DESARROLLO DE SOFTWARE PARA SISTEMAS EMPOTRADOS

Práctica 1: Instalación y configuración del entorno de desarrollo

Héctor Pérez
Michael González

Presentación

- Contacto
  - Grupo de trabajo: Ingeniería Software y Tiempo Real
  - Despacho: número 3053 (3ª planta)
  - Correo electrónico: perezh@unican.es
- Las prácticas representan el 30% de la nota
- Evaluación continua en el **laboratorio**
  - trabajo realizado en cada sesión de prácticas
  - presentación del trabajo realizado en un **informe**
    - conocimientos técnicos, claridad, precisión
    - respuesta a las preguntas planteadas
    - uso de capturas de pantalla

Índice

Introducción a la plataforma

- Plataforma hardware empotrada ***Lego Mindstorms EV3***
  - Procesador ARM9 a 300 Mhz
  - 16 MB de memoria Flash
  - 64MB de memoria RAM
  - Lector de tarjetas microSD
  - Pantalla de 178x128 pixels
  - Altavoces
  - Comunicaciones vía USB o Bluetooth
  - Batería recargable de 2050mAh
  - 4 puertos de adquisición de datos (entrada)
  - 4 puertos de ejecución de comandos (salida)

Introducción a la plataforma

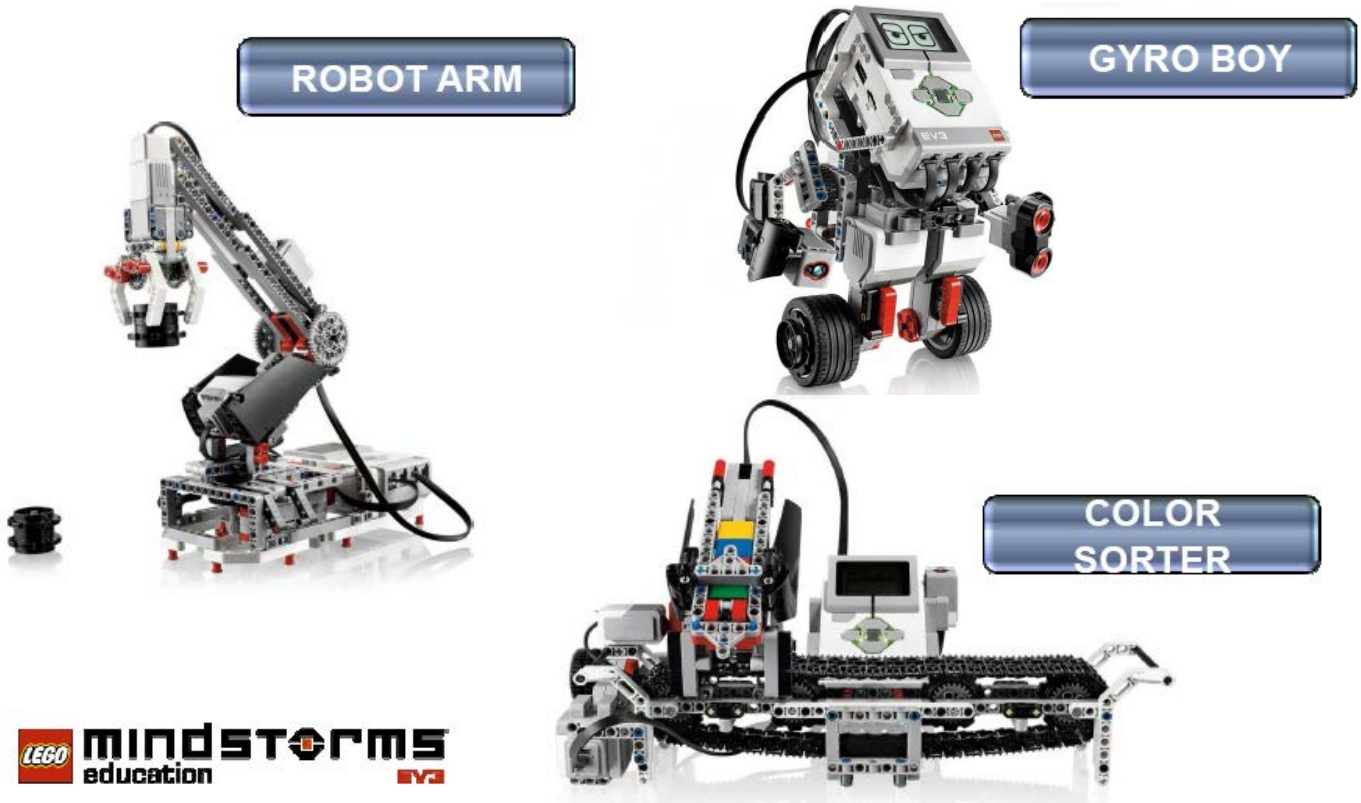
- Interacción directa con el hardware
  - Funcionamiento de los dispositivos estándares (sensores de color, giróscopo, ultrasonidos, etc) y no estándares (cámara digital)
  - Distintas tecnologías para la comunicación:
    - bus I2C, señales analógicas, comunicación serie, etc

Introducción a la plataforma

- Sistema operativo ***EV3Dev***
  - No soportado oficialmente
  - Sistema en continuo desarrollo por la comunidad
  - Características generales
    - Entorno **Linux** para sistemas empotrados
    - Interfaz estándar POSIX

  - Características de tiempo real
    - Sistema no orientado a tiempo real estricto
      - soporta concurrencia, políticas de planificación, protocolos de sincronización, etc.

