Articles » Internet of Things » Boards / Embedded devices » Arduino

Creating Arduino programs in Eclipse

Jan Dolinay, 23 Jun 2015 CPOL

5.00 (5 votes)

Article on setting up Eclipse to build programs for Arduino Uno.

Introduction

This article describes how to set up Eclipse IDE and AVR Eclipse plugin to be able to develop programs on a Windows computer for Arduino Uno. It should work also for other AVR-based Arduinos.

There are many articles on the Internet about developing programs for Arduino, but most of them are out of date, too complicated, based on Linux system and so on. This is quite natural considering the speed with which the tools are changing, so this article will sooner or later also become out of date. But for now (middle of 2015) it should work fine and I believe it is also "reasonably simple" (read as simple as I could make it).

This article is based on the Eclipse Luna SR2 with AVR Eclipse plugin and Arduino IDE 1.6.5-r2 (all tools in their latest versions as of June 22, 2015).

Background

I probably don't need to tell you why use Eclipse instead of the Arduino IDE. If you plan to create more complex project with your Arduino or even see yourself as an embedded software developer some day, you will soon outgrow the Arduino IDE. And then there are two main options: Eclipse or Atmel Studio. If you read this, you probably decided for Eclipse, which is choice I like myself too.

Once you decided for Eclipse, there are two main roads you can choose:

- Arduino Eclipse plugin (http://www.baeyens.it/eclipse/) or
- AVR Eclipse plugin (http://avr-eclipse.sourceforge.net/wiki/index.php/The_AVR_Eclipse_Plugin).

The first road is easier, but I did not like it when I tried it some time ago. Personally, I consider it somewhere in between the real-world programming and the Arduino IDE. For example, the compiler options you can set are very limited. If you decided to use the real-world tools, why stop half way through? But the plugin is developing quickly, so maybe in the current version things have improved. But anyway, this article is about the second road, using AVR Eclipse plugin.

Install the tools

We will need the following tools:

- Arduino software (http://www.arduino.cc/en/Main/Software)
- Eclipse IDE (https://eclipse.org/downloads/)
- AVR Eclipse plugin for Eclipse (installed from Eclipse IDE, do not download separately)
- MinGW tools (http://www.mingw.org/)

Here are installation instructions:

Arduino
From http://www.arduino.cc/en/Main/Software download the Arduino software.

Current version (June 2015) is 1.6.5-r2. You can either download installer or a zip file. In case of zip file just extract it to any folder on your computer, for example, to c:\programs\arduino1.6.5-r2.

Eclipse
From https://eclipse.org/downloads/ download the **Eclipse IDE for C/C++ Developers**;

Current version (June 2015) is Luna SR2. I recommend using 32-bit version even for 64-bit Windows, because I am not sure if everything would work with 64-bit Eclipse.

There is no installation needed. You download a .zip file, which you can extract into any folder on your computer, for example, to c:\programs\eclipse.
To start Eclipse IDE, go to the eclipse folder and run **eclipse.exe**.
You may want to create link on your desktop for quick access.

When Eclipse starts, you need to select location for your workspace. This is a folder on your computer where all your projects will be located. Later you can use several workspaces and change the default location, but for now I recommend just accepting the default offered by Eclipse and moving on.

AVR Eclipse plugin
Start the Eclipse IDE and in the main menu select Help > Install New Software.
In the *Work with* box enter this update site address:
http://avr-eclipse.sourceforge.net/updatesite
and press Enter on the keyboard.

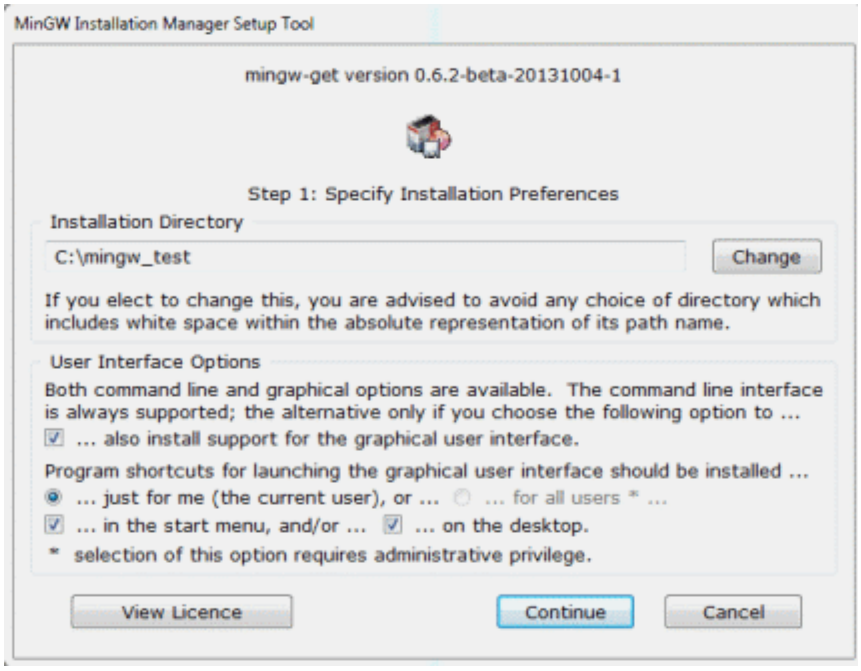
After a while the list below will display AVR Eclipse Plugin item. Select it by checking the box next to the plugin name and click the Next button below.
Follow the wizard to install the plugin. It takes quite a while. You will need to agree with license agreement and confirm installing of unsigned plugin. After the installation, accept the offer to restart Eclipse.

Tip: You can find tutorial with pictures on the AVR Eclipse plugin website at:
http://avr-eclipse.sourceforge.net/wiki/index.php/Plugin_Download

MinGW tools
The MinGW tools are needed to obtain the GNU make utility. If you have Atmel Studio 6.x installed on your computer, you can skip this step; you will find make.exe in the [Atmel Studio location]\shellUtils. We will need this path in AVR Eclipse plugin configuration as described in the next section.

To install MinGW:
From http://www.mingw.org/ follow the **Downloads** link on the left.
This will take you to http://sourceforge.net/projects/mingw/files/.
Near the top of the page you will find "Looking for the latest version?" line and a link to download.
Download the **mingw-get-setup.exe** (direct link: http://sourceforge.net/projects/mingw/files/latest/download?source=files).

Run the setup program. In the first screen leave all settings at defaults, change the Installation Directory if you prefer and click Continue.



MinGW Installation Manager Setup Tool

mingw-get version 0.6.2-beta-20131004-1

Step 1: Specify Installation Preferences

Installation Directory

C:\mingw_test [Change]

If you elect to change this, you are advised to avoid any choice of directory which includes white space within the absolute representation of its path name.

User Interface Options

Both command line and graphical options are available. The command line interface is always supported; the alternative only if you choose the following option to ...

☑ ... also install support for the graphical user interface.

Program shortcuts for launching the graphical user interface should be installed ...

[✓ ... just for me (the current user)], or ... ○ ... for all users * ...

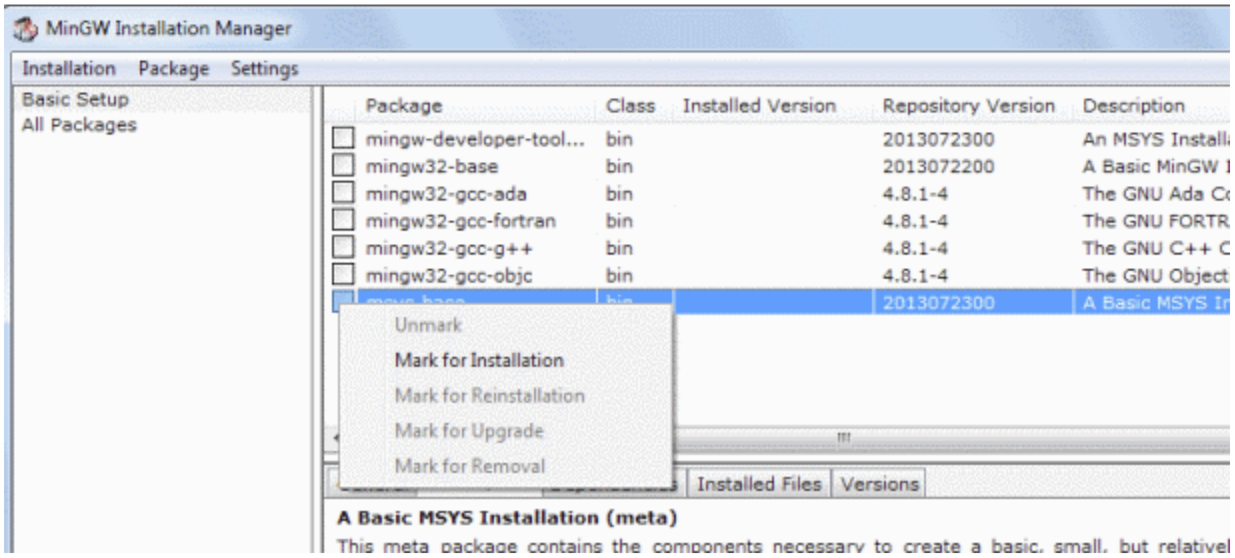
☑ ... in the start menu, and/or ... ☑ ... on the desktop.

