

**UNIVERSIDAD DE COSTA RICA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS DE LA
COMPUTACIÓN E INFORMÁTICA**

CI2657- ROBÓTICA

Prof. Bach. Kryscia Daviana Ramírez Benavides

Using ant build files

Elaborado por:

Julio Calderon B11226
julioalderonc@gmail.com

Gabriel Quesada B25275
gab.quesada56@gmail.com

Juan Carlos Solano B26441
yuk2794@gmail.com

Leonardo Villalobos B27219
artist_artistian@hotmail.com

22 de agosto 2016

Tabla de contenidos

[Tabla de contenidos](#)

[1. Tema](#)

[2. Objetivos](#)

[3. Enunciado](#)

[4. Desarrollo](#)

[4.1. Especificación de Ant](#)

[4.2. Uso de Ant](#)

[5. Referencias](#)

1. Tema

Apache Ant es una herramienta, similar a make de unix, para automatizar la construcción de software. La herramienta fue escrita en Java, y se especializa en la compilación y empaquetado de proyectos escritos en Java. Tiene la característica que utiliza archivos en formato xml para especificar el proceso de construcción, lo que lo convierte en un formato portátil entre sistemas operativos.

Aunque el método recomendado para utilizar leJOS EV3 sea mediante Eclipse y el respectivo plugin, es posible utilizar archivos de formato ant. De esta forma, se puede subir un .jar al robot EV3 y que este ejecute sus contenidos. Los archivos ant se pueden utilizar desde terminal o desde un IDE que soporte este formato.



Figura #1. Ant: Logo de Apache Ant

2. Objetivos

- Detallar la estructura de un archivo xml Ant
- Explicar cómo utilizar Ant desde terminal y desde IDE
- Exponer un ejemplo de cómo utilizar Ant para compilar un proyecto para EV3

3. Enunciado

El enunciado de esta asignación trata de generar un archivo y una presentación para resumir el tema asignado (Archivos ant para EV3) y exponer un ejemplo del mismo.

4. Desarrollo

4.1. Especificación de Ant

Ant por defecto utiliza el archivo llamado build.xml como instrucciones para la construcción del programa. Se puede adicionar un archivo de propiedades para

incluir variables y acortar el tamaño del archivo de build. El uso de ant evita el uso del plugin de eclipse para compilar, subir y correr archivos .jar en EV3.

Un archivo de ant define un proyecto, que a su vez tiene varios objetivos (target). Cada uno de estos es una acción o un paso para completar el build. Compilar el código fuente, limpiar el directorio de trabajo y empaquetar los archivos compilados son ejemplos de objetivos de un proyecto. Cada target tiene uno o más pasos, que pueden tener como resultado la creación o borrado de archivos. Ejemplos de pasos son crear un jar, borrar un directorio o compilar con javac. Los pasos pueden tener pasos anidados, dando un gran grado de flexibilidad.

Para poder compilar un proyecto es necesario generar un script de ant para que todas las secuencias necesarias sean hechas, este script se guarda como un archivo xml, usualmente llamado build.xml, para ello se presentará ejemplos de las distintas partes que pueden componer el build.xml de ant. Un xml típico de ant se compone de:

1. Definiciones que se utilizarán en el script:

```
<property name="jar" value="${ant.project.name}.jar" />
<property name="main-class" value="${ant.project.name}" />
<property name="ev3.ipaddr" value="10.0.1.1" />
<property name="ev3.user" value="root" />
<property name="ev3.password" value="" />
<property name="ev3classes.path"
value="/home/eric/git/ev3/ev3classes" />
<property name="src" value="src" />
<property name="build" value="bin" />
<property name="dist" value="dist" />
```

O puede utilizarse el formato de propiedad, en un archivo aparte.

