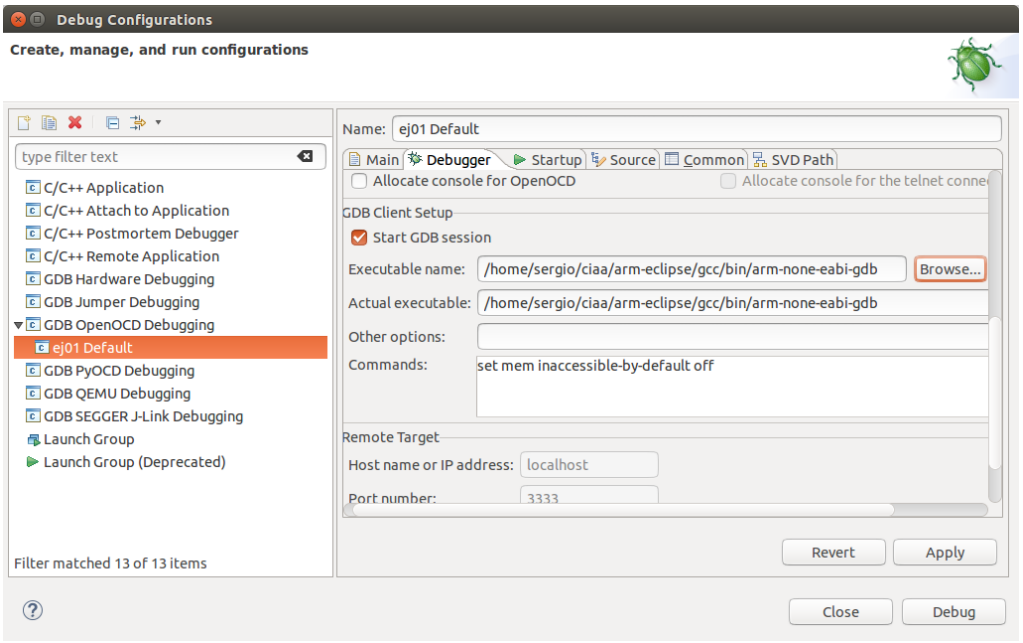

Figura 7: Configuración del depuración cliente `arm-none-eabi-gdb`.

- OpenOCD permite la manipulación de diferentes parámetros de tiempo de ejecución en un sistema embebido. Desde la conexión realizada a través de la terminal envíe los siguientes comandos analizando su sentido a partir de la documentación de OpenOCD (capítulo 15 *General Commands*):
  a) `reset halt`
  b) `reg`
  Observe el valor del registro `sp` y `pc`. ¿Son consistentes estos valores?
  c) `resume`
  Observe la EDU-CIAA NXP, verifique a comenzado a ejecutarse la aplicación.
  d) `halt`
  La aplicación deberá detenerse.
  e) mdw phys 0x1a000000 1
  ¿Que valor observó? ¿Es consistente con la configuración del proyecto?
- `shutdown`

Observe que la terminal la terminal donde ejecutó `OpenOCD` y la finalización de la aplicación.

Documente la funcionalidad de los comandos utilizados así como otros que considere de interés.

9. Cree un nuevo proyecto, a partir del anterior, de modo que la aplicación se compile sin necesidad de recompilar las librerías LPCOpen (considere que la aplicación puede compilarse en modo *debug* o *release* y el proyecto debe enlazarse con las librerías adecuadas)

10. Cree un nuevo proyecto a partir de la actividad anterior de modo que las funciones `hardware.c` y `util.c` sean utilizadas como librerías de enlazado estático.

2. Más aplicaciones!

Tome el proyecto entregado por el docente y genere la documentación asociada a los archivo `util.h` y `hardware.h` ejecutando `make doc`. Analice la documentación generada. A partir de las funciones allí definidas desarrolle aplicaciones para los siguientes enunciados. En cada caso, documente tanto las funciones que cree como el algoritmo de solución y las observaciones que crea pertinentes.