* selection of this option requires administrative privilege.

In the next step wait for the download of some files and click Continue.

You should see a MinGW Installation Manager window. In the list of available packages click on the box next to "**msys base**" (the last item in the list).

From the context menu select Mark for Installation.



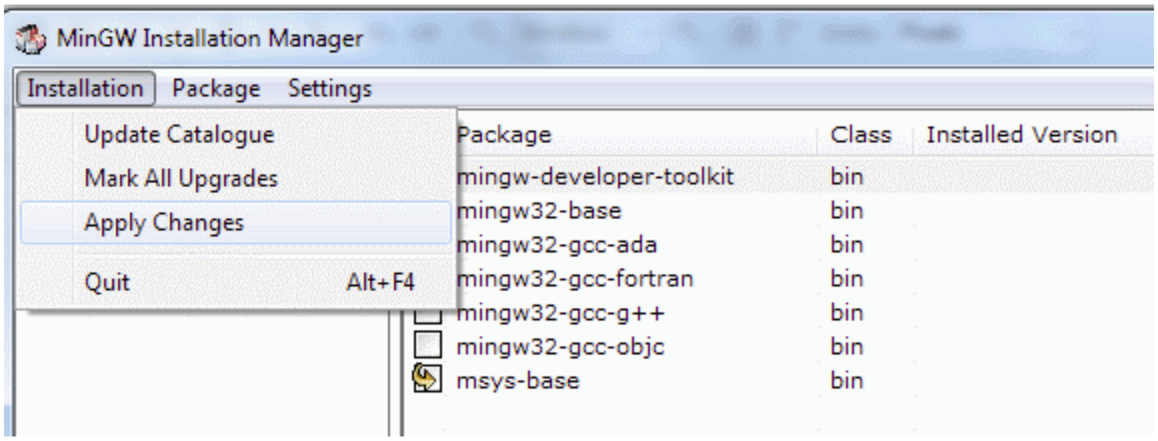
MinGW Installation Manager

	Package	Class	Installed Version	Repository Version	Description
☐	mingw-developer-tool...	bin		2013072300	An MSYS Install
☐	mingw32-base	bin		2013072200	A Basic MinGW I
☐	mingw32-gcc-ada	bin		4.8.1-4	The GNU Ada C
☐	mingw32-gcc-fortran	bin		4.8.1-4	The GNU FORTR
☐	mingw32-gcc-g++	bin		4.8.1-4	The GNU C++ C
☐	mingw32-gcc-objc	bin		4.8.1-4	The GNU Object
☐	msys-base	bin		2013072300	A Basic MSYS In

A Basic MSYS Installation (meta)

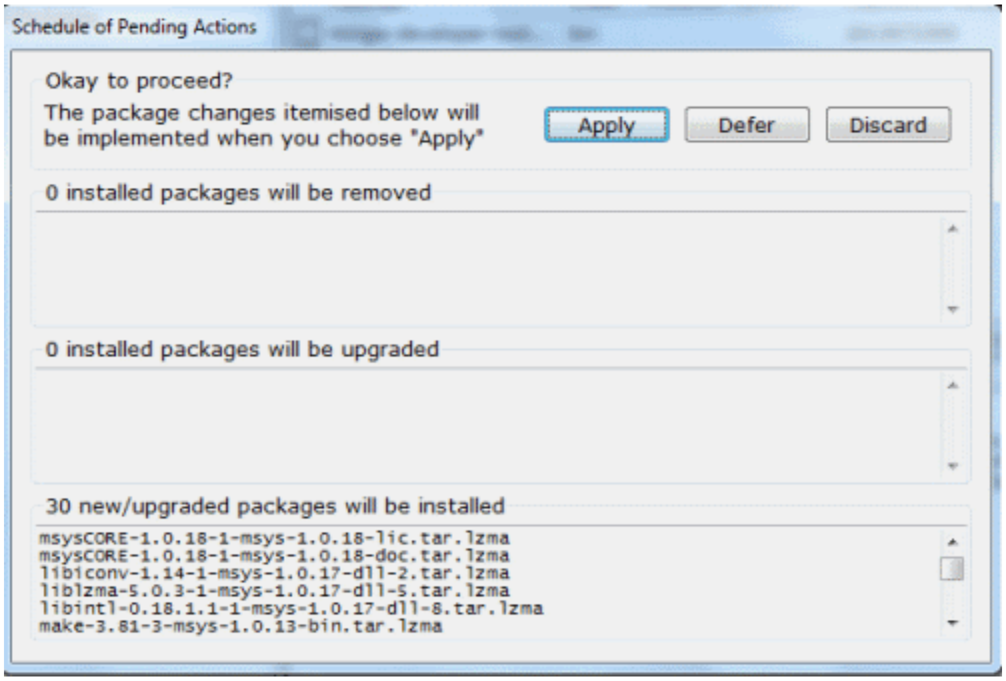
This meta package contains the components necessary to create a basic, small, but relativel

From the menu at the top of the window select Installation > Apply Changes.



Package	Class	Installed Version
mingw-developer-toolkit	bin	
mingw32-base	bin	
mingw32-gcc-ada	bin	
mingw32-gcc-fortran	bin	
mingw32-gcc-g++	bin	
mingw32-gcc-objc	bin	
msys-base	bin	

In the next step click Apply button.



Okay to proceed?
The package changes itemised below will be implemented when you choose "Apply"

0 installed packages will be removed

0 installed packages will be upgraded

30 new/upgraded packages will be installed

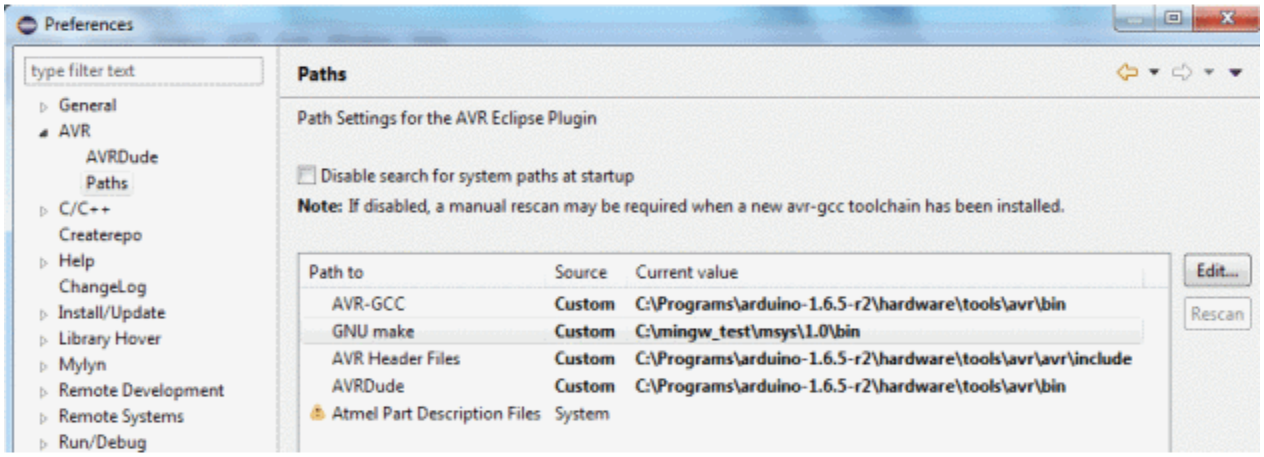
```
msysCORE-1.0.18-1-msys-1.0.18-lic.tar.lzma
msysCORE-1.0.18-1-msys-1.0.18-doc.tar.lzma
libiconv-1.14-1-msys-1.0.17-dll-2.tar.lzma
liblzma-5.0.3-1-msys-1.0.17-dll-5.tar.lzma
libintl-0.18.1.1-1-msys-1.0.17-dll-8.tar.lzma
make-3.81-3-msys-1.0.13-bin.tar.lzma
```

After a while the MinGW tools should be installed. Close the window.

We are now ready to configure the AVR Eclipse plugin.

Set up the AVR Eclipse plugin

Start Eclipse and in the main menu select Window > Preferences.
In the Preferences window expand AVR > Paths.
Use the Edit... button on the right to set the paths as shown in the following picture.



Path Settings for the AVR Eclipse Plugin

Disable search for system paths at startup

Note: If disabled, a manual rescan may be required when a new avr-gcc toolchain has been installed.

