UNIVERSIDAD
COMPLUTENSE
MADRID

Proyecto de Innovación

Convocatoria 2016/2017

Proyecto Nº 5

Framework libre para el desarrollo de aplicaciones en el escenario de “Internet of Things”

Alberto Antonio Del Barrio García

Facultad de Informática

Arquitectura de Computadores y Automática (Arquitectura y Tecnología de Computadores e Ingeniería de Sistemas y Automatización)

1. Objetivos propuestos en la presentación del proyecto:

Inicialmente planteamos los siguientes objetivos:
- Desarrollo de un entorno gráfico de programación y depuración unificado para placas de desarrollo.
- Desarrollo de un entorno de depuración basado en OpenOCD, como el que se utiliza en diversas asignaturas impartidas en la Facultad de Informática.
- Desarrollo de un entorno basado en herramientas open-source, por tanto, accesible a todos los alumnos.
- Desarrollo de un entorno multiplataforma, es decir, capaz de funcionar en diversos sistemas operativos.
- Preparación de un instalador para facilitar la puesta en marcha de la nueva herramienta.
- Escritura de varias guías de usuario para configurar cada tipo de proyecto en función de la placa objetivo.
- Acoplar las prácticas propuestas en la ficha docente con el uso del nuevo framework.
- Mejorar el aprendizaje del alumnado mediante el entendimiento de la configuración de cada tipo de proyecto. Comprender por qué hay que seleccionar un tipo de procesador/arquitectura/alineamiento en memoria, etc. reforzará estos conceptos en los estudiantes.
- Generación de alguna publicación de carácter educativo basada en el sistema propuesto.

2. Objetivos alcanzados:

De manera general podemos considerar que todos los objetivos han sido alcanzados, salvo el estudio del impacto del proyecto en el aprendizaje de los alumnos, que deberá probarse en el curso académico siguiente. Análogamente, la publicación de carácter educativo deberá esperar al curso siguiente para poder contar con datos de satisfacción por parte del alumnado. A continuación enumeramos los hitos del proyecto:

- Desarrollo de un entorno gráfico de programación y depuración unificado para placas de desarrollo.
- Desarrollo de un entorno de depuración basado en OpenOCD, como el que se utiliza en diversas asignaturas impartidas en la Facultad de Informática.
- Desarrollo de un entorno basado en herramientas open-source, por tanto, accesible a todos los alumnos.
- Desarrollo de un entorno multiplataforma, es decir, capaz de funcionar en diversos sistemas operativos.
- Escritura de varias guías de usuario para configurar cada tipo de proyecto en función de la placa objetivo.
- Acoplar las prácticas propuestas en la ficha docente con el uso del nuevo framework. Las prácticas de la ficha docente son compatibles con el nuevo framework.
- En su lugar del instalador se han desarrollado guías de preparación del entorno dependiendo del sistema operativo. Asímismo, se proporciona una máquina virtual basada en Ubuntu 14.04 con el entorno instalado y con prácticas guía.

3. Metodología empleada en el proyecto:

La metodología seguida durante el desarrollo de este Proyecto de Innovación Docente fue la siguiente:

- Selección de placas de desarrollo: dado que el presupuesto recibido fue de 0€, se ha trabajado con las placas disponibles: S3CEV40 y Discovery STM32f429.

- Estudio para la selección de componentes necesarios en el diseño del sistema. A la hora de construir el nuevo entorno es crítico seleccionar las toolchain adecuadas para la compilación y depuración de los proyectos. Se utilizará como criterio de selección su adecuación a las familias de procesadores ARM, que son la base de la mayor parte de sistemas empotradosimplementados en el área del “Internet of Things”.

- Diseño e implementación del sistema completo.

- Desarrollo de un programa “Hola Mundo” para cada una de las placas objetivo.

- Test de conexión con cada una de las placas objetivo.

- Depuración de cada programa “Hola Mundo” en las placas objetivo.

- Preparación de guías de usuario para configurar los proyectos en función de la placa objetivo.

- Preparación de prácticas para utilizar el entorno desarrollado.

4. Recursos humanos:

Alberto A. Del Barrio es un joven Doctor que defendió la tesis en 2011. Desde 2012 es Profesor Ayudante Doctor en la Facultad de Informática. Ha sido profesor y coordinador de Sistemas Empotrados Distribuidos (SED), perteneciente al Máster en Informática e implantada por él en el curso 2013/2014. Asimismo, cuenta con varias publicaciones docentes y evaluaciones positivas por el sistema Docentia. Durante el curso 2013/2014 fue el responsable del PIMCD nº 18, que cuenta ya con más de 2300 descargas. Además ha participado en otros dos PIMCDs.

