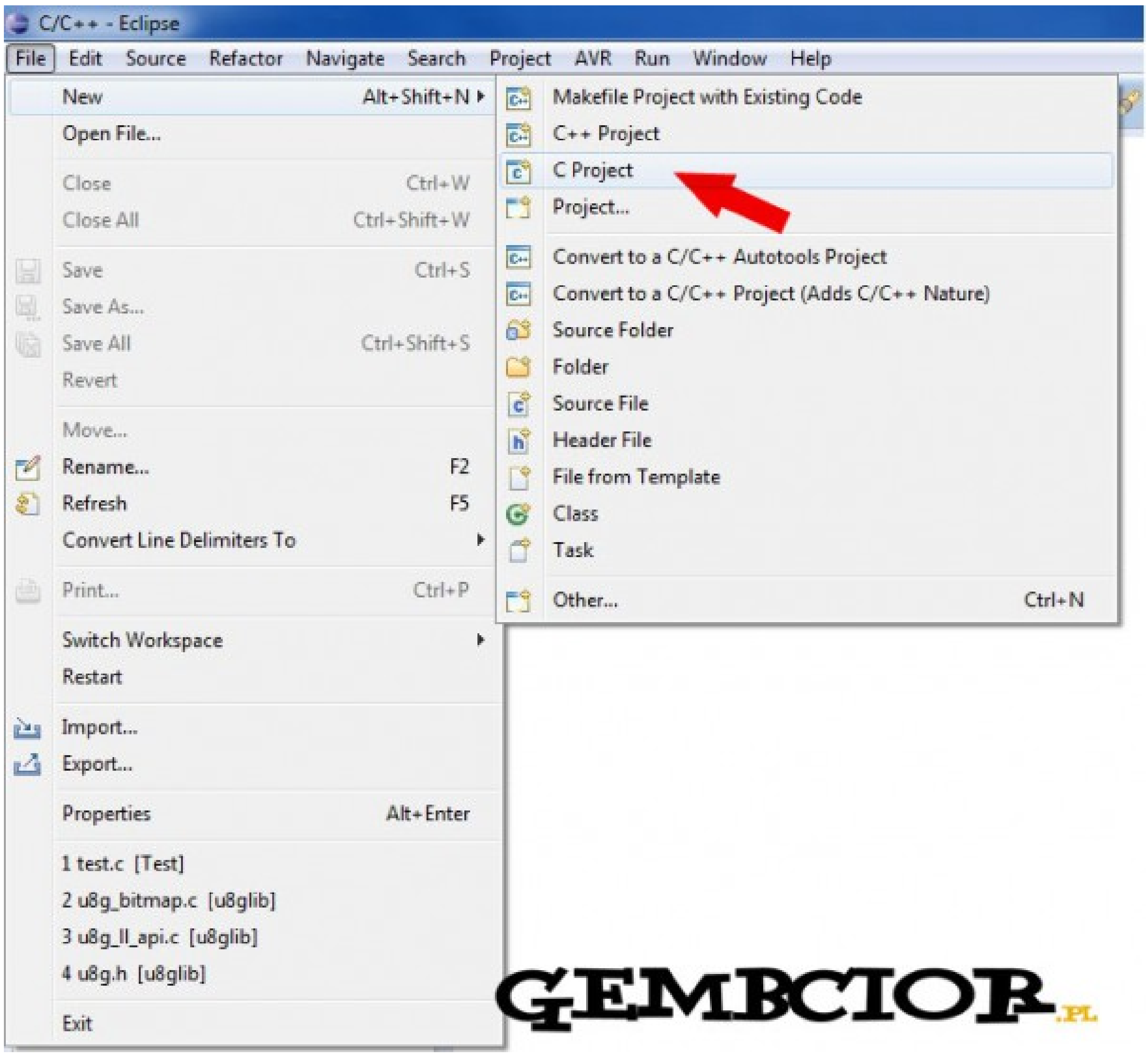
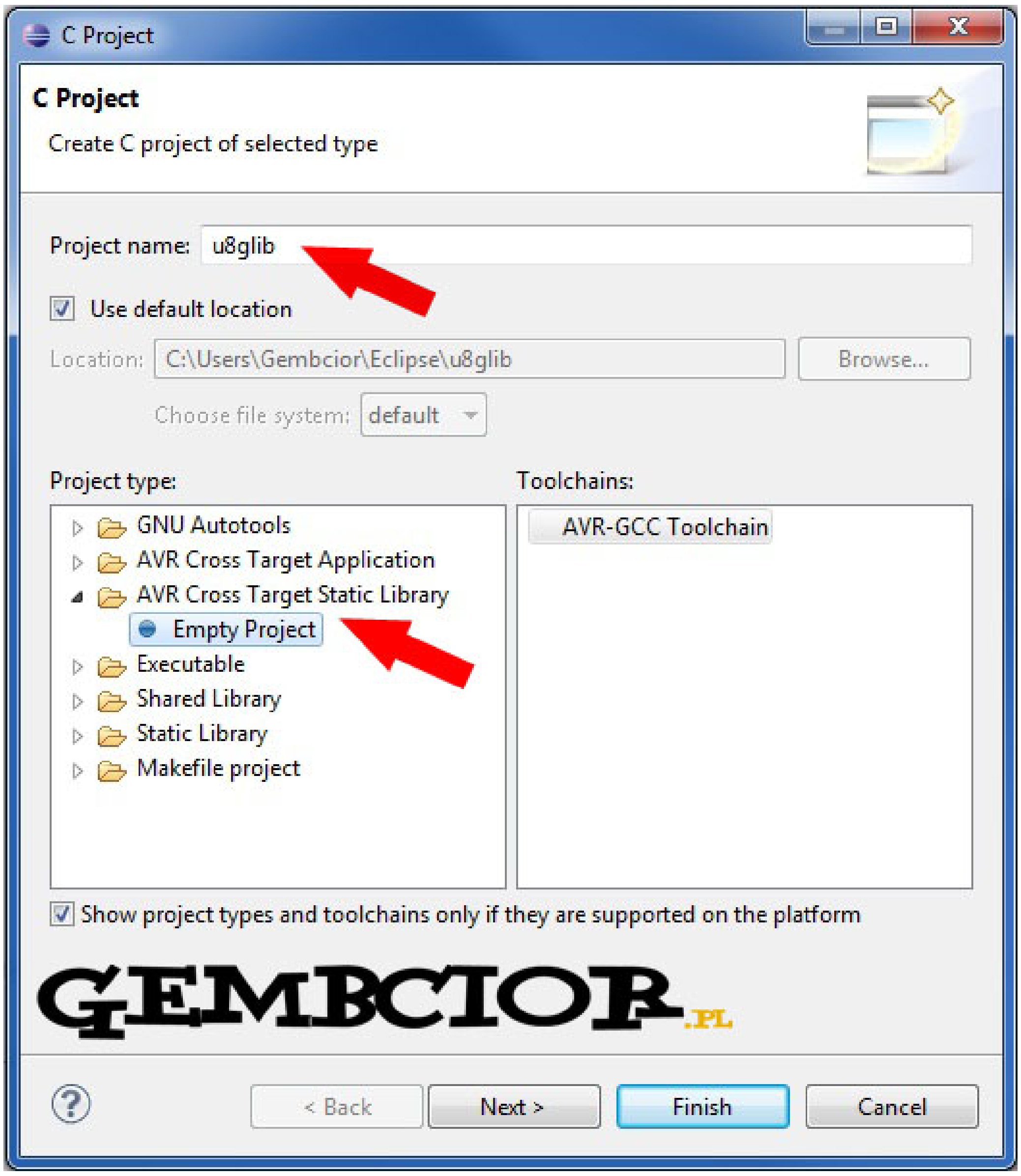
Recently, one of my readers requested me to write an instruction how to run graphic library U8glib for Eclipse IDE with AVR plugin. Installation of the library is very similar to installing Atmel Studio plugin.