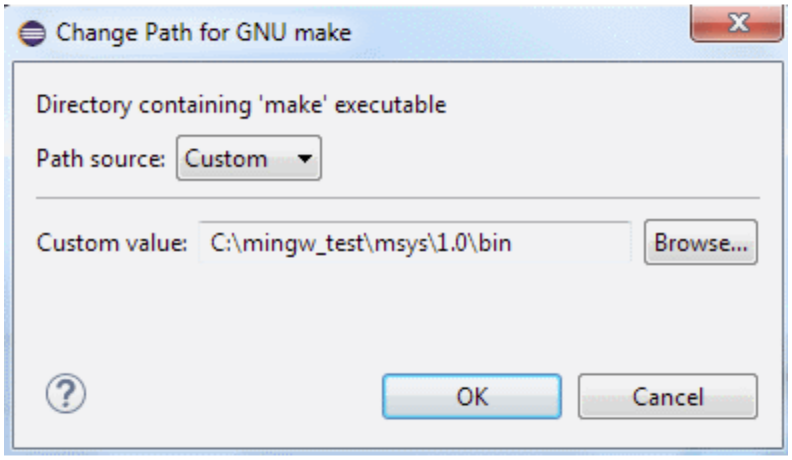
Path to	Source	Current value
AVR-GCC	Custom	C:\Programs\arduino-1.6.5-r2\hardware\tools\avr\bin
GNU make	Custom	C:\mingw_test\msys\1.0\bin
AVR Header Files	Custom	C:\Programs\arduino-1.6.5-r2\hardware\tools\avr\avr\include
AVRDude	Custom	C:\Programs\arduino-1.6.5-r2\hardware\tools\avr\bin
Atmel Part Description Files	System	

Here are the path on my computer:

AVR-GCC: C:\Programs\arduino-1.6.5-r2\hardware\tools\avr\bin
GNU make: C:\mingw_test\msys\1.0\bin
AVR Header Files: C:\Programs\arduino-1.6.5-r2\hardware\tools\avr\avr\include
AVRDude: C:\Programs\arduino-1.6.5-r2\hardware\tools\avr\bin

You can see that most of the paths point to the Arduino location. The GNU make tool (make.exe) path points to the MinGW location. The paths may be different on your system, depending on where you installed the tools.

To set the paths, use the Browse button in the window for path selection and select Path source: Custom; see the picture below.



Directory containing 'make' executable

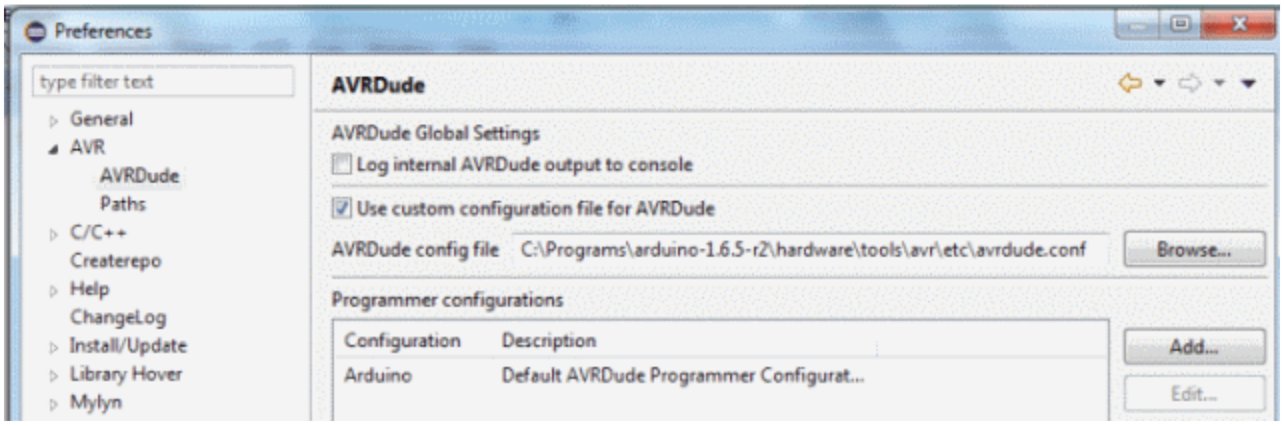
Path source: Custom

Custom value: C:\mingw_test\msys\1.0\bin

Next, select the AVRDude category on the left.

Check the box "Use custom configuration file for AVRDude".

In “AVRDude config file” box browse to the path of this file in your Arduino installation. Here is how it looks:



AVRDude

AVRDude Global Settings

☐ Log internal AVRDude output to console

☑ Use custom configuration file for AVRDude

AVRDude config file C:\Programs\arduino-1.6.5-r2\hardware\tools\avr\etc\avrdude.conf

Programmer configurations

Configuration	Description
Arduino	Default AVRDude Programmer Configurat...

The config file path is **[your Arduino path]\hardware\tools\avr\etc\avrdude.conf**.

Close the preferences window by clicking the OK button.

Note: We will return to this window again in the next step, but it seems closing the Preferences window is required to apply the changes. Without this, the next step fails with error that AVRDude cannot find its configuration file.

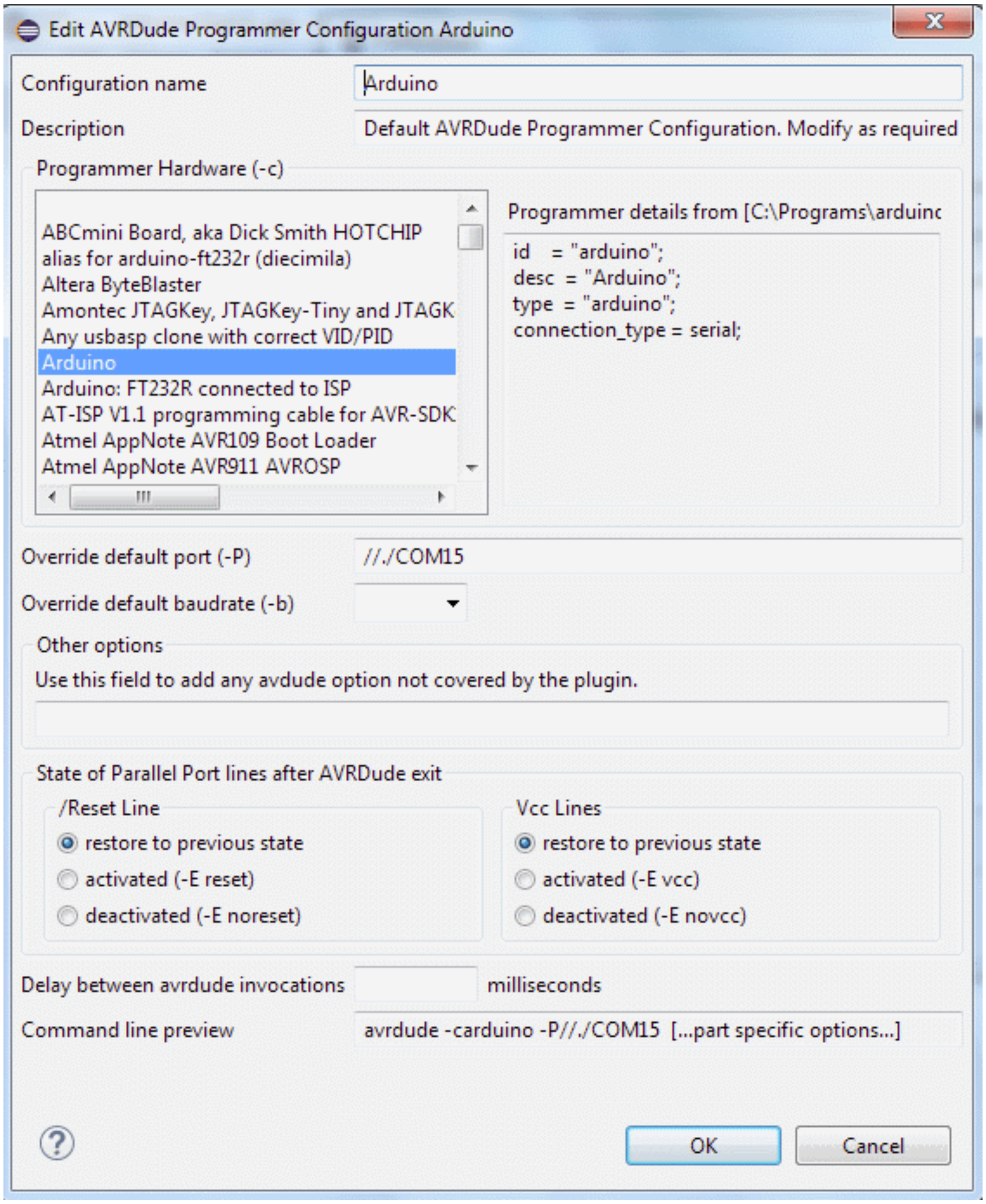
Open again the Preferences from Window menu.

Select AVR > AVRDude.

Click the Add button on the right-side of the the Programmer configuration list.

In the window which opens, select Arduino from the “Programmer Hardware” list.
Enter some name in the Configuration name field above the list, for example, **Arduino**.
Enter the serial port to which your Arduino is connected into the "Override default port" box below the list.

Note that the port name must be written like this on Windows: **"//./COM15"**. (with your COM port number instead of 15, of course). See the picture below.