Introducción a la plataforma



Objetivos

- Instalación del sistema operativo ***EV3Dev*** *en LEGO*
  - mediante un sistema de arranque dual
- Configuración de la interfaz de comunicación
  - entre el sistema de desarrollo y el sistema empotrado
- Ejecución de aplicaciones básicas
- Diferenciar los modos de desarrollo en programación de sistemas empotrados
  - compilación nativa
  - compilación cruzada
- Depuración de aplicaciones

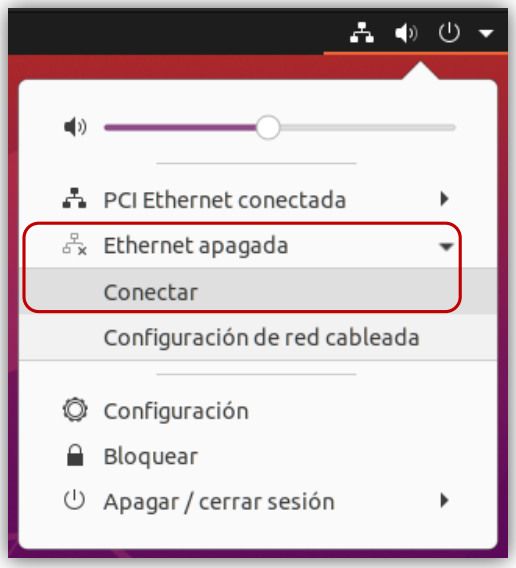
Instalación del sistema operativo ev3dev

- Descargar la imagen del sistema operativo de *moodle*
- Copiar la imagen con el sistema operativo a la tarjeta SD
  - descomprimir la imagen del kernel proporcionada en moodle en el directorio /tmp
  - realizar la copia por el terminal con la herramienta *balena-etcher*:

    sudo balena-etcher /tmp/file.img
- Arrancar el sistema operativo y explorar su menú gráfico
  - advertencia: **no** actualizar el ev3dev a versiones más recientes del kernel
- Conectar el sensor de contacto
  - comprobar que funciona correctamente a través del menú gráfico

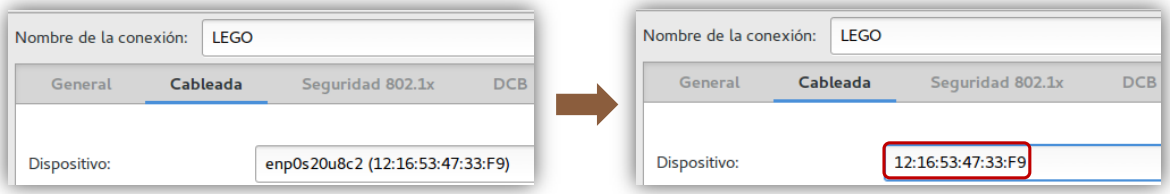
Configuración de la interfaz de comunicación (1/3)

- Arrancar completamente el sistema operativo *ev3dev* en el LEGO
- Conectar el host y el LEGO mediante el cable USB
  - crear una conexión nueva entre ambos sistemas con el comando nm-connection-editor
    - si no aparece automáticamente la conexión puede ser necesario activarla manualmente



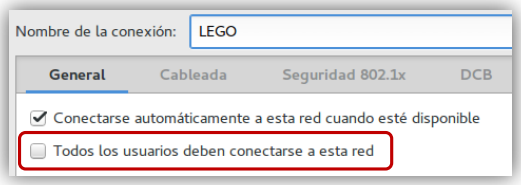
Configuración de la interfaz de comunicación (2/3)

- Conectar el host y el LEGO mediante el cable USB
  - crear una conexión nueva entre ambos sistemas con el comando nm-connection-editor
    - seleccionar la MAC correspondiente al LEGO
    - algunas versiones de Ubuntu cambian el nombre del dispositivo en función del puerto usb al que esté conectado
      - se recomienda usar la MAC como nombre directamente

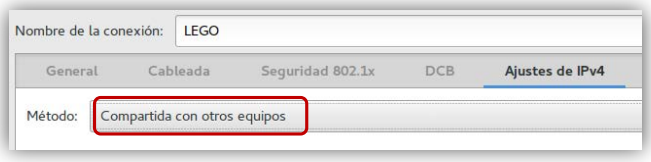


Configuración de la interfaz de comunicación (3/3)

- Conectar el host y el LEGO mediante el cable USB
  - crear una conexión nueva entre ambos sistemas con el comando nm-connection-editor
    - indicar que la conexión **únicamente** esté disponible para vuestro usuario