2. Los targets para compilación:

```
<path id="compile.classpath">
  <fileset dir="${ev3classes.path}">
    <include name="**/*.jar" />
  </fileset>
</path>
<target name="init">
  <tstamp />
  <mkdir dir="${build}" />
</target>
```

```

<target name="compile" depends="init" description="compile the
sources">
    <javac srcdir="${src}" destdir="${build}"
includeantruntime="false">
        <classpath refid="compile.classpath" />
    </javac>
</target>

```

3. El empaquetado:

```

<target name="dist" depends="compile" description="Create the jar
for the EV3">
    <mkdir dir="${dist}" />
    <jar jarfile="${dist}/${jar}" includes="*.class"
basedir="${build}">
        <fileset dir="${src}" excludes="**/*.java"/>
    </jar>
</target>
<target name="upload" depends="dist" description="Upload the jar
to the EV3">
    <scp file="${dist}/${jar}" todir="${ev3.user}@${ev3.ipaddr}:"
password="${ev3.password}" />
</target>

```

4. La configuración de corrida:

```

<target name="run" depends="upload">
    <sshexec
        host="${ev3.ipaddr}"
        username="${ev3.user}"
        password="${ev3.password}"
        command="jrun -cp ${jar} ${main-class}"
    />
</target>

```

5. La limpieza del directorio:

```

<target name="clean" description="clean up">
    <delete dir="${build}" />
    <delete dir="${dist}" />
</target>

```

Una vez creado el build.xml conteniendo este script con sus componentes, solo es necesario abrirlo con el visor de ant y ejecutarlo, con lo cual el proyecto será compilado en un jar y montado en el EV3 de forma automática. Cabe mencionar

que la mayoría de estas secciones son targets, siendo solo la primera de definiciones.

4.2. Uso de Ant

Ya que sabemos en qué consiste Ant, y cómo funciona el build.xml, estamos listos para saber cómo se corre. Hay dos formas de utilizarlo: desde un IDE que soporte formato Java, o desde la terminal de Linux siendo la última la más sencilla.

Desde la terminal

Una vez que se haya instalado el Apache Ant como lo describe su documentación, correrlo desde la terminal es tan fácil como escribir en la misma el comando “ant”. Cuando no se especifican argumentos, Ant busca el archivo `build.xml` en el directorio actual, si lo encuentra, utiliza este archivo para completar el build y corre el target especificado en el atributo `default` de la etiqueta `target`.

Para hacer que Ant utilice otro archivo para el build que no sea `build.xml`, solamente se pone el comando “-`buildfile file`” donde file es el nombre del archivo que se quiere utilizar para el build. Si utiliza el comando “-`buildfile file`” Ant va a buscar el archivo primero en el directorio actual, luego en el directorio padre, y continuará así hasta encontrarlo, o alcanzar el root.

También se pueden especificar properties en la línea de comandos. Esto se puede hacer con la opción -D `property=value` donde property es el nombre de la propiedad. Si se especifica una propiedad que también está en el build file, el valor especificado en la línea de comandos va a sobrescribir el especificado en el build file.

A continuación se muestran todas las opciones de Ant en la línea de comandos:

```

ant [options] [target [target2 [target3] ...]]
Options:
-help, -h           print this message and exit
-projecthelp, -p    print project help information and exit
-version            print the version information and exit
-diagnostics       print information that might be helpful to
                    diagnose or report problems and exit
                    be extra quiet
-quiet, -q          be extra quiet
-silent, -S        print nothing but task outputs and build failures
-verbose, -v       be extra verbose
-debug, -d         print debugging information
-emacs, -e         produce logging information without adornments
-lib <path>        specifies a path to search for jars and classes
-logfile <file>   use given file for log
  -l <file>         ''
-logger <classname> the class which is to perform logging
-listener <classname> add an instance of class as a project listener
-noinput           do not allow interactive input
-buildfile <file> use given buildfile
  -file <file>     ''
  -f <file>        ''
-D<property>=<value> use value for given property
-keep-going, -k   execute all targets that do not depend
                    on failed target(s)
-propertyfile <name> load all properties from file with -D
                    properties taking precedence
-inputhandler <class> the class which will handle input requests
-find <file>        (s)earch for buildfile towards the root of
  -s <file>         the filesystem and use it
-nice number       A niceness value for the main thread:
                    1 (lowest) to 10 (highest); 5 is the default
-nouserlib        Run ant without using the jar files from ${user.home}/.ant/lib
-noclasspath     Run ant without using CLASSPATH
-autoproxy       Java 1.5+ : use the OS proxies
-main <class>     override Ant's normal entry point

```

Figura #2. Ant Shell: Opciones en la línea de comandos

Desde un IDE

NetBeans: Para la descripción de el uso de Ant con netbeans, nos basamos en un tutorial que se encuentra en <http://wiki.netbeans.org>. Este tutorial se basa en el uso de un archivo wrap.xml además de el build.xml, el cual envuelve todos los target de el build.xml usando una macro especial que agrega una configuración lógica específica.