Edit AVRDude Programmer Configuration Arduino

Configuration name: Arduino

Description: Default AVRDude Programmer Configuration. Modify as required

Programmer Hardware (-c)

- ABCmini Board, aka Dick Smith HOTCHIP
- alias for arduino-ft232r (diecimila)
- Altera ByteBlaster
- Amontec JTAGKey, JTAGKey-Tiny and JTAGK
- Any usbasp clone with correct VID/PID
- Arduino
- Arduino: FT232R connected to ISP
- AT-ISP V1.1 programming cable for AVR-SDK
- Atmel AppNote AVR109 Boot Loader
- Atmel AppNote AVR911 AVROSP

Programmer details from [C:\Programs\arduinc

```
id   = "arduino";
desc  = "Arduino";
type  = "arduino";
connection_type = serial;
```

Override default port (-P): //./COM15

Override default baudrate (-b):

Other options

Use this field to add any avdude option not covered by the plugin.

State of Parallel Port lines after AVRDude exit

/Reset Line: [✓ restore to previous state] activated (-E reset) deactivated (-E noreset)

Vcc Lines: [✓ restore to previous state] activated (-E vcc) deactivated (-E novcc)

Delay between avrdude invocations: milliseconds

Command line preview: `avrdude -carduino -P//./COM15  [...part specific options...]`

Close the window with OK and then close the Preferences window also. The IDE is now ready for Arduino development.

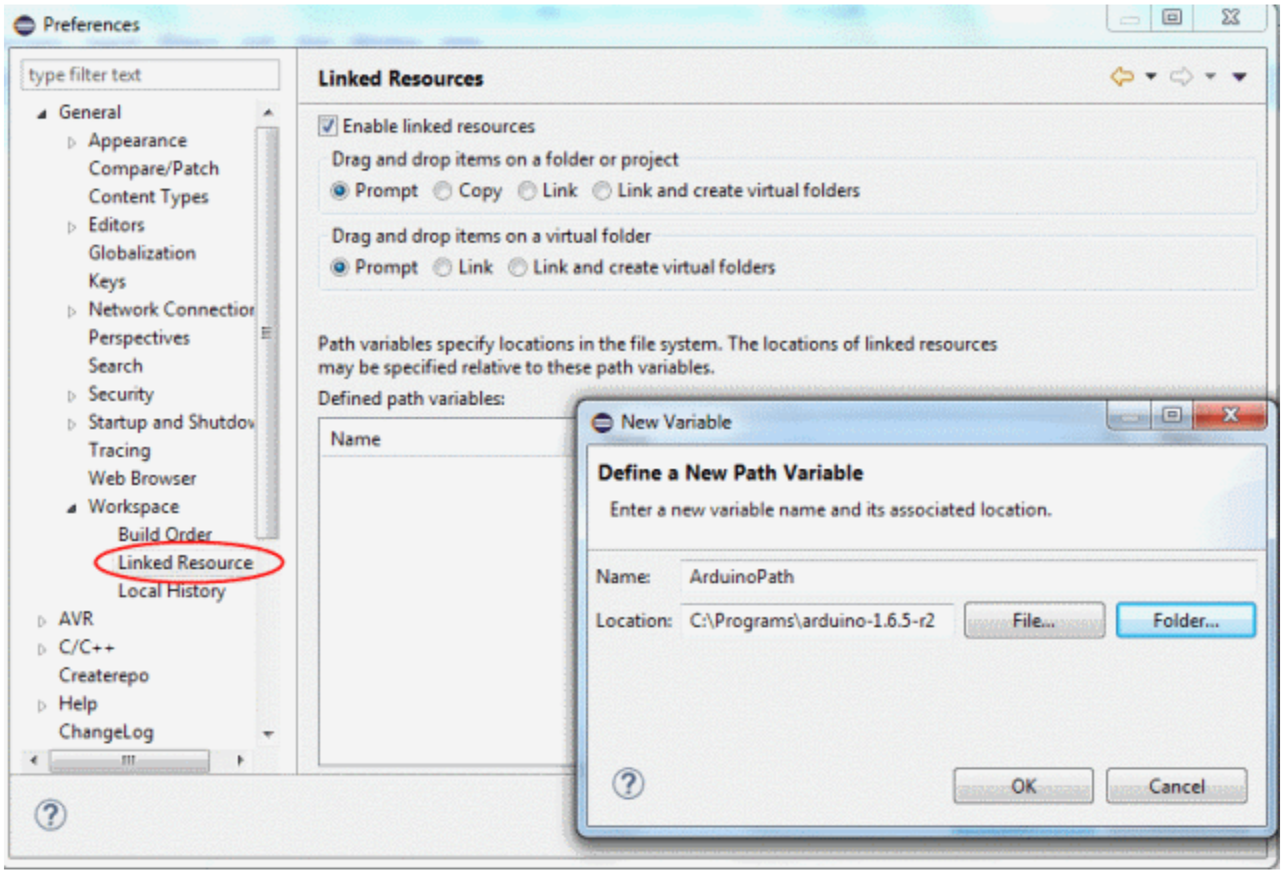
Create your first Arduino program

This section describes how to create project with Arduino MCU Framework (the software library), so that you can use the Arduino functions in your program.
First, we will create a path variable in Eclipse which will allow us to easily update our projects when new version of Arduino appears. This is only needed once in a workspace.

In Eclipse main menu select Window > Preferences.

In the Preferences window expand General > Workspace > Linked Resources.

Click the New button to add a new path variable.
Name it **ArduinoPath** and point it to the main folder of your Arduino installation, in my case c:\programs\arduino-1.6.5-r2. See the picture.



[✓] Enable linked resources

Drag and drop items on a folder or project
[✓ Prompt] Copy Link Link and create virtual folders

Drag and drop items on a virtual folder
[✓ Prompt] Link Link and create virtual folders

Path variables specify locations in the file system. The locations of linked resources may be specified relative to these path variables.

Defined path variables:

Define a New Path Variable

Enter a new variable name and its associated location.

Name: ArduinoPath

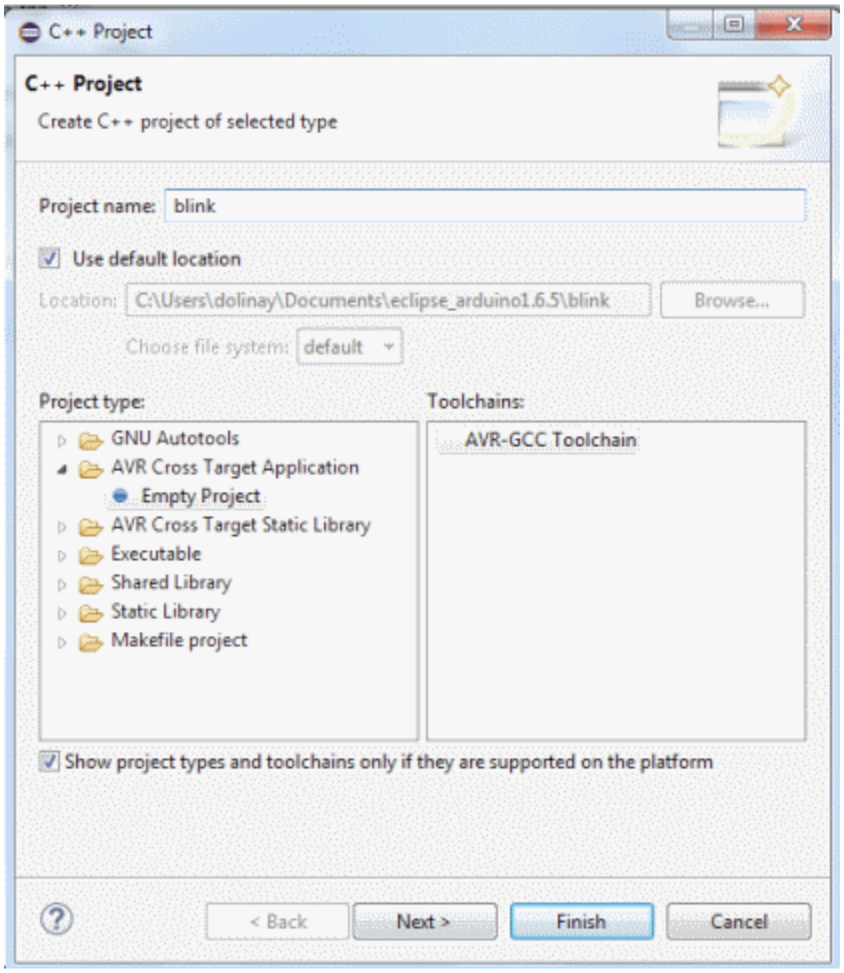
Location: C:\Programs\arduino-1.6.5-r2

Close the preferences window.

Now we are ready to create new project.

In Eclipse go to File menu > New > C++ Project.

Enter name of your project, for example, **blink**.
Select AVR Cross Target Application > Empty Project and click Next.