    - seleccionar conexión compartida en los ajustes de IPv4

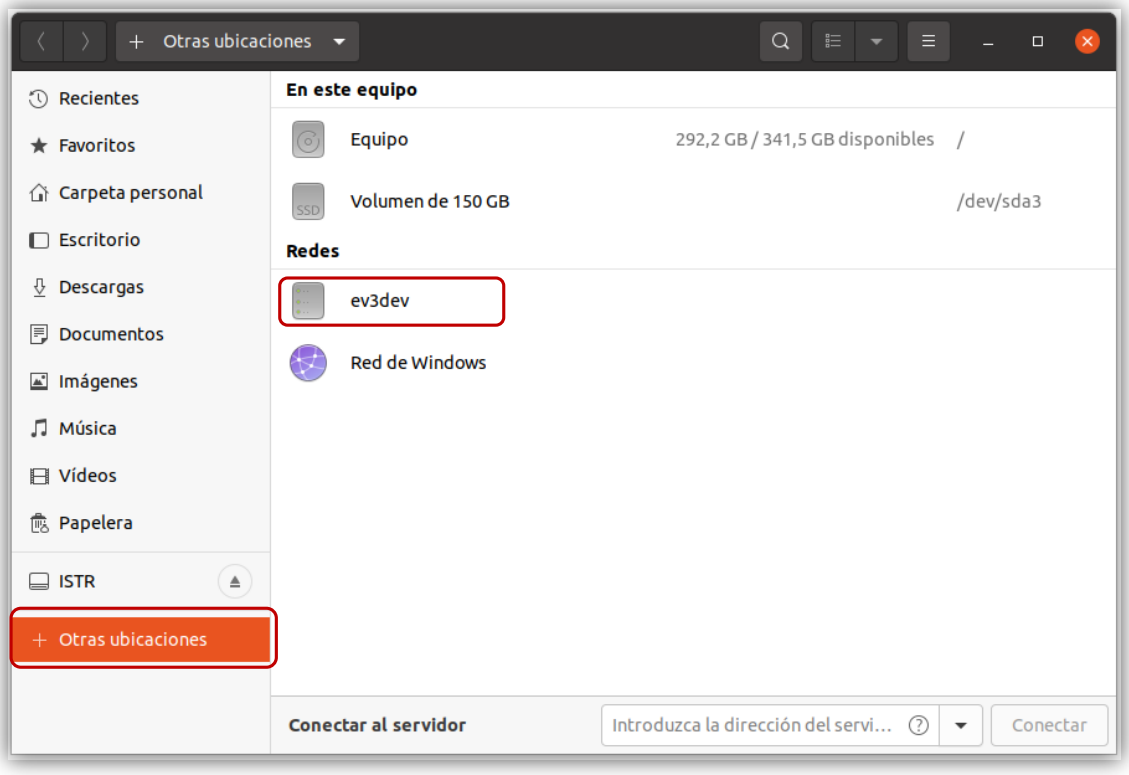


Acceso remoto al sistema empotrado (1/2)

- Configurar el host para:
  - conectarse a través de *ssh* al usuario *robot*
    - dado que es una operación que se repetirá varias veces a lo largo del curso, puede ser interesante automatizar la conexión remota
      - p.ej, utilizando la autenticación basada en llave pública-privada
      - no debe alterarse la configuración del *host*
  - acceder gráficamente al sistema de ficheros del LEGO
    - manualmente, introduciendo en el navegador de archivos

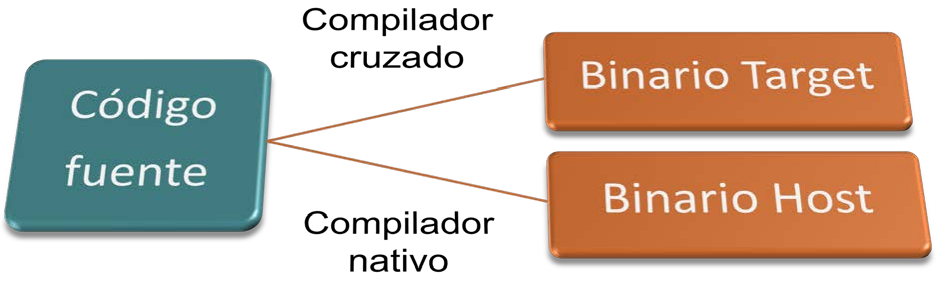
      sftp://robot@IP/home/robot
    - automáticamente

Acceso remoto al sistema empotrado (2/2)



Compilación nativa y Compilación cruzada

- Diferenciar los modos de desarrollo en programación de sistemas empotrados
  - compilación nativa
    - se desarrolla en el mismo computador que se ejecuta
  - compilación cruzada
    - se desarrolla en un computador y se ejecuta en otro



