Índice

Utilización básica de firmware_v3

1. Conceptos básicos

- **firmware_v3** es un proyecto en base a *makefile* que actúa como framework para el desarrollo de *firm*ware para sistemas embebidos en lenguajes C/C++.
- El ***makefile*** posee una serie de ***targets*** invocables con *make*, los cuales permiten: compilar el programa, descargar el programa a la plataforma, etc.
- Para los IDEs, firmware_v3 se comporta como **un único proyecto, sin embargo está compuesto por múltiples programas** para compilar y descargar a las plataformas.

2. Flujo básico de trabajo con firmware_v3

2.1. Programas existentes dentro del proyecto firmware_v3

Este flujo de trabajo propuesto es válido tanto para los programas de ejemplo provistos, como para programas creados previamente por el usuario en firmware_v3.

Los pasos a seguir son:

1. Seleccionar plataforma de hardware (*board*) a utilizar dentro del proyecto firmware_v3.
2. Seleccionar programa a utilizar dentro del proyecto firmware_v3.
3. Compilar el programa seleccionado.
4. Descargar el programa seleccionado a la plataforma de hardware (*board*) seleccionada.

Tanto *seleccionar el programa* como *la plataforma de hardware* a utilizar se puede realizar de manera **manual** (editando archivos de texto) o mediante el uso de la **interfaz gráfica** (requiere el programa Zenity).

Nota: Si instaló el paquete de software provisto (CIAA Software) o tiene Linux Ubuntu 18.04 o superior ya tiene disponible Zenity.

2.1.1. Seleccionar plataforma de hardware (*board*) a utilizar dentro del proyecto firmware_v3

De forma manual

Para seleccionar la *board* de forma manual se debe:

- Crear un archivo de texto nombrado `board.mk` dentro de la carpeta firmware_v3.
- Definir la variable `BOARD` el el archivo `board.mk` de acuerdo a la plataforma elegida.

Ejemplo de archivo `board.mk`:

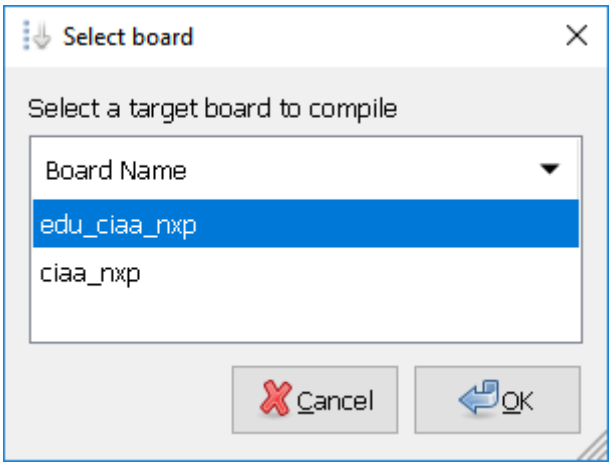
```
BOARD = edu_ciaa_nxp
```

Mediante la interfaz gráfica

Se realiza utilizando el *target* del Makefile de firmware_v3 `select_board` mediante el comando:

`make select_board`

El cual lanza un menú en pantalla que presenta las posibles plataformas de hardware disponibles:



Al seleccionar una de ellas y presionar OK se crea automáticamente un archivo de texto nombrado `board.mk` dentro de la carpeta de firmware_v3 que contiene el nombre de la plataforma de hardware seleccionada.

2.1.2. Seleccionar programa a utilizar dentro del proyecto firmware_v3

De forma manual

Para seleccionar la el programa a utilizar (para compilar, descargar o limpiar) se debe:

- Crear un archivo de texto nombrado `program.mk` dentro de la carpeta firmware_v3.
- Definir en el archivo `program.mk` las variables `PROGRAM_NAME` y `PROGRAM_PATH` de acuerdo a l programa que se desea utilizar (PROGRAM_PATH es relativa a la carpeta de firmware_v3, dejar vacía esa variable si el programa está en la carpeta firmware_v3.