C++ Project

C++ Project

Create C++ project of selected type

Project name: blink

[✓] Use default location

Location: C:\Users\dolinay\Documents\eclipse_arduino1.6.5\blink

Choose file system: default

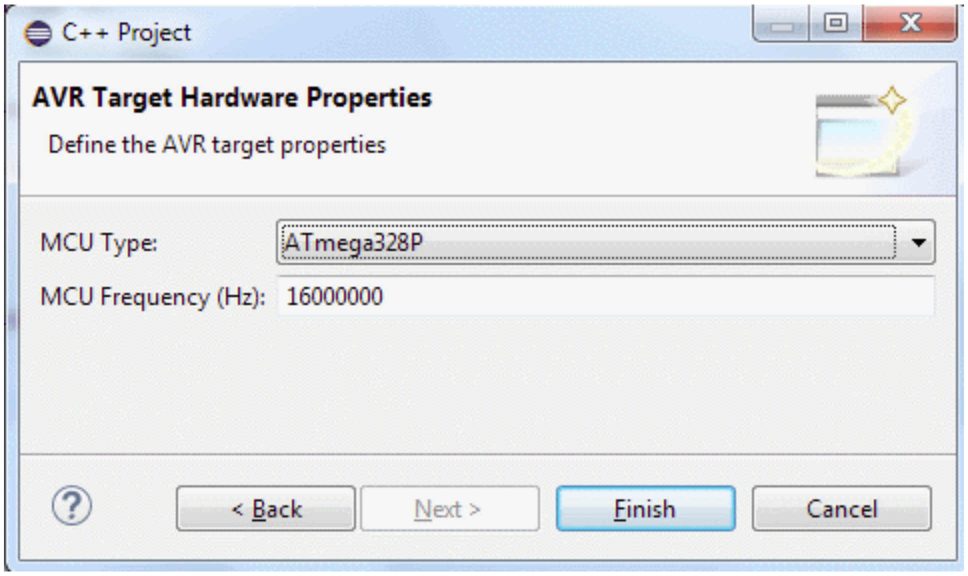
Project type:
- GNU Autotools
- AVR Cross Target Application
  - Empty Project
- AVR Cross Target Static Library
- Executable
- Shared Library
- Static Library
- Makefile project

Toolchains:
- AVR-GCC Toolchain

[✓] Show project types and toolchains only if they are supported on the platform

In the next step leave everything as is and click Next.

In the next step select MCU Type: **ATmega328P** and MCU Frequency (Hz): **16000000**. (that is 16 MHz).



C++ Project

AVR Target Hardware Properties

Define the AVR target properties

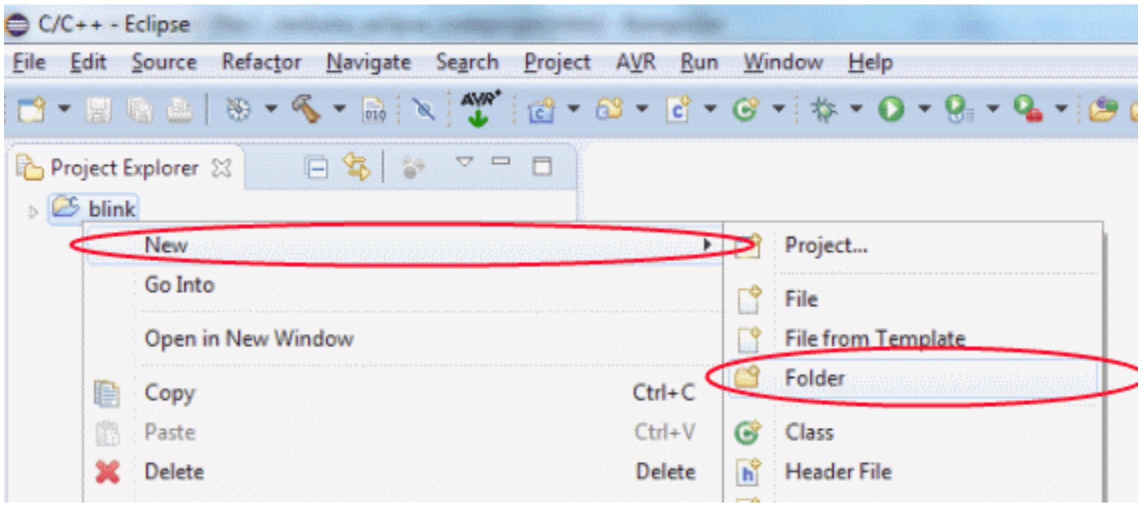
MCU Type: ATmega328P

MCU Frequency (Hz): 16000000

Click Finish.

We now have an empty project and will add links to the Arduino files into the project.

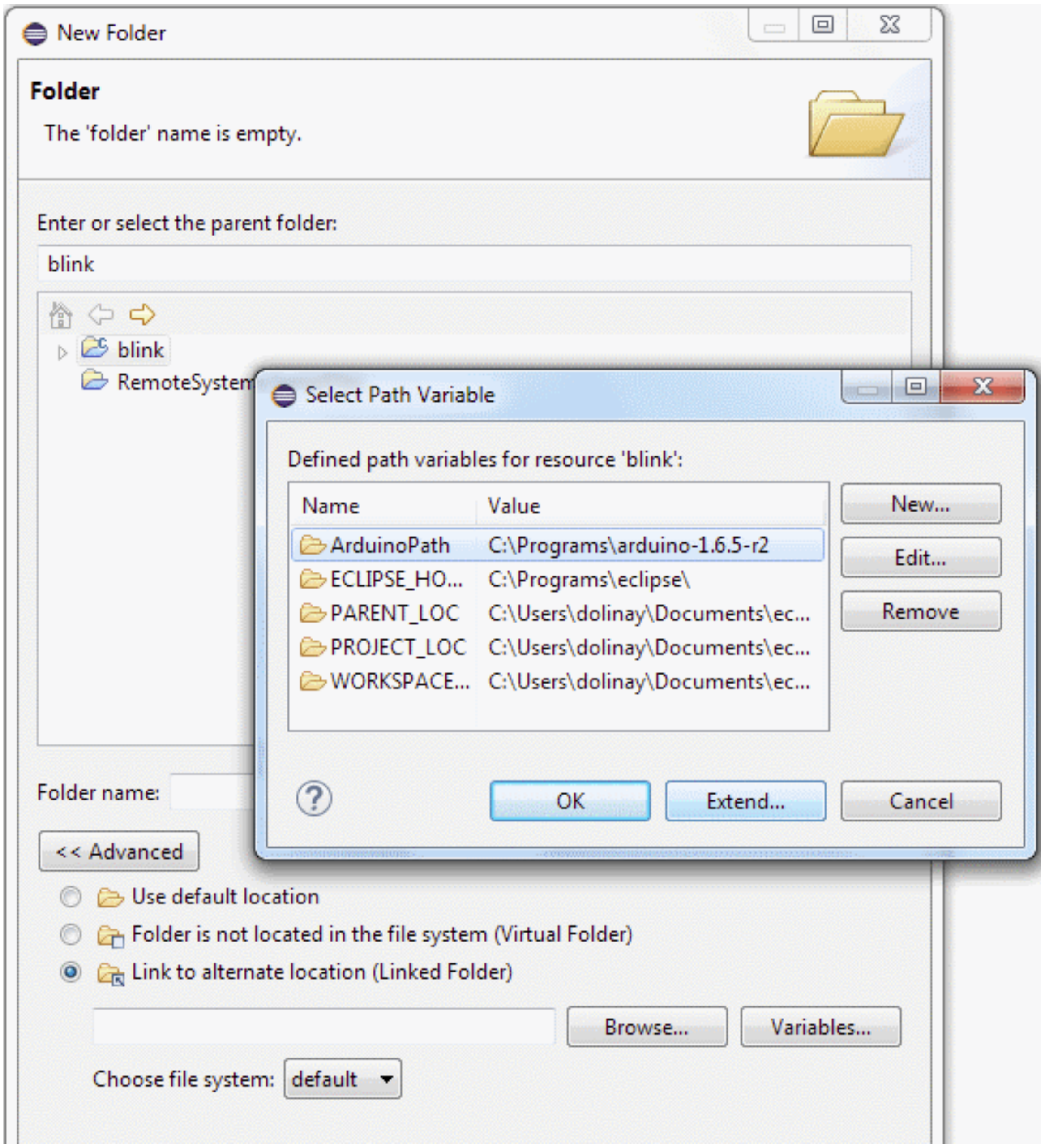
In the Project Explorer window in Eclipse right-click on the project and select New > Folder from the context menu.



In the New Folder window expand the Advanced options in the lower part of the window and select the "Link to Alternate location (Linked folder)" option.

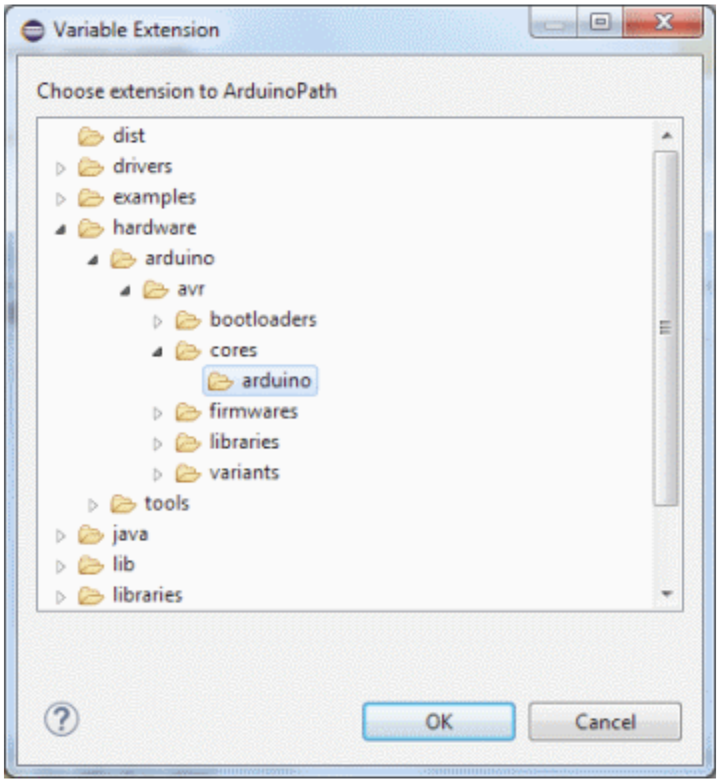
Then click the Variables... button and select our ArduinoPath variable from the list.