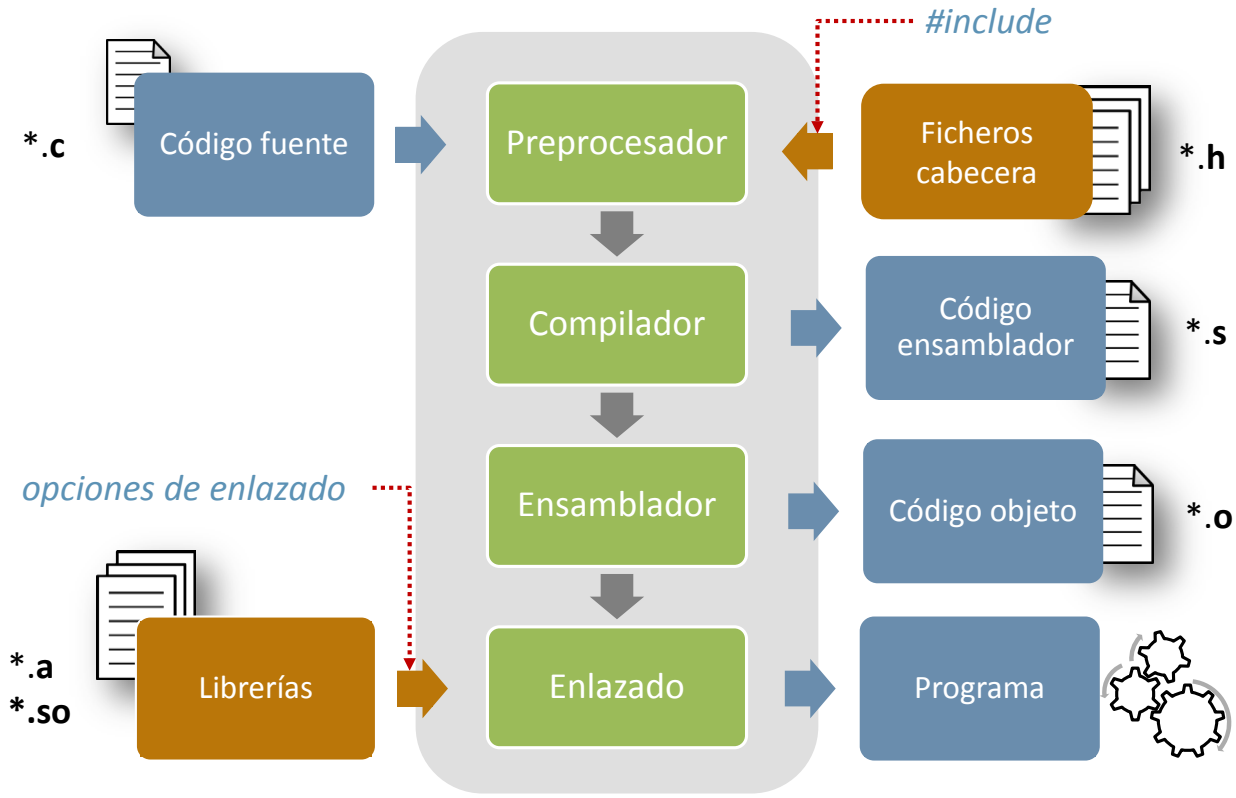
Configuración del entorno para la compilación nativa

- Instalación del entorno de desarrollo nativo en ev3dev
  - incluido en el paquete *build-essential*

    sudo apt-get update

    sudo apt-get install build-essential
- Mientras termina la instalación del entorno de desarrollo nativo:
  - anotar la versión del kernel
  - anotar el valor de los siguientes flags de configuración que se encuentran en el fichero de configuración del kernel dentro del directorio /boot
    - CONFIG_PREEMPT
    - CONFIG_HIGH_RES_TIMERS
    - CONFIG_HZ
  - crear los siguientes directorios
    - /home/robot/bin para almacenar los ejecutables
    - /home/robot/src para almacenar las fuentes

El proceso de compilación C en Linux



Opciones más habituales del compilador GCC

Fase	Opción	Significado
Compilación	-Wall	Habilita todos los warnings
	-Werror	Trata los warnings como errores de compilación
	-o *output*	Especifica el nombre del fichero ejecutable
	-I*dir1*	Directorios de búsqueda de ficheros de cabeceras (.h)
	-O*nivel*	Especifica el nivel de optimización (0-3)
	-g	Habilita la depuración
Enlazado	-L*dir1*	Directorios de búsqueda de librerías (.a, .so)
	-l*lib1*	Nombres de las librerías utilizadas

Fundamentos de C: uso de librerías (1/2)

- Ejemplo de uso de la librería *libintro*

fichero de cabecera	*fichero de librería*
intro.h	**lib**intro.a
	prefijo* + *nombre* + *tipo de librería

- La directiva #include permite utilizar las funciones del fichero de cabecera en un programa
  - se escribe en la parte de *dependencias* del programa
    - se sugiere incluir primero las cabeceras de la librería estándar entre **< >**
    - posteriormente incluir aquellas desarrolladas por los usuarios entre **" "**
- Los ficheros de cabecera **no** incluyen el código fuente de las funciones

Fundamentos de C: uso de librerías (2/2)

- Ejemplo de uso de la librería *libintro*

fichero de cabecera	*fichero de librería*
intro.h	**lib**intro.a
	prefijo* + *nombre* + *tipo de librería

- El código de las funciones se incluye en la *etapa de enlazado*
  - este paso depende del compilador; en el caso del compilador ***gcc***:
    - en general, las funciones de la librería estándar se incluyen por defecto
    - para las librerías no estándar, se añade el flag -l y el nombre de la librería

```
gcc -o hello_io hello_io.c -lintro
```

- No incluir las librerías es un error muy común

```
hello_io.c:(.text+0x16): undefined reference to `get_user_int'
collect2: error: ld returned 1 exit status
```

Ejecución de aplicaciones con compilación nativa

- Implementa una aplicación tipo “hola mundo” que imprima un texto por el terminal
  - crea un fichero C con la aplicación
  - compila la aplicación desde el LEGO
    - a través de un terminal remoto conectado mediante *ssh*
    - utiliza el compilador *gcc* con el siguiente comando

      ```
      gcc -Wall -Werror -o hello hello.c
      ```
    - ejecuta la aplicación y comprueba que funciona correctamente
  - vuelve a compilar la aplicación “hello world” y anota el tiempo de compilación mediante la herramienta *time*
    - es necesario borrar el código objeto y el ejecutable