Carlos Roa es técnico especialista II en nuevas tecnologías del departamento de Arquitectura de Computadores y Automática. Ha colaborado en 4 PIMCD: nº 50 (2014-2015), PIMCD2007/204 (2007-2008 y su posterior ampliación 2008-2009) y "Diseño de prácticas reales para un laboratorio unificado la asignatura de ISA" (2006-2007).

Joaquín Recas es profesor de la Facultad de informática desde 2005 y Contratado Doctor desde 2013. Ha recibido 8 evaluaciones positivas desde 2008 y participado en 3 proyectos de innovación docente desde 2005. Centra sus esfuerzos de investigación en los sistemas empotrados y tratamiento de señales en tiempo real. También ha participado en 7 proyectos con empresa bajo el amparo del Art. 83.

Guillermo Botella es Profesor Contratado Doctor por la Facultad de Físicas. Ha participado como profesor del Máster de Formación de Profesorado de Secundaria (especialidad Informática) desde su inicio (curso 2009-2010) y durante todas las ediciones. Ha participado en 5 PIMCDs, posee 7 publicaciones docentes y atesora 4 evaluaciones positivas (Programa Docentia) durante los cursos recientes. También es profesor SED.

Christian Tenllado es profesor de la Facultad de Informática desde 2003 y Profesor Contratado Doctor desde 2005. Ha participado en tres Proyectos de Innovación Docente entre los años 2012-2014 y cuenta con una publicación docente. También ha sido profesor del título propio Master en Desarrollo de Videojuegos de la Facultad de Informática durante 6 cursos. Desde el curso 2012/2013 ha tenido 7 evaluaciones positivas por el programa Docentia. Además, ha participado en 9 proyectos de transferencia tecnológica.

Luis Piñuel es Profesor Titular de la Facultad de Ciencias Físicas desde 2006. Ha participado en 6 PIMCDs entre los años 2007-2013 y cuenta con una publicación docente. Ha puesto en marcha y coordinado diversas asignaturas de laboratorio/prácticas tanto de la Facultad de Informática como de la Facultad de Ciencias Físicas. Además, ha participado en 8 proyectos de transferencia tecnológica, 7 de los cuales en calidad de investigador responsable. Ha sido Secretario Académico de la Facultad de Ciencias Físicas desde junio de 2010 a abril de 2015 y actualmente co-dirige un proyecto del Plan Nacional (TIN2015-65277-R).

5. Desarrollo de las actividades:

Como se ha mencionado anteriormente, se han conseguido prácticamente todos los objetivos con la salvedad del despliegue del framework en una asignatura y del artículo educativo. Además, debido a la falta de financiación, las guías de configuración tan solo se han desarrollado para las placas disponibles: S3CEV40 y Discovery STM32f429.

Pese a estos impedimentos, presentamos una serie de guías para customizar el entorno de desarrollo en tres sistemas operativos: Windows, Linux y Mac OS. Asímismo, se presentan las guías de creación y configuración de proyecto para las placas objetivo.

Por último, se ha preparado la imagen de una distribución de Ubuntu 14.04 (Ubuntu64ARM.ova) con el framework instalado. Dicha imagen está preparada para el motor de virtualización VirtualBox (que es gratuito).

Asímismo, se proporcionan dos prácticas de prueba dentro del workspace:

- discoTest. Esta práctica está desarrollada sobre la STM32f429 y su propósito es encender y apagar un led a través del botón.
- maletinTest. Esta práctica está desarrollada sobre la S3CEV40 y su propósito es mostrar un contador ascendente/descendente módulo 6, usando los botones y el display 7-segmentos.

Los proyectos están configurados y las depuraciones preparadas. Para lanzar una depuración habría que ejecutar primero el openocd (con la configuración correspondiente) y, a continuación, seleccionar el perfil de depuración asociado al proyecto.

Notas:

Usuario: arm

PWD: arm1234

Directorio de eclipse: $HOME/eclipse

Directorio de workspace: $HOME/workspace

Directorio de las herramientas instaladas: /usr/local/gnuarmeclipse

Toolchains: versión 5.4 de gcc-arm-none-eabi y 0.10 de openocd.

6. Anexo:

Se presentan varios anexos con la guía de instalación del framework en distintos sistemas operativos, así como la configuración de proyectos para las placas disponibles.

Nota: debido a la incompatibilidad del visor de registros para la S3CEV40 (aka *maletín del ARM*) con Eclipse Neon, el framework se basa en Eclipse Mars.