1. Download and unzip the archive with our library which can be downloaded HERE

2. Run Eclipse and create a New C Project.



- Select AVR Cross Target Static Library
- and give name to the project, ex. ‘u8glib’.


C Project

C Project

Create C project of selected type

Project name: u8glib

[✓] Use default location

Location: C:\Users\Gembcior\Eclipse\u8glib

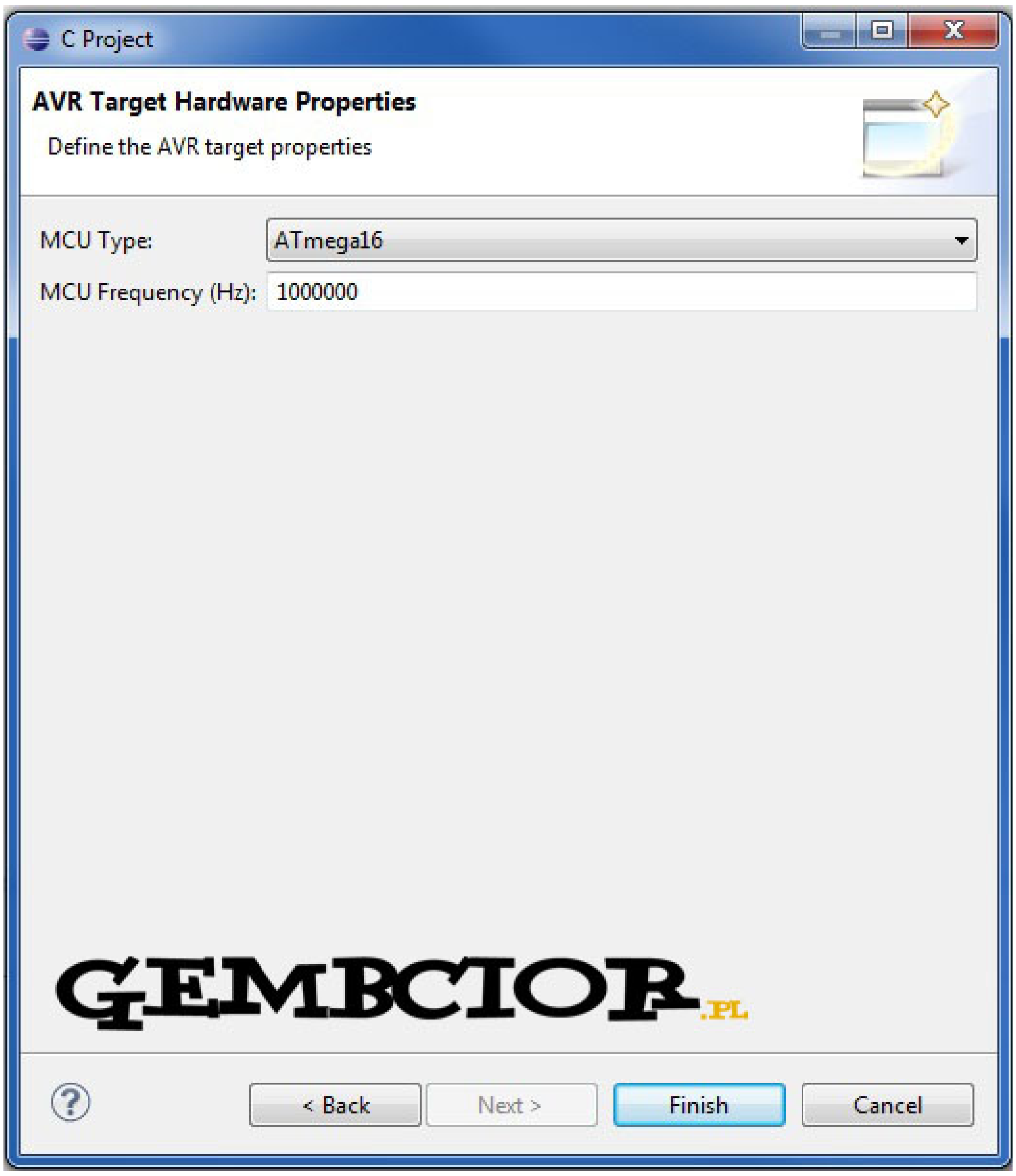
Choose file system: default

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

Toolchains:
- AVR-GCC Toolchain

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

- Click 2x Next and select your microcontroller type and frequency.
- I chose ATmega16 on which I am testing this library and set my frequency to 1MHz.