Compilación cruzada

Interfaz de comunicación

Host u ordenador de desarrollo

- Compilador cruzado
- Depurador
- IDE
- Otras herramientas

Target o sistema empotrado

- Aplicaciones
- Librerías
- Kernel
- Gestor de arranque

Configuración del entorno para la compilación cruzada

- Instala el compilador cruzado para el procesador ARM9 utilizado por el LEGO EV3
  - los binarios están disponible en moodle
- Compila nuestra aplicación tipo “hello world” con el compilador cruzado arm-ev3-linux-gnueabi-gcc
  - compara el tiempo de compilación con el caso anterior
- Ejecuta la aplicación y comprueba que funciona correctamente

Uso de un IDE o entorno de desarrollo: *Eclipse* (1/2)

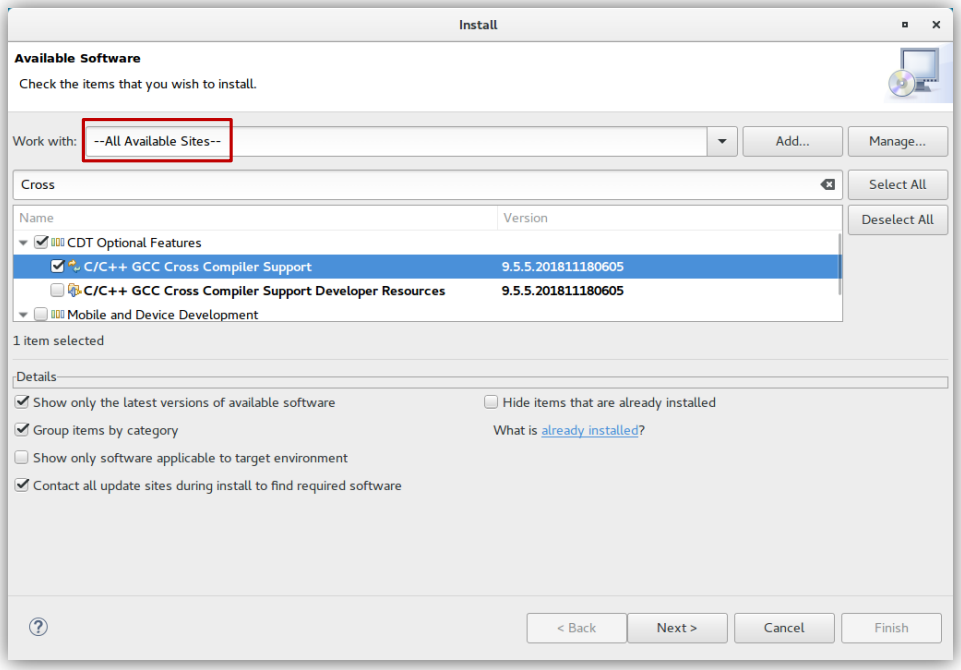
- Instalación en el laboratorio
  - por defecto, están instalados las versiones **Java** y **C** del *Eclipse IDE*
- Instalación personal
  - en caso de linux nativo, puede ser necesario instalar paquetes de desarrollo para 32 bits

    sudo apt-get install gcc-multilib make
  - puede ser necesario instalar el soporte para desarrollo cruzado en C en *Eclipse*
    - **cuidado** con las categorías de los que dependen los paquetes
    - se muestra un ejemplo en la siguiente transparencia

CDT Main Features	CDT Optional Features	General Purpose Tools
• C/C++ Development Tools	• C/C++ GCC Cross Compiler Support • C/C++ Remote Launch	• Remote System Explorer End-User Runtime • Remote System Explorer User Actions

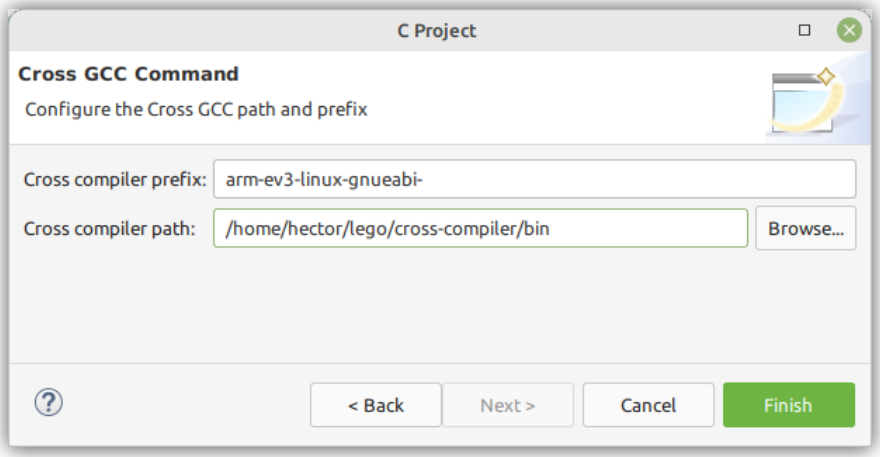
Uso de un IDE o entorno de desarrollo: *Eclipse* (2/2)