Actividades:

2.1. Cerradura digital

- Cerradura digital:

  Desarrolle una cerradura digital a partir de los recursos disponibles en la EDU-CIAA NXP. El sistema deberá funcionar según los siguientes parámetros:

  - Se utilizarán las teclas presentes en la placa para ingresar las contraseñas, por lo que estas deberán componerse de números conteniendo solo los dígitos del 1 al 4.
  - La cantidad de dígitos de cada contraseña deberá ser 6
  - La contraseña por defecto (establecida al momento de encender la cerradura) deberá ser 123321.
  - Al encender el equipo se asume que la cerradura estará cerrada y se desbloqueará con la clave establecida de fábrica.
  - Una vez abierta la cerradura, los próximos 6 dígitos que se ingresen fijarán la nueva clave.

- El funcionamiento de la cerradura es cíclico, por defecto está cerrada, se desbloquea con la clave de fábrica, espera a que se establezca una nueva clave, se bloquea y espera a ser desbloqueada con la clave establecida.
- Cuando la cerradura está cerrada, al presionarse cada tecla deberá iluminarse durante 500 ms el led asociado. En el caso del led RGB, utilice el color azul.
- Si un código ingresado es incorrecto deberá parpadear 5 veces el led rojo, por el contrario, si es correcto deberá parpadear 5 veces el led verde.
- Luego de establecer una nueva clave todos los leds deberán parpadear 5 veces indicando que la clave se ha establecido.

2.2. Juego de cara o cruz

- Juego cara o cruz:

  Implemente un juego de Cara o Cruz donde el jugador pueda seleccionar una opción y la EDU-CIAA genere aleatoriamente uno de estos valores. Si hay coincidencia el jugador gana o pierde en caso contrario. Considere los siguientes parámetros de funcionamiento:

  - El juego inicia mostrando una secuencia de encendido de leds hasta que el usuario presiona la tecla 2.
  - Al presionar la tecla 2, comienzan a parpadear los leds Rojo y Verde indicando que debe seleccionarse una opción (Rojo = Cara, Verde = Cruz). Al presionar en la tecla 3 o 4, debe seleccionarse el color correspondiente manteniendo el led encendido y apagando el otro.
  - Luego de seleccionar una de las opciones, se deberá generar aleatoriamente un valor de cara o cruz. Para esto el led RGB deberá cambiar cíclicamente entre Rojo y Verde, durante un tiempo aleatorio entre 5 y 10 segundos. El color activo al cabo de finalizar el tiempo es el resultado selección.
  - Finalizado el juego deberá representarse su resultado. En ambos casos los leds activos (el seleccionado por la EDU-CIAA y el elegido por el jugador) deberán parpadear. Si el jugador ganó deberán encender y apagar cada 75 ms, en caso contrario cada 300 ms. Esta situación deberá mantenerse hasta que el usuario presione la tecla 1, momento en el cual comienza nuevamente la secuencia del juego.

  Notas de implementación:

  - Existen diferentes modelos de EDU-CIAA NXP en cuales se ha dado diferente orden a los colores de los leds. En el enunciado anterior se ha considerado que se tiene la disposición de la figura 8. Por esto, considere verificar la distribución de los los colores de los leds en la placa que utilice.
  - El juego utiliza valores aleatorios, considere qué valor utilizar como semilla aleatoria y justifique su elección.
  - Utilizando las herramientas de depuración consideradas en este y en trabajos anteriores, verifique que la secuencia de valores aleatorios generados sea diferente, del mismo modo que la semilla aleatoria. Utilice también estas herramientas para verificar el correcto funcionamiento del sistema.
  - Utilice funciones para resolver las diferentes etapas del juego.

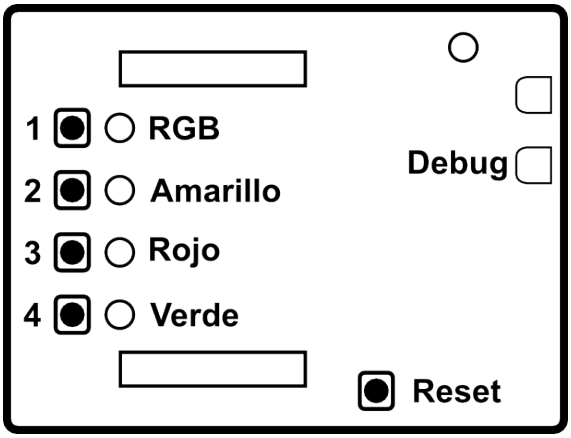

Figura 8: Distribución de leds y teclas consideradas en el enunciad del juego de Cara o Cruz.