AVR Target Hardware Properties

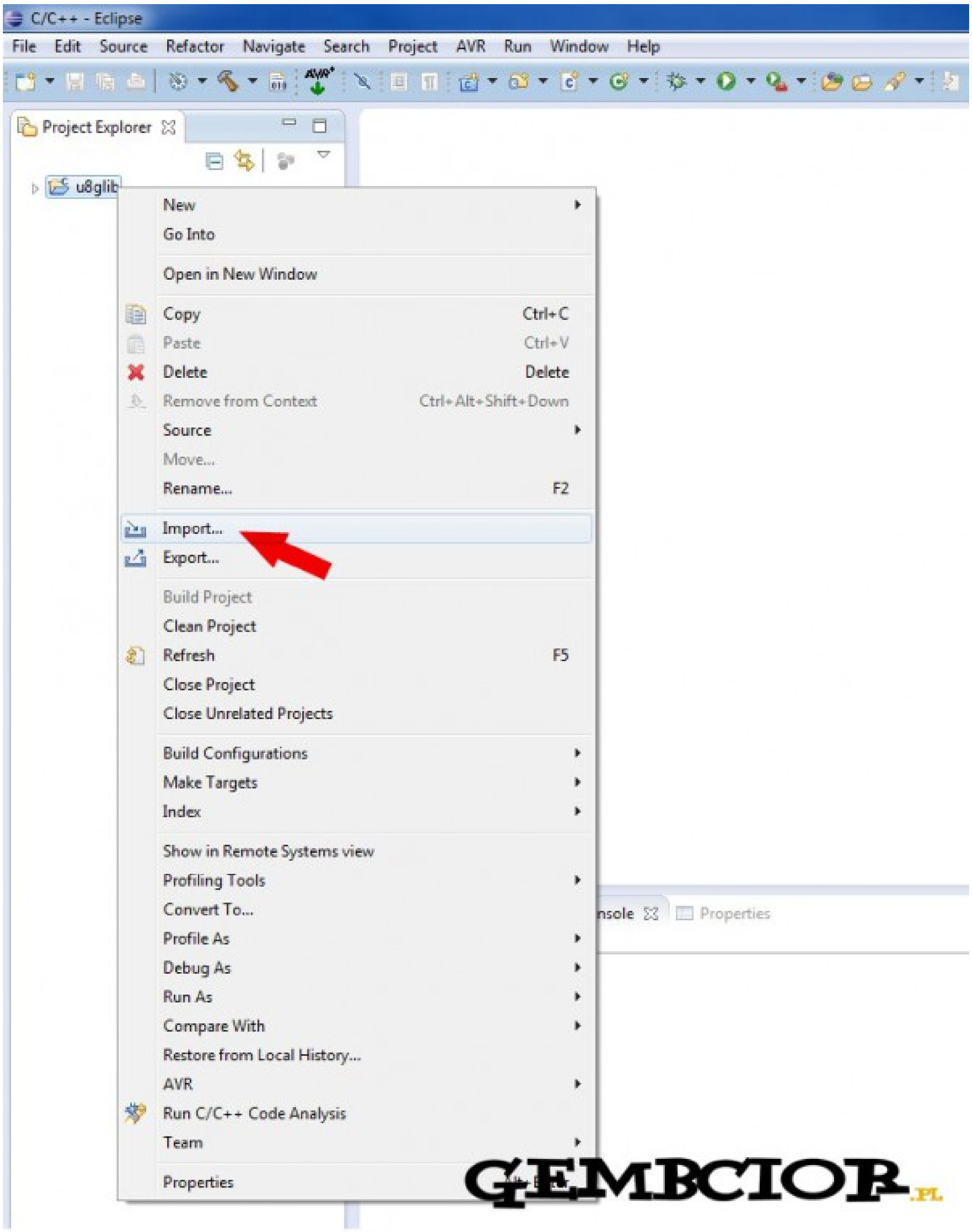
Define the AVR target properties

MCU Type: ATmega16

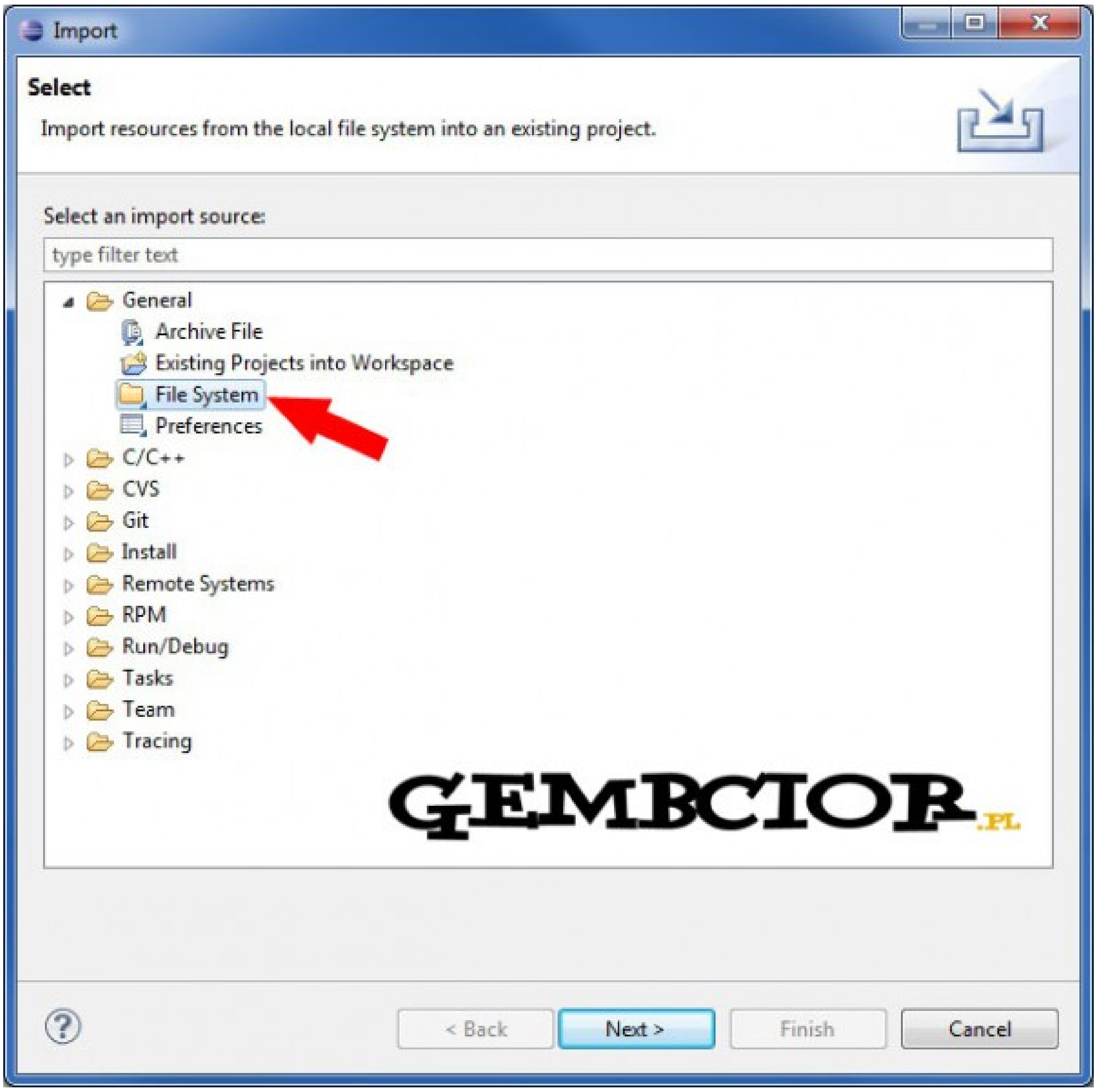
MCU Frequency (Hz): 1000000

3. Adding library files