- Menú Help -> Install New Software



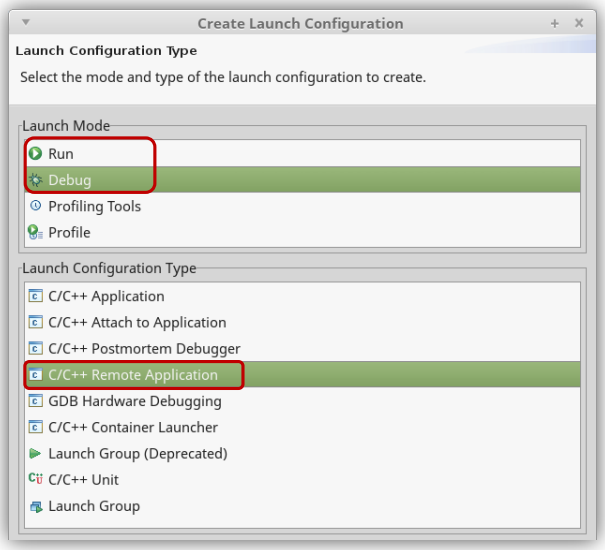
Automatización del proceso de desarrollo en *eclipse* (1/5)

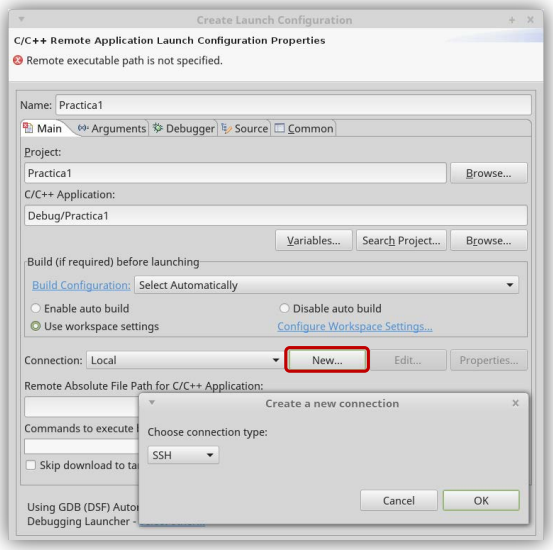
- Crear un **proyecto C** (no C/C++) y configurarlo para que utilice:
  - por sencillez, se recomienda utilizar un **único proyecto** para todas las fuentes
    - y crearemos subcarpetas para cada una de las prácticas
    - elegir un nombre genérico (p.ej., practicas)
  - el compilador cruzado para ARM9
  - otros proyectos pueden requerir configuración adicional



Automatización del proceso de desarrollo en *eclipse* (2/5)

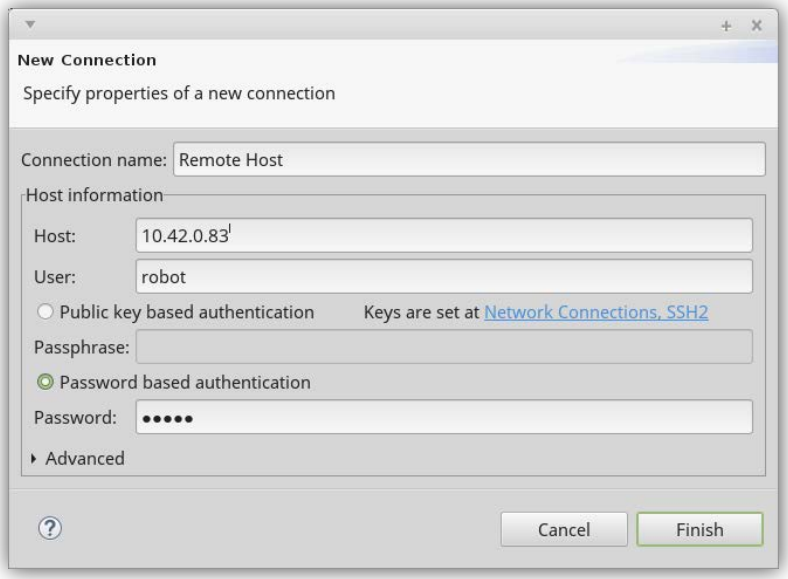
- Crear una carpeta p1 y añadir un fichero C
  - copiar el código de la aplicación previa tipo “hello world”
- Compilar y generar el ejecutable en eclipse
- Configurar la ejecución remota en el robot desde eclipse





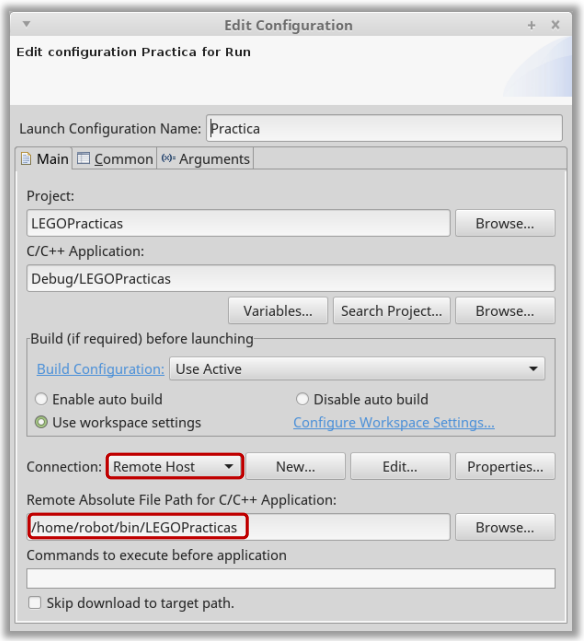
Automatización del proceso de desarrollo en *eclipse* (3/5)