Then click Extend... button.

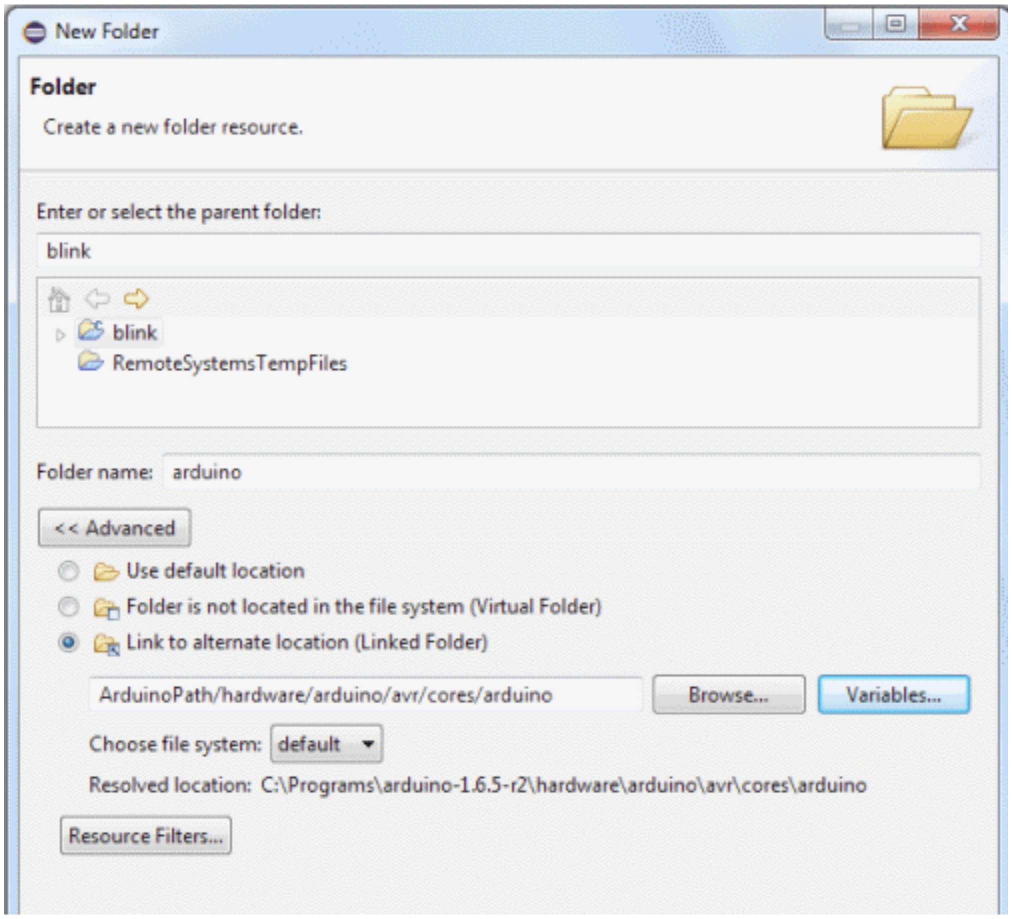

New Folder
Folder
The 'folder' name is empty.
Enter or select the parent folder:
blink
blink
RemoteSystem
Folder name:
<< Advanced
Use default location
Folder is not located in the file system (Virtual Folder)
Link to alternate location (Linked Folder)
Browse...
Variables...
Choose file system: default
Select Path Variable
Defined path variables for resource 'blink':
New...
Edit...
Remove
OK
Extend...
Cancel

Name	Value
ArduinoPath	C:\Programs\arduino-1.6.5-r2
ECLIPSE_HO...	C:\Programs\eclipse\
PARENT_LOC	C:\Users\dolinay\Documents\ec...
PROJECT_LOC	C:\Users\dolinay\Documents\ec...
WORKSPACE...	C:\Users\dolinay\Documents\ec...

In the Variable Extension window you should see the sub-folders of the Arduino folder.
Browse to hardware\arduino\avr\cores\arduino and click OK.



Now you will be back in the New Folder window and the path to our linked folder should point to **ArduinoPath/hardware /arduino/avr/cores/arduino**; see the picture below.

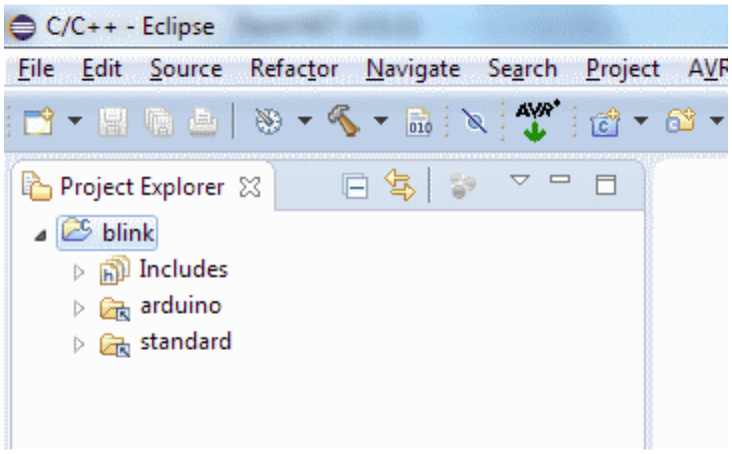

New Folder
Folder
Create a new folder resource.
Enter or select the parent folder:
blink
blink
RemoteSystemsTempFiles
Folder name: arduino
<< Advanced
Use default location
Folder is not located in the file system (Virtual Folder)
Link to alternate location (Linked Folder)
ArduinoPath/hardware/arduino/avr/cores/arduino
Browse...
Variables...
Choose file system: default
Resolved location: C:\Programs\arduino-1.6.5-r2\hardware\arduino\avr\cores\arduino
Resource Filters...

Click Finish to close the window.

Repeat the same procedure for another folder. The location of this folder will be:

ArduinoPath/hardware/arduino/avr/variants/standard.

Now you should see two (linked) sub-folders in your project: arduino and standard.



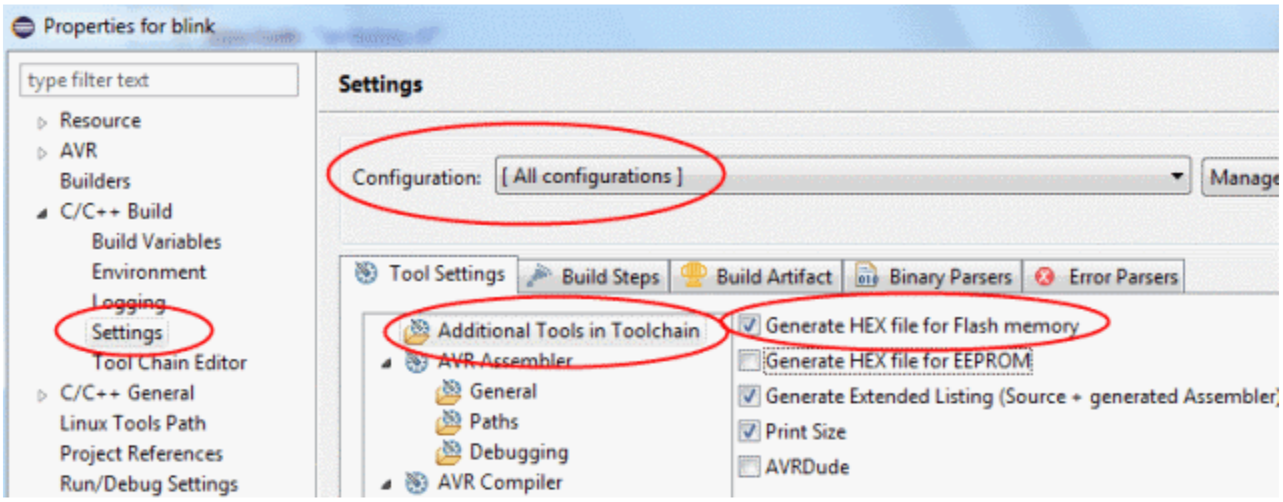
Go to preferences for your project (Alt + Enter or right-click the project and select Properties).

Expand C/C++ Build > Settings category.

At the top of the window, in "Configuration" list select [All configurations].

Select the Additional Tools in Toolchain sub-category.

Check the box next to "Generate HEX file for Flash memory".
The HEX file is the output of the build process which is later uploaded to the microcontroller on your Arduino board.