6.1) Instalación en Windows:

Probado en Windows 7 x64:

1) Instalar JDK 1.8

2) Instalar Eclipse Mars para C/C++

3) Instalar gcc-arm-none-eabi-4_9-2015q2-20150609-win32

3.1) Añadir ruta al path

4) Instalar gnu make tools (make-3.81.exe)

4.1) Añadir ruta al path

5) Instalar gnu coreutils (coreutils-5.3.0.exe)

6) Instalar libusb (usar st-link_v2_usbdriver.exe)

7) Dentro de Eclipse:

7.1) Instalar GNU ARM Eclipse Plugins (http://gnuarmeclipse.sourceforge.net/updates) C/C++ Cross development tools

7.2) Instalar el visor de registros (Eclipse Marketplace, EmbSysRegView 0.2.5.)

7.2.1) En ${eclipse_home}\plugins\org.eclipse.cdt.embsysregview.data_0.2.5.r180\data\armv7 crear la carpeta "Samsung" y copiar ahí el fichero s3c44b0x.xml

Editar el archivo y cambiar la 2ª línea por

<!DOCTYPE model SYSTEM "../../embsysregview.dtd">

Para el maletín:

8) Instalar drivers olimex (como especifica en el primer video la primera vez que se conecta el maletín)

https://www.youtube.com/watch?v=3TGzKmU3Tgo&feature=youtu.be

https://www.youtube.com/watch?v=ySCvoDzABZM&feature=youtu.be

https://www.youtube.com/watch?v=sAic6oN3pFA&feature=youtu.be

6.2) Instalación en Linux

Probado sobre Ubuntu 14.04

1) Instalar JDK 1.8

2) Instalar Eclipse Mars C/C++

3) sudo apt-get -y install lib32z1 lib32ncurses5 lib32bz2-1.0

4) Descargar el último tarball de Launchpad con el arm-none-eabi-gcc y demás

Descomprimir el tarball e instalar.

Por ejemplo en: /usr/local/

cd /usr/local

sudo tar xjf ~/Downloads/gcc-arm-none-eabi-4_9-2015q3-20150921-linux.tar.bz2

El resultado debería ser un directorio como /usr/local/gcc-arm-none-eabi-4_9-2015q3

Para comprobar si el compilador es funcional:

/usr/local/gcc-arm-none-eabi-4_8-2014q1/bin/arm-none-eabi-gcc --version

arm-none-eabi-gcc (GNU Tools for ARM Embedded Processors) 4.8.3 20140228 (release)

5) Dentro de Eclipse ir a Eclipse Marketplace:

5.1) Instalar GNU ARM Eclipse (Si no funciona, descargar de sourceforge e instalar manualmente)

Para más información pueden consultarse:

http://gnuarmeclipse.github.io/plugins/install/

http://gnuarmeclipse.github.io/toolchain/install/

http://gnuarmeclipse.github.io/debug/install/

6) Instalar openocd

6.1) Descargar el último tarball de:

https://github.com/gnuarmeclipse/openocd/releases

Para más información puede consultarse:

http://gnuarmeclipse.github.io/debug/openocd/

7) Instalar el driver de stlink para la Discovery

sudo apt-get install libudev-dev libudev1

Importante: Instalar libusb-1.0-9-dev o posterior (descargar libusb-1.0.21-dev de sourceforge)

https://sourceforge.net/projects/libusb

Descargar el driver stlink de:

https://github.com/texane/stlink

e instalar siguiendo los pasos especificados en:

https://github.com/texane/stlink/blob/master/doc/compiling.md

Importante: copiar fichero de reglas de etc/udev/rules.d en /etc/udev/rules.d y ejecutar como root (o reiniciar)

udevadm control --reload-rules

udevadm trigger

Con esto ya tendríamos todo para la Discovery.

Para el maletín del ARM:

8) Instalar los drivers para el olimex.

sudo apt-get install libftdi-dev libftdi1

Crear fichero de reglas olimex-arm-usb-ocd-h.rules /etc/udev/rules.d y copiar el siguiente contenido

PID: 0x002b

VID: 0x15ba

SUBSYSTEM="usb", ACTION="add", ATTRS{idProduct}=="002b", ATTRS{idVendor}=="15ba", MODE="660", GROUP="plugdev", RUN+="/bin/setfacl -m u:'NOMBRE USUARIO':rw- /dev/$name"

Dentro de la carpeta del openocd, si no existieran copiar los ficheros: embest_s3cev40.cfg (al directorio scripts/board), samsung_s3c44b0x.cfg (a scripts/target) y arm-fdi-ucm.cfg a scripts/test

Editar arm-fdi-ucm.cfg para corregir la localización de la interfaz ftdi (seguramente en interface/ftdi)

9) Instalación del visor de registros

9.1) Al igual que en Windows, instalar el visor de registros (Eclipse Marketplace, EmbSysRegView 0.2.5.)