- Configurar la ejecución remota en el robot desde eclipse
  - crear una nueva conexión remota con el LEGO
  - añadir nuestras credenciales de acceso



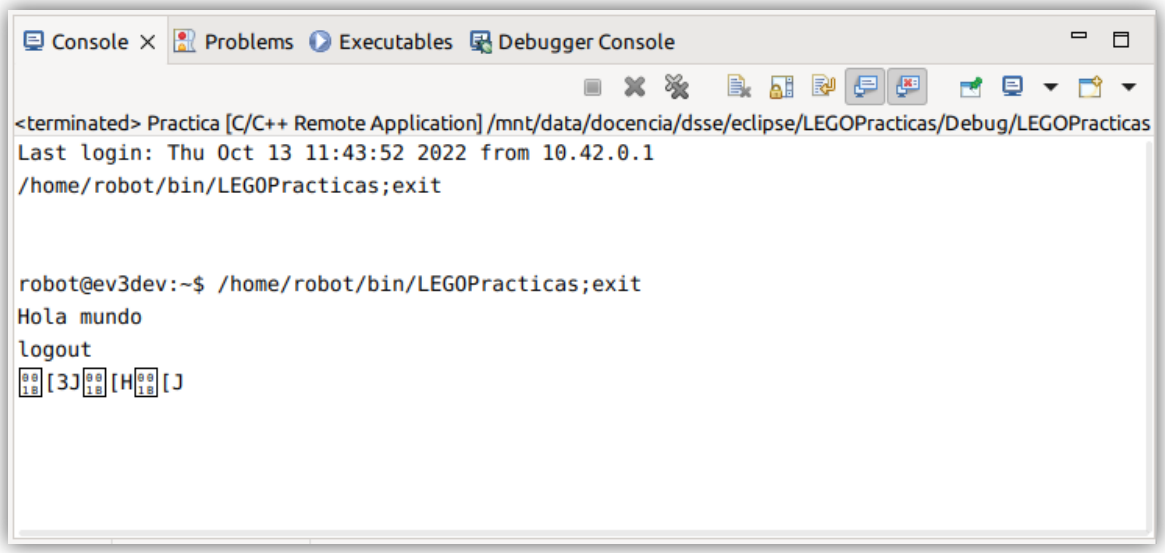
Automatización del proceso de desarrollo en *eclipse* (4/5)

- Configurar la ejecución remota en el robot desde eclipse
  - establecer el path absoluto para la ejecución remota



Automatización del proceso de desarrollo en *eclipse* (5/5)

- Ejemplo de salida del programa en *Eclipse*



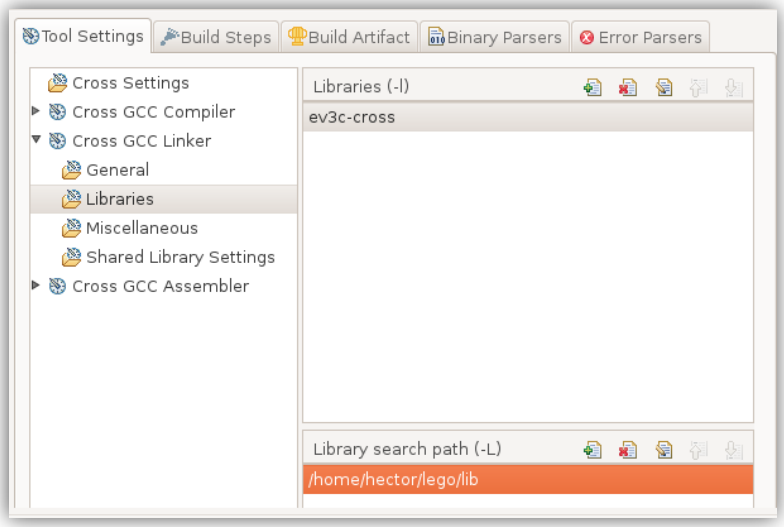
Ejecución de aplicaciones básicas con recursos

- El uso de recursos generalmente comprende tres etapas:

- Implementa un "hola mundo" que utilice el display LCD para la salida del texto
  - todos los recursos están disponibles en ***moodle***
  - las funciones de uso del LCD están definidos en la librería ***libev3c-cross.a*** y en el fichero de cabecera ***ev3c_lcd.h***
    - consulta la **documentación** de la librería
  - escribir el texto en la última línea del LCD
    - pantalla LCD de 178x128 pixels

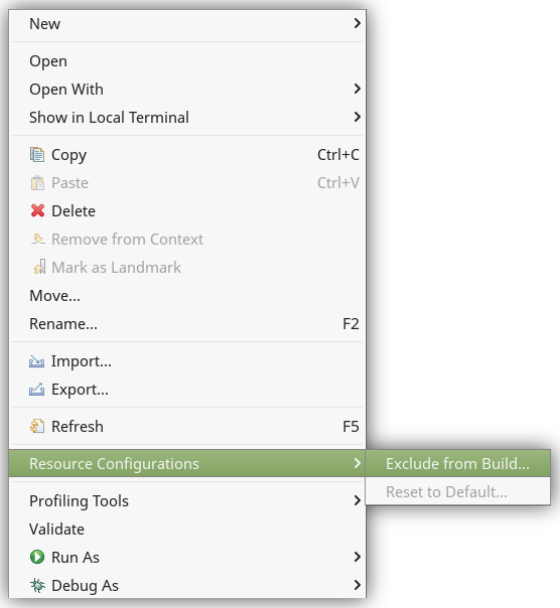
Configuración del uso de recursos en *eclipse* (1/2)