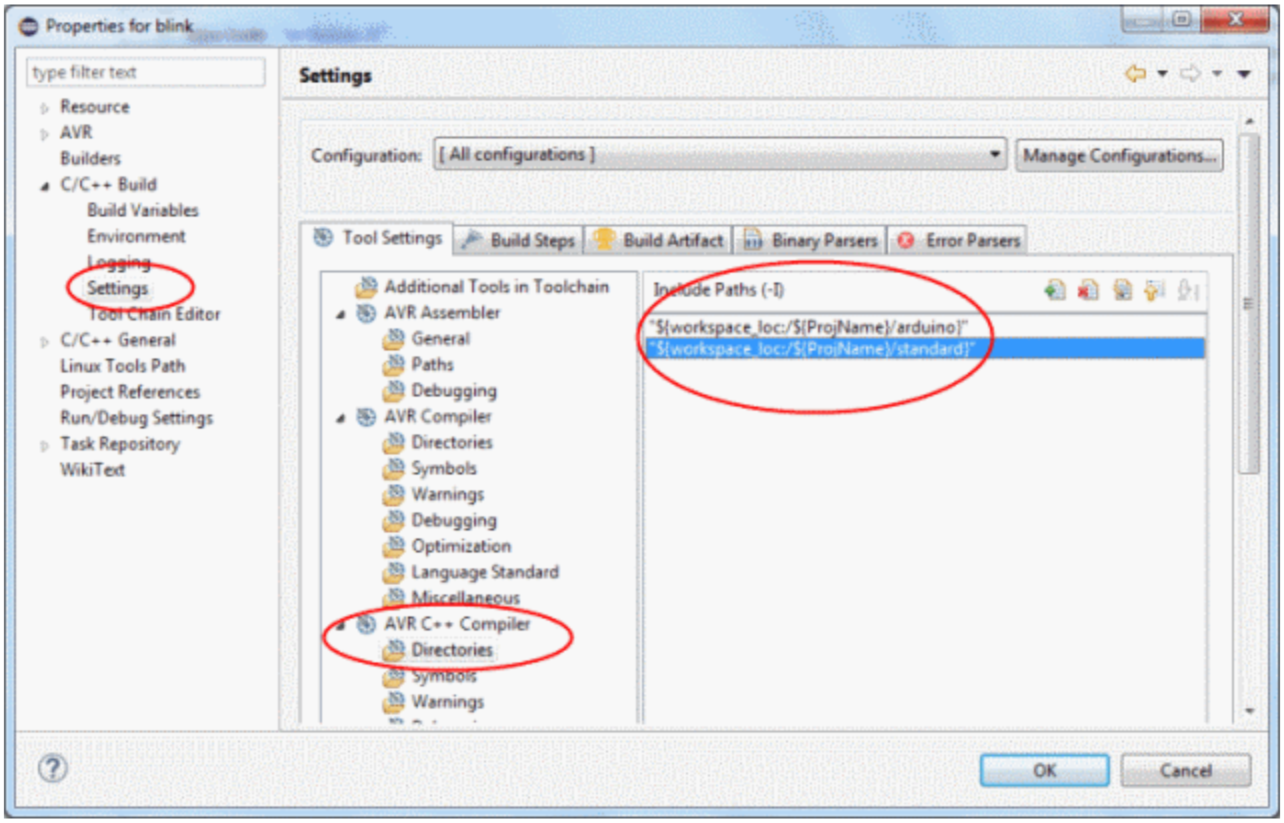
Configuration: [All configurations]

- [✓] Generate HEX file for Flash memory
- [ ] Generate HEX file for EEPROM
- [✓] Generate Extended Listing (Source + generated Assembler)
- [✓] Print Size
- [ ] AVRDude

Now select the sub-category **AVR C++ Compiler** > **Directories** and add the two Arduino folders which we have added to our project into the list.
The easiest way is to copy-paste the following two paths, including the quotation marks (click the small Add button and then paste one path; repeat for the other):

"${workspace_loc:/${ProjName}/arduino}"
"${workspace_loc:/${ProjName}/standard}"

Alternatively, click the Add button, then Workspace button in the Add directory path window. In the Folder selection window browse to [your_project]/arduino and click OK. Click OK again to close the Add directory path window. Repeat the same for the "standard" folder.

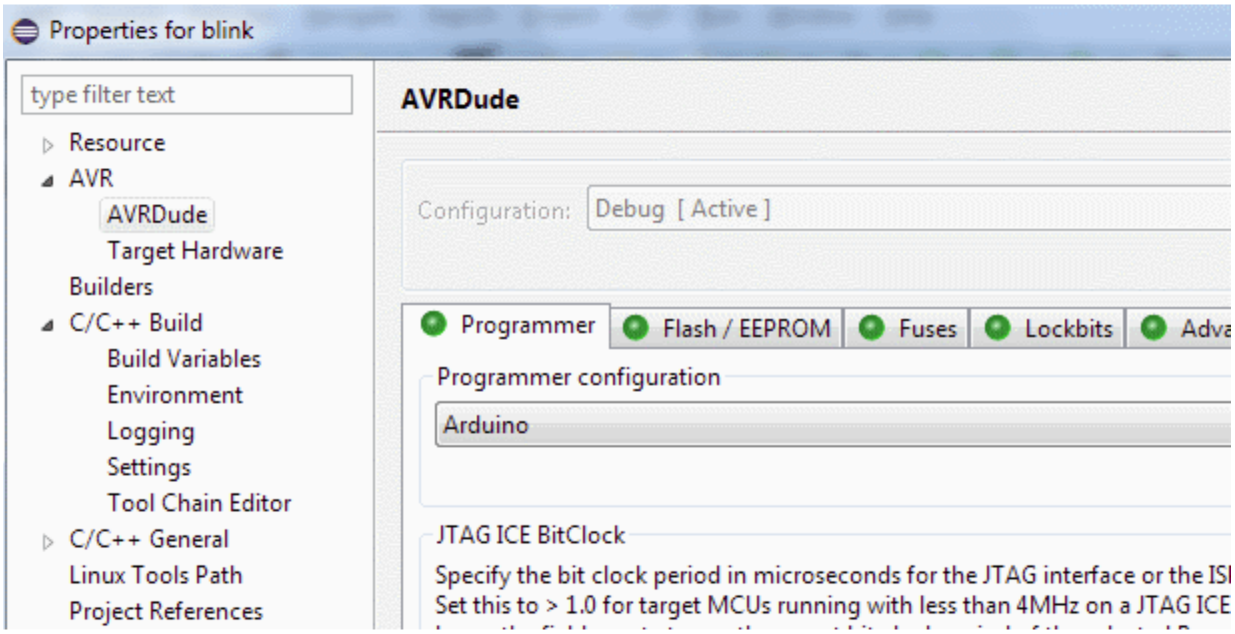


"${workspace_loc:/${ProjName}/arduino}"
"${workspace_loc:/${ProjName}/standard}"

This tells the compiler to search for source code in these locations.

Repeat the same step also for the directories of the C compiler in **AVR C Compiler** > Directories.

Still in the Preferences window, select AVR > AVRDude and in the "Programmer Configuration" list select Arduino. This is the programmer we have created earlier in the Set up the AVR Eclipse plugin section. If you do not see any item in the list please go back to that section and make sure you have done all the steps.



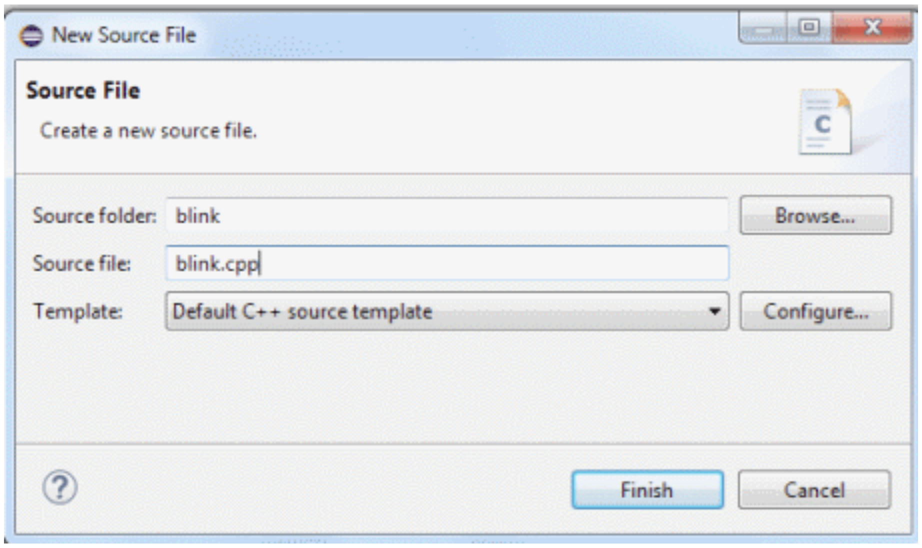
Configuration: Debug [Active]

Arduino

Specify the bit clock period in microseconds for the JTAG interface or the ISI
Set this to > 1.0 for target MCUs running with less than 4MHz on a JTAG ICE

Now we will add source file with our setup() and loop() functions.