1. Se debe descargar el archivo [Media:nbproject_NetbeansedAnt.zip](#) que se encuentra en la página del tutorial, y extraerlo dentro del directorio de <ant_src_root>. Ahora se puede observar el directorio debajo de el directorio . Hay tres archivos debajo el directorio project.properties, project.xml, y wrap.xml
2. Modifique el archivo project.properties para poner las propiedades project.dir, jdk12.home, jdk14.home, y jdk16.home.
3. Desde el menú principal vaya a File > Open Project. Una vez aquí especifique <ant_src_root> y click en Open Project. La ventana de proyectos va a contener un nodo de proyecto para el Ant que acabamos de crear. Ahora ya se puede desarrollar Ant con el IDE NetBeans.

Eclipse:

1. Se debe crear un proyecto nuevo donde se van a meter todas las clases que necesitamos para nuestro programa que correrá en el EV3. Para este ejemplo vamos a crear un Hola Mundo simple.

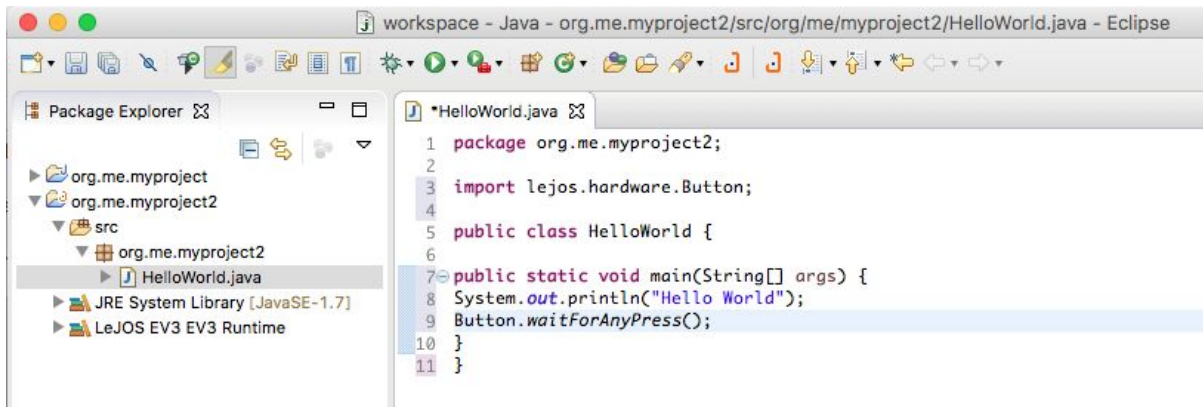


Figura #3. HelloWorld.java: Ejemplo en Eclipse

2. Luego, copiamos archivos build.xml y build.properties desde algún proyecto de ejemplo desde la carpeta EV3 que tenemos ligada a Eclipse. Dentro de estos archivos, vamos a cambiar los nombres de los archivos para que al hacer el build con ant, use esos archivos.