Ejemplo de archivo `program.mk`:

```
PROGRAM_PATH = examples/c
PROGRAM_NAME = app
```

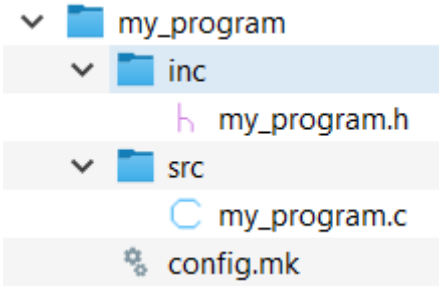
Con este archivo se selecciona el programa nombrado **app** (el nombre de la carpeta que contiene los archivos del programa le da nombre al mismo) dentro de la ruta **firmware_v3/examples/c**.

Mediante la interfaz gráfica

Se realiza utilizando el *target* del Makefile de firmware_v3 `select_program` mediante el comando:

```
make select_program
```

Que lanza un menú en pantalla que permite elegir la carpeta del programa a utilizar:



Al seleccionar la carpeta del programa deseado y presionar OK se crea automáticamente un archivo de texto nombrado `program.mk` dentro de la carpeta de firmware_v3. En la imagen anterior se ve que se selecciona el programa blinky.

Nota: Procure no entrar en la carpeta del programa elegido. Simplemente seleccione la carpeta del programa y presione Ok sin entrar en la carpeta del mismo si no no se ejecutará correctamente este comando.

2.1.3. Compilar el programa seleccionado

Se realiza utilizando el *target* del Makefile de firmware_v3 `all` mediante el comando:

```
make all
```

o simplemente:

```
make
```

ya que es el *target* por defecto.

De esta forma se inicia la compilación cruzada del programa seleccionado para la plataforma seleccionada junto a todas sus dependencias.

2.1.4. Descargar el programa seleccionado a la plataforma de hardware (*board*) seleccionada

Se realiza utilizando el *target* del Makefile de firmware_v3 `download` mediante el comando:

```
make download
```

Se abre OpenOCD y se descarga el programa a la memoria de la plataforma de hardware seleccionada.

2.2. Creación de programas y bibliotecas

2.2.1. Crear un nuevo programa dentro del proyecto firmware_v3

Un programa puede generase manualmente o mediante la interfaz gráfica.

De forma manual

Cada programa consiste en una carpeta (cuyo nombre no puede contener espacios ni caracteres latinos) que incluye al menos 2 subcarpetas, una nombrada `src` (aquí se ponen los archivos de código fuente, .c, .cpp, .ino o .s), y otra nombrada `inc` (aquí se ponen los archivos de cabeceras .h or .hpp).

Ejemplo de programa en lenguaje C nombrado `my_program.c`:

```
#include "sapi.h"
int main( void )
{
   boardInit();
   while(true){
      gpioToggle(LED);
      delay(200);
   }
}
```

Ejemplo de archivo de cabecera `my_program.h`:

```
#ifndef __MY_PROGRAM_H_
#define __MY_PROGRAM_H_
   // Your public declarations...
#endif /*__MY_PROGRAM_H_*/
```

Además se puede definir un archivo nombrado `config.mk`, que sirve para configurar opciones de compilación y bibliotecas a incluir en el programa.

Ejemplo de `config.mk` con valores por defecto:

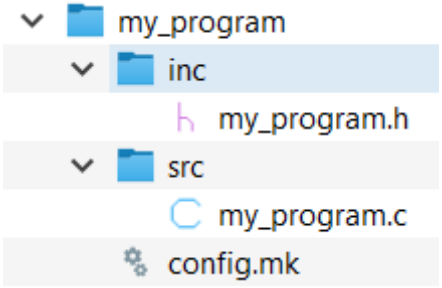
```
# Compilation and libraries default values

# Compile options
VERBOSE=n
OPT=g
USE_NANO=y
USE_LTO=n
SEMIHOST=n
USE_FPU=y
ENFORCE_NOGPL=n

# Libraries
USE_LPCOPEN=y
USE_SAPI=y
```

Nota: Si no define este archivo se tomarán dichos valores por defecto.

Finalmente, la estructura completa de un programa es:



Nota: el archivo `my_program.h` es opcional.