9.2) En ${eclipse_home}\plugins\org.eclipse.cdt.embsysregview.data_0.2.5.r180\data\armv7 crear la carpeta "Samsung" y copiar ahí el fichero s3c44b0x.xml

Editar el archivo y cambiar la 2ª línea por

<!DOCTYPE model SYSTEM "../../embsysregview.dtd">

6.3) Instalación en Mac OS

Probado en Mac OS Sierra (v10.12.5)

1) Instalar JDK 1.8

2) Instalar Eclipse Mars C/C++

3) sudo apt-get -y install lib32z1 lib32ncurses5 lib32bz2-1.0

4) Descargar el último tarball de Launchpad con el arm-none-eabi-gcc y demás

Descomprimir el tarball e instalar.

Por ejemplo en: /usr/local/

cd /usr/local

sudo tar xjf ~/Downloads/gcc-arm-none-eabi-4_9-2015q3-20150921-linux.tar.bz2

El resultado debería ser un directorio como /usr/local/gcc-arm-none-eabi-4_9-2015q3

Para comprobar si el compilador es funcional:

/usr/local/gcc-arm-none-eabi-4_8-2014q1/bin/arm-none-eabi-gcc --version

arm-none-eabi-gcc (GNU Tools for ARM Embedded Processors) 4.8.3 20140228 (release)

5) Dentro de Eclipse ir a Eclipse Marketplace:

5.1) Instalar GNU ARM Eclipse (Si no funciona, descargar de sourceforge e instalar manualmente)

Para más información pueden consultarse:

http://gnuarmeclipse.github.io/plugins/install/

http://gnuarmeclipse.github.io/toolchain/install/

http://gnuarmeclipse.github.io/debug/install/

6) Instalar openocd por medio de port

sudo port install openocd

Con esto ya tendríamos todo para la Discovery.

Para el maletín del ARM:

7) Editar los ficheros de configuración

Dentro de la carpeta del openocd, si no existieran copiar los ficheros: embest_s3cev40.cfg (al directorio scripts/board), samsung_s3c44b0x.cfg (a scripts/target) y arm-fdi-ucm.cfg a scripts/test

Editar arm-fdi-ucm.cfg para corregir la localización de la interfaz ftdi (seguramente en interface/ftdi)

8) Instalación del visor de registros

8.1) Al igual que en Windows, instalar el visor de registros (Eclipse Marketplace, EmbSysRegView 0.2.5.)

8.2) En ${eclipse_home}\plugins\org.eclipse.cdt.embsysregview.data_0.2.5.r180\data\armv7 crear la carpeta "Samsung" y copiar ahí el fichero s3c44b0x.xml

Editar el archivo y cambiar la 2ª línea por

<!DOCTYPE model SYSTEM "../../embsysregview.dtd">

6.4) Configuración del entorno de trabajo

Maletín ARM:

==========

Crear proyecto:

1) New C Project

1.1) Executable-> Empty Project-> Cross ARM GCC

1.2) Toolchain-> GNU Tools for ARM Embedded Processors (arm-none-eabi-gcc)

1.3) Toolchain path-> 'Ruta del arm-none-eabi-gcc'

Configurar proyecto:

Project->Settings

1) Target processor:

1.1) ARM family-> arm7tdmi

1.2) Architecture-> armv4t

1.3) Instruction set-> ARM (-marm)

1.4) Endianness-> Little endian (-mlittle-endian)

1.5) Float ABI-> Library (soft)

1.6) Unaligned access-> Toolchain default

2) Linker->General

2.1) Desmarcar-> Remove unused sections (-Xlinker --gc-sections)

2.2) Marcar-> Do not use standard start files (-nostartfiles)

2.3) General-> Añadir ld_script

3) Añadir commonEclipse a las opciones Compiler y Assembler

Debug:

0) Configurar openocd

0.1) External Tools Configurations-> New

0.2) Location-> 'Ruta del openocd'

0.3) Arguments-> -f 'Ruta del fichero de configuración'

1) Debugger

1.1) GDB Command-> arm-none-eabi-gdb

1.2) JTAG Device-> Generic TCP/IP

1.2.1) Port number-> 3333

1.3) Using Legacy GDB Hardware Debugging Launcher

2) Startup

2.1) Desmarcar "Reset and Delay"

2.2) Desmarcar "Halt"

2.3) Escribir en el cuadro de texto: monitor reset init

STM32F429:

==========

Crear proyecto:

1) New C Project

    1.1) Executable-> STM32F4XX C/C++

    1.2) Target processor settings

        1.2.1) Chip family-> STM32F429xx

        1.2.2) Content-> Empty

Debug:

0) Configurar openocd (igual que para el maletín, pero cambiando la ruta del fichero de configuración)

El resto es igual que para el maletín.