- Crear un fichero C con el "hello world" en la pantalla LCD
  - se debe crear en la carpeta p1
  - configurar el proceso de compilación en la **propiedades del proyecto**
    - opción *C/C++ Build → Settings*
    - configurar los directorios de búsqueda de las cabeceras
    - configurar las librerías utilizadas



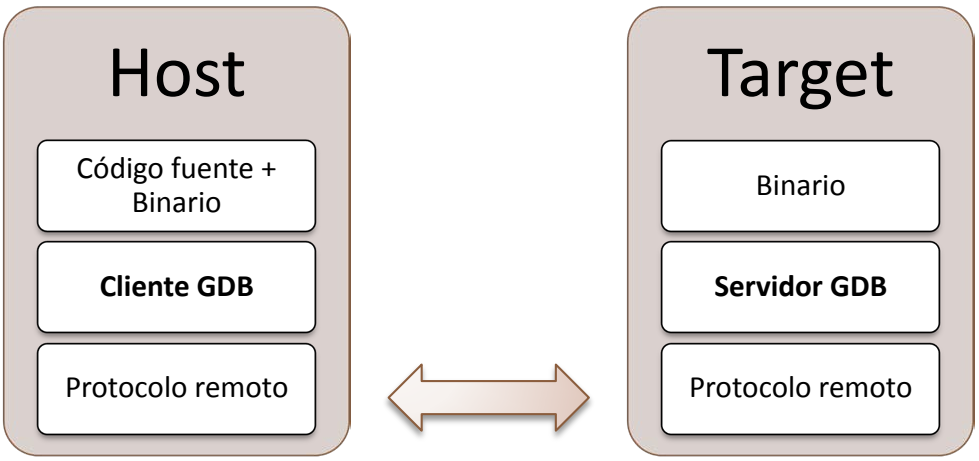
Configuración del uso de recursos en *eclipse* (2/2)

- Crear un fichero C con el “hello world” en la pantalla LCD
  - eclipse sólo permite tener un ***main*** en cada proyecto
    - podemos indicar que no compile el programa anterior
    - podemos crear un proyecto por cada ejecutable



Depuración remota de aplicaciones

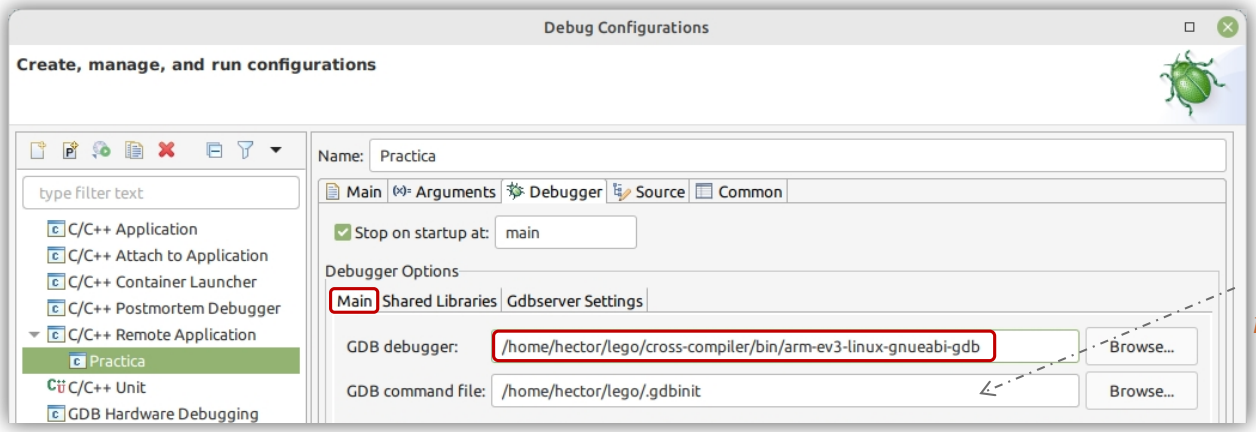
- Volver a compilar uno de los ejemplos con **opciones de depuración**
  - no incluir optimizaciones
- Coherencia en la cadena de herramientas de desarrollo



Depuración remota de aplicaciones en *eclipse* (1/2)

- Copia el servidor del gdb a nuestro *target*
  - el fichero se denomina *arm-ev3-linux-gnueabi-gdbserver*
  - el directorio destino puede ser, por ejemplo, /home/robot/bin
- Configura eclipse para utilizar el depurador de nuestro compilador cruzado
  - ubicación en el host del binario *arm-ev3-linux-gnueabi-gdb*
  - ubicación en el target del *arm-ev3-linux-gnueabi-gdbserver*
- Para casos más avanzados, puede ser necesario añadir comandos de configuración a un fichero *.gdbinit*
- Depura una aplicación tipo “hello world” y comprobar que funciona correctamente

Depuración remota de aplicaciones en *eclipse* (2/2)



no es necesario modificarlo