Mediante la interfaz gráfica

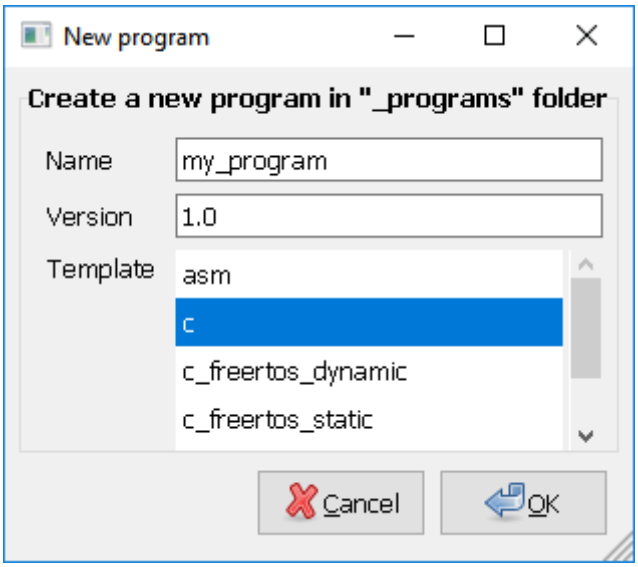
Se realiza utilizando el *target* del Makefile de firmware_v3 `new_program` mediante el comando:

`make new_program`

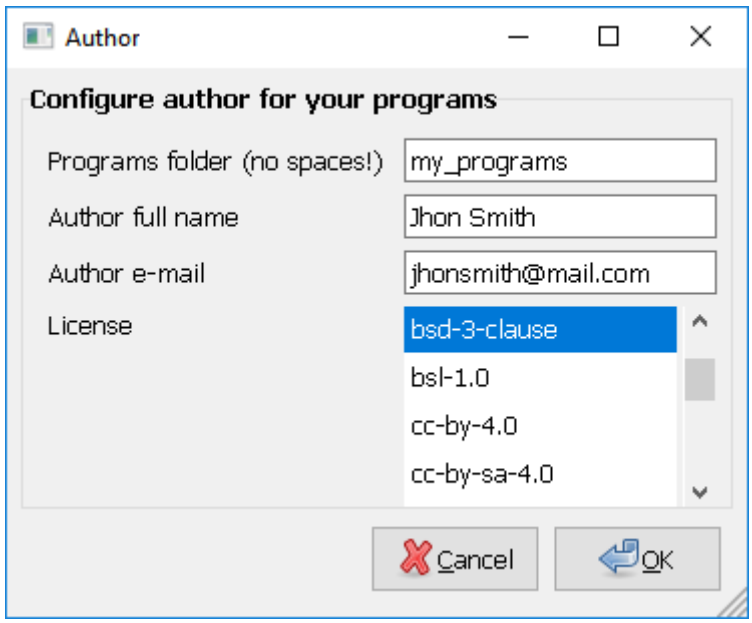
que inicia una ventana donde se puede elegir:

- El nombre del programa (requerido).
- Su versión (opcional).
- La plantilla de programa a utilizar (requerido).

Ejemplo:



Note: La primera vez pedirá al usuario que complete las preferencias de usuario:



Estas preferencias de usuario incluyen:

- Carpeta donde se guardarán los programas generados (sin espacios o caracteres latinos, requerido).
- Nombre completo del autor (requerido).
- Mail del autor (requerido).
- Licencia del programa.

Todos son campos requeridos.

Las preferencias definidas se guardan en el archivo:

`firmware_v3/scripts/user/user_preferences.cfg`

que se puede borrar o editar manualmente si se desean cambiar las mismas.

2.2.2. Crear un nuevo módulo de biblioteca dentro del proyecto firmware_v3

Bibliotecas globales

La carpeta `libs` incluye bibliotecas que pueden utilizarse desde cualquier programa (bibliotecas globales).

El `Makefile` permite compilar dos tipos de bibliotecas:

- **Biblioteca simple**. Consiste en una carpeta (con nombre sin espacios, ni caracteres latinos) que en su interior incluye dos carpetas, una nombrada `src` (aquí se ponen los archivos de código fuente, .c, .cpp, .ino o .s), y otra nombrada `inc` (aquí se ponen los archivos de cabeceras .h or .hpp). Esta clase de bibliotecas son compiladas automáticamente por el Makefile.
- **Bibliotecas avanzadas**. Consisten en bibliotecas con una estructura compleja de archivos y carpetas, por ejemplo, LibUSB. En este caso se requiere crear un makefile propio de la biblioteca nombrado `module.mk`. Se puede ver como ejemplo el makefile de la biblioteca dentro de la biblioteca sAPI.