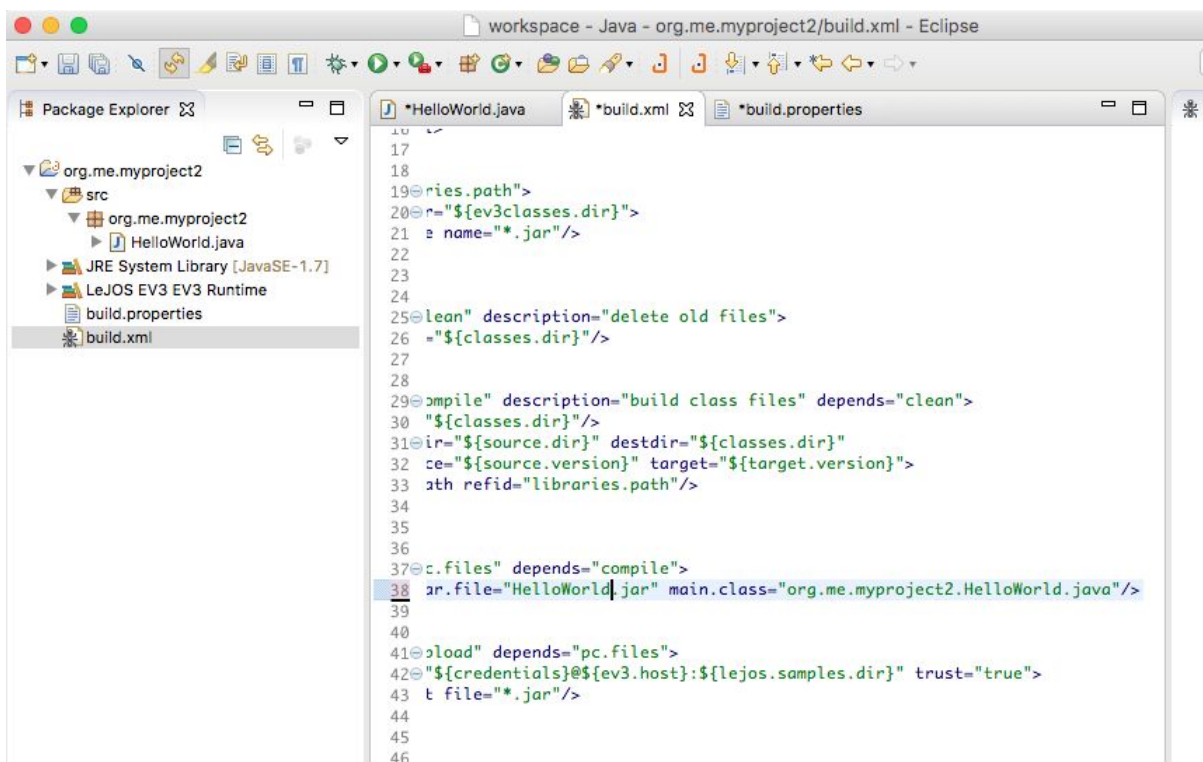


Figura #4. build.xml: Archivo en formato Ant

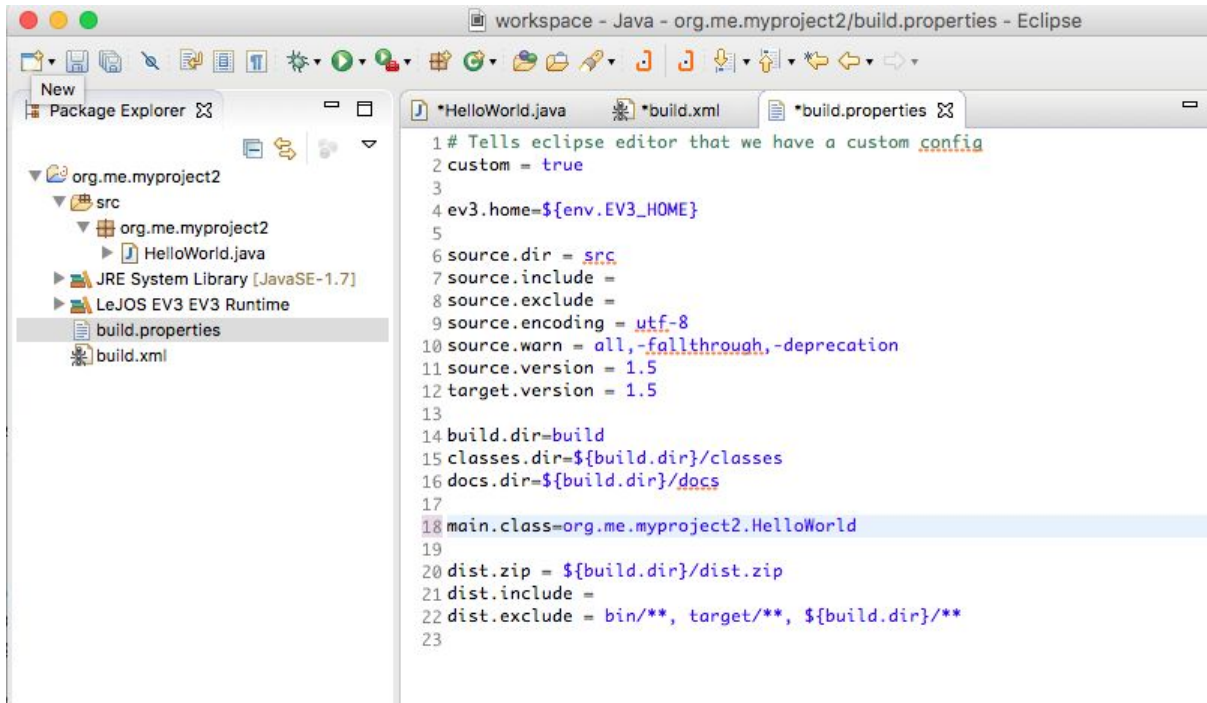


Figura #5. build.properties: Archivo de propiedades

3. Luego, hacemos click derecho en el archivo build.xml y ejecutamos el proyecto con Ant. Debemos tener el EV3 conectado para este punto, ya que lo va a subir directamente al robot y lo va a ejecutar.

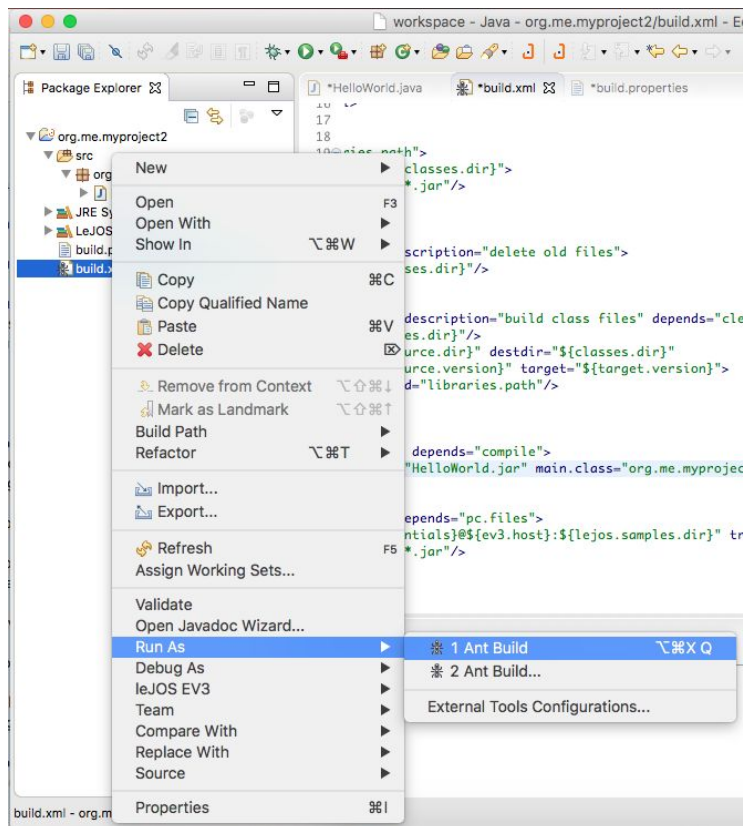


Figura #6. Eclipse: Corriendo el archivo Ant

5. Referencias

Guía de leJOS EV3 sobre archivos ant:

<https://sourceforge.net/p/lejos/wiki/Using%20ant%20build%20files/>

Manual de la fundación Apache sobre archivos ant:

<https://ant.apache.org/manual/using.html>

Página en Wikipedia sobre Apache Ant: https://en.wikipedia.org/wiki/Apache_Ant

Guí para primer java en EV3:

<http://www.pobot.org/A-first-Java-program-for-the-EV3.html?lang=fr>