Right-click the project and select New > Source File.
Enter some name for the file with .cpp extension and click Finish. The name could be the same as the project's name or sketch.cpp etc.



Source folder: blink
Source file: blink.cpp
Template: Default C++ source template

Click Finish to create the file. It will appear under the project in Project Explorer. Double-click it to open it.

Paste the following code into the new file:

```
#include "arduino.h"

void setup(void)
{
    pinMode(13, OUTPUT);
}

void loop(void)
{
    digitalWrite(13, HIGH);
    delay(200);
    digitalWrite(13, LOW);
    delay(2500);
}
```

Save the file.

Note: In default configuration Eclipse does not save the files before build. I recommend enabling auto-save in Window > Preferences > General > Workspace: "Save automatically before build". Without this you will often forget to save the changes after modifying your code and will be actually building the old version, without these changes.

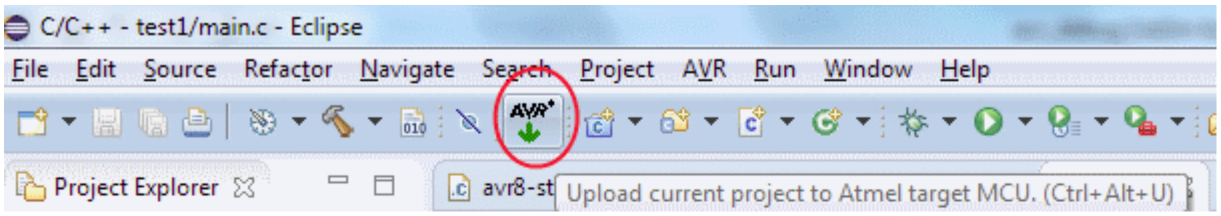
Build your project.

The project should built without errors, there will be some warnings about delay functions which have no effect on our program. The build may take little longer first time because all the Arduino core functions are built, but later, the compiler is smart enough to build only changed files, so it takes only few seconds to build the program.

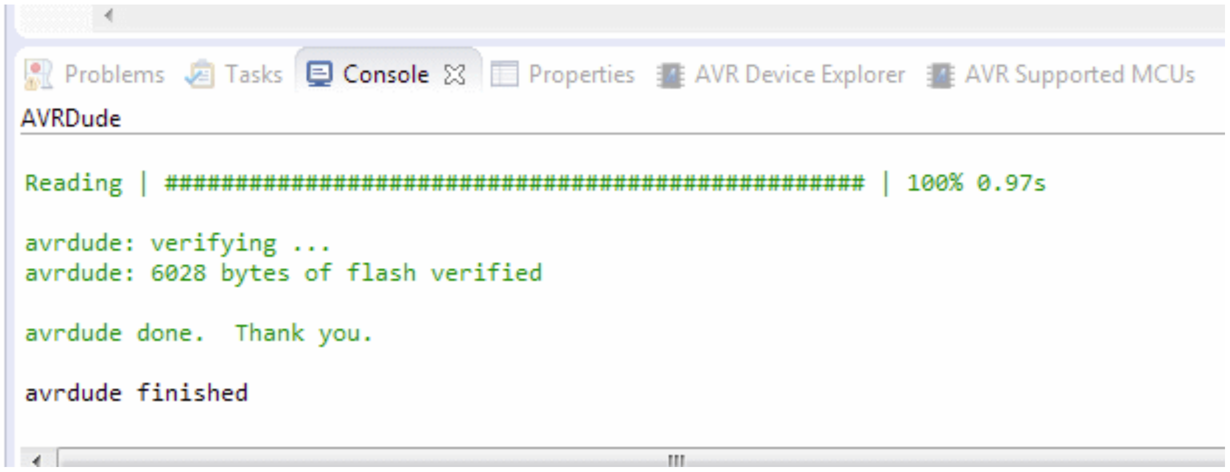
Now upload your project to your Arduino board.

First, connect the Arduino board to your computer. Give the computer enough time recognize the board.

Now click the AVR icon in Eclipse toolbar.


C/C++ - test1/main.c - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
avr8-st
Upload current project to Atmel target MCU. (Ctrl+Alt+U)

Your program will be uploaded to the board. Check the results in the Console window in lower part of Eclipse. There will be some messages from AVRDude. It should end with "avrdude done. Thank you."

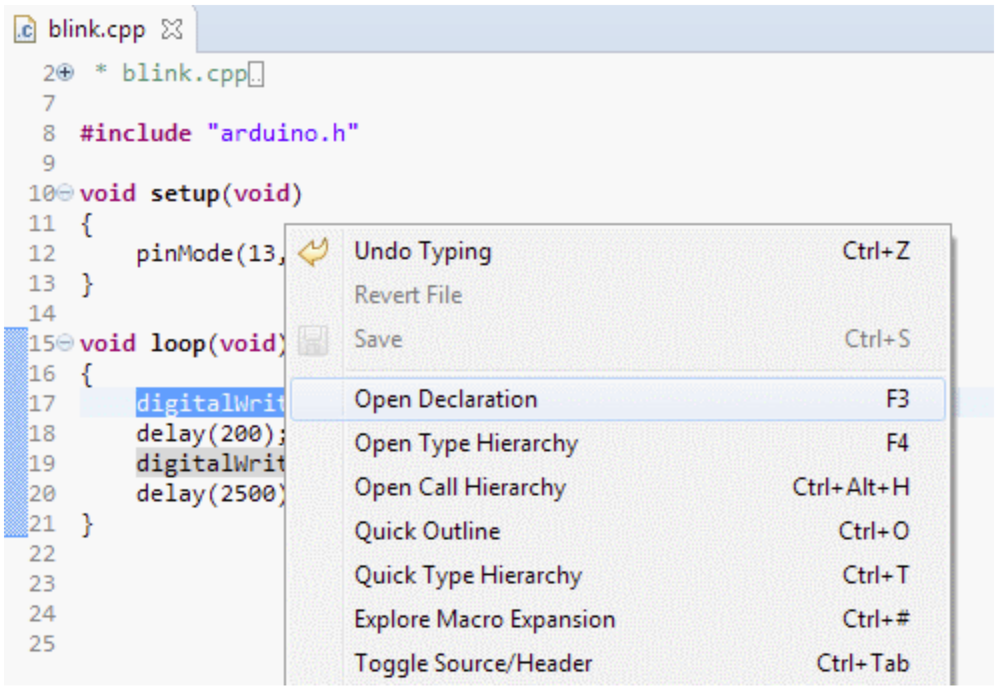


```
Reading | ################################################## | 100% 0.97s

avrdude: verifying ...
avrdude: 6028 bytes of flash verified

avrdude done.  Thank you.

avrdude finished
```

You should now see that the LED on your Arduino is blinking. This means our program is running.

If you get an error message "port is blocked" you probably have wrong COM port number in the AVR Eclipse plugin configuration. To fix this go to Window > Preferences > AVR > AVRDude. In the "Programmer configurations" list select your Arduino item and click Edit. Make sure the number of COM port (for example, //./COM15) is correct. You can find this number either in the Arduino IDE or in Hardware Manager of your computer.

You are now ready to develop your programs in Eclipse. Don't miss some of the cool features of a professional IDE, for example, you can look at the source code of the Arduino functions: just select the function name, right-click and select Open Declaration.



```
 * blink.cpp

#include "arduino.h"

void setup(void)
{
    pinMode(13,
}

void loop(void)
{
    digitalWrit
    delay(200);
    digitalWrit
    delay(2500)
}
```

blink.cpp
Undo Typing Ctrl+Z
Revert File
Save Ctrl+S
Open Declaration F3
Open Type Hierarchy F4
Open Call Hierarchy Ctrl+Alt+H
Quick Outline Ctrl+O
Quick Type Hierarchy Ctrl+T
Explore Macro Expansion Ctrl+#
Toggle Source/Header Ctrl+Tab

Points of Interest

In most tutorials I have seen the Arduino core functions are pre-build into a static library first and then this library is added to your project(s). In this tutorial we link the Arduino files directly to the project and build them together with our code. In my opinion this is easier to set up, easier to update and the build does not take unbearably long, so I consider this a better option than using static library.

If you update to a new version of Arduino software, you just need to change the value for the ArduinoPath variable and all the projects should work with the new version.

History

2015-06-23 First version.

License

Share

About the Author

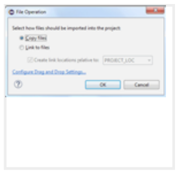
Jan Dolinay
Tomas Bata University in Zlin
Czech Republic

Works at Tomas Bata University in Zlin, Czech Republic. Interested in programming in general and especially programming microcontrollers.

You may also be interested in...

Arduino Unleashed

Microsoft Guide to Modern Dev/Test


Debugger for Arduino

How-To Intel IoT Code Samples: Watering System

Interfacing an Arduino with LCDs

Capturing Customer Information from Driver's Licenses using LEADTOOLS

Comments and Discussions

42 messages have been posted for this article Visit **http://www.codeproject.com/Articles/1003347/Creating-Arduino-programs-in-Eclipse** to post and view comments on this article, or click **here** to get a print view with messages.