  Ejemplo de archivo `module.mk` de la biblioteca avanzada **sAPI**:

```
ifeq ($(USE_SAPI),y)

SAPI_BASE=libs/sapi/sapi_v0.5.2

DEFINES+=USE_SAPI

INCLUDES += -I$(SAPI_BASE)/base/inc
INCLUDES += -I$(SAPI_BASE)/soc/core/inc
INCLUDES += -I$(SAPI_BASE)/soc/peripherals/inc
INCLUDES += -I$(SAPI_BASE)/soc/peripherals/usb/device/inc
INCLUDES += -I$(SAPI_BASE)/soc/peripherals/usb/host/inc
INCLUDES += -I$(SAPI_BASE)/board/inc
INCLUDES += -I$(SAPI_BASE)/abstract_modules/inc

SRC += $(wildcard $(SAPI_BASE)/base/src/*.c)
SRC += $(wildcard $(SAPI_BASE)/soc/core/src/*.c)
SRC += $(wildcard $(SAPI_BASE)/soc/peripherals/src/*.c)
SRC += $(wildcard $(SAPI_BASE)/soc/peripherals/usb/device/src/*.c)
SRC += $(wildcard $(SAPI_BASE)/soc/peripherals/usb/host/src/*.c)
SRC += $(wildcard $(SAPI_BASE)/board/src/*.c)
SRC += $(wildcard $(SAPI_BASE)/abstract_modules/src/*.c)

# External Peripherals
include $(SAPI_BASE)/external_peripherals/module.mk

endif
```

Bibliotecas locales a un programa

Para las bibliotecas dentro de un programa de usuario específico tenemos también las mismas dos opciones, bibliotecas *básicas* y *avanzadas*. En ambos casos se crean de la misma forma que las bibliotecas globales. El único cambio es que en el caso de ser una biblioteca básica se guardan los archivos que la componen en las subcarpetas `inc` y `src` dentro de la carpeta del programa a la cual pertenece, y en el caso de una biblioteca avanzada, se deben agregar al archivo `config.mk` del programa las rutas (*paths*) de los archivos de esta nueva biblioteca.

3. Lista completa de *Targets* disponibles

A continuación se listan todos los *targets* disponibles en firmware_v3:

***Targets* principales**

- `all` Compilar programa seleccionado.
- `clean` Eliminar archivos de compilaciones previas (remueve la carpeta *out* dentro del programa seleccionado). Es necesario ejecutarlo sobre le programa seleccionado al cambiar opciones de compilación del programa (archivo *config.mk*) o al cambiar la plataforma de hardware (*board*) para la cual se compilará el programa.
- `clean_all` Eliminar archivos de compilaciones previas de todos los programas dentro de la carpeta firmware_v3.
- `download` Descarga el programa seleccionado a la *board* seleccionada.
- `erase` Borra la flash de la *board* seleccionada. Es necesario resetear la plataforma luego de aplicarlo.
- `new_program` Permite gráficamente crear un nuevo programa dentro de la carpeta seleccionada por el usuario (la primera vez se pide el nombre de la carpeta donde se guardaran los programas y se crea dentro de la carpeta firmware_v3).
- `select_board` Permite elegir gráficamente la plataforma de hardware (*board*) para la cual se compilará el programa.
- `select_program` Permite elegir gráficamente el programa a utilizar (compilar, descargar, limpiar, etc.).
- `size` Muestra la ocupación del programa compilado para el programa seleccionado en las distintas áreas de memoria de la *board* seleccionada.

Otros *targets*

- `debug` Depurar con Embedded IDE (usa los *targets .debug* y *.run*, se utiliza con Embedded IDE y junto a gdbFront. **Advertencia:** es una característica en desarrollo.
- `.info` Información de la compilación.
- `.axf_to_bin` Convertir archivo *.axf* a *.bin*.
- `.elf_to_bin` Convertir archivo *.elf* a *.bin*.
- `.debug` Depurar con Embedded IDE (*debug*).
- `.run` Depurar con Embedded IDE (*run*).
- `.hardware_test` Ejecutar tests en hardware.
- `.test_build_all` Ejecuta un test que comprueba que todos los programas compilen de forma correcta. **Advertencia:** que compilen no significa que funcionen de la forma esperada.
- `.enforce_no_gpl` : Chequea que no haya código (L)GPL en tu proyecto.

Más información

Volver al README.