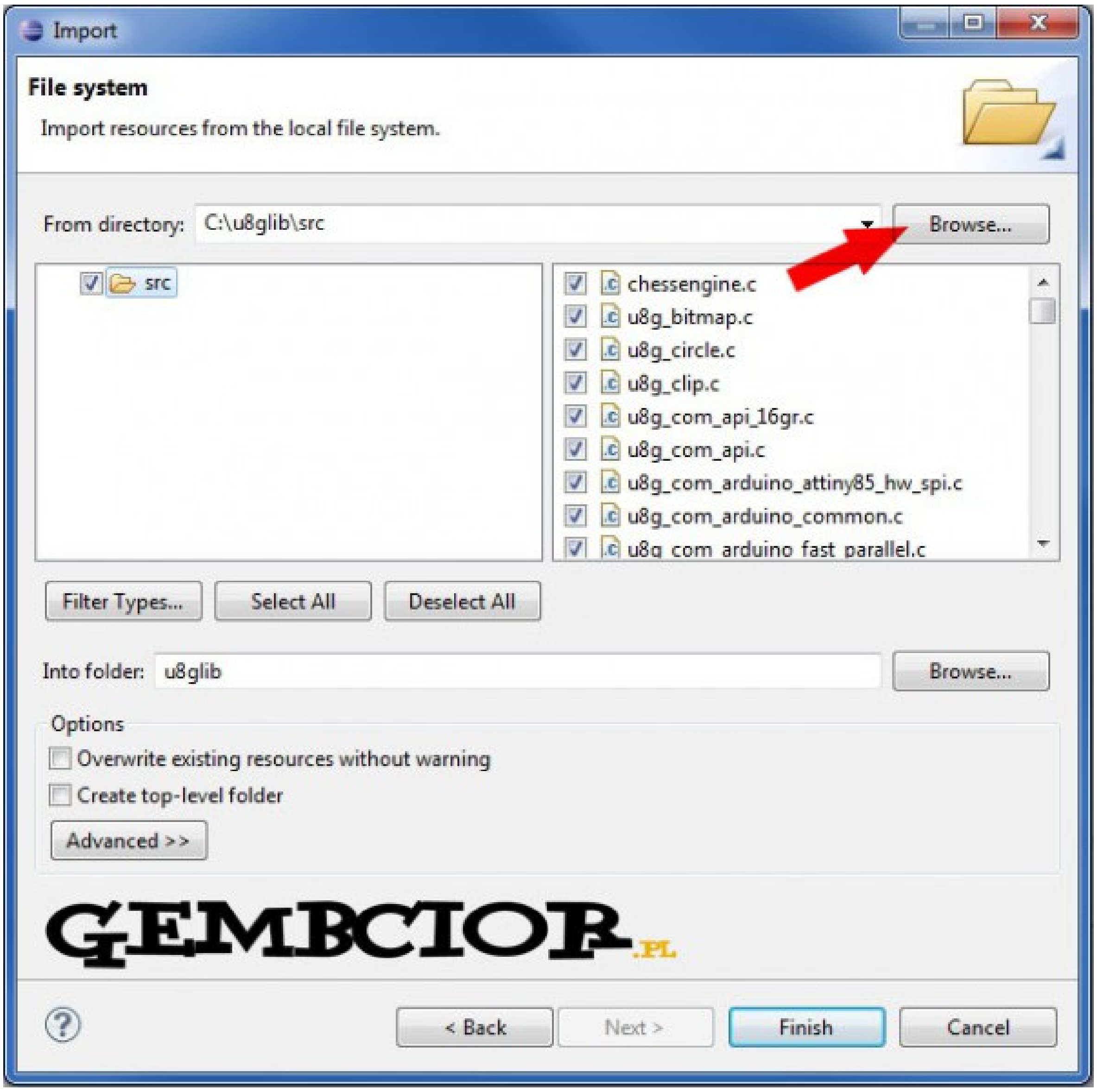
- Right click our project in Project Explorer
- and choose Import from the list.



- A window will pop up in which you choose the import source.
- Expand General branch and select File System.



- On the next window click Browse
- and choose the path where library files are located. On my computer it was C:\u8glib\src
- Select all files and click Finish.



Import

File system

Import resources from the local file system.

From directory: C:\u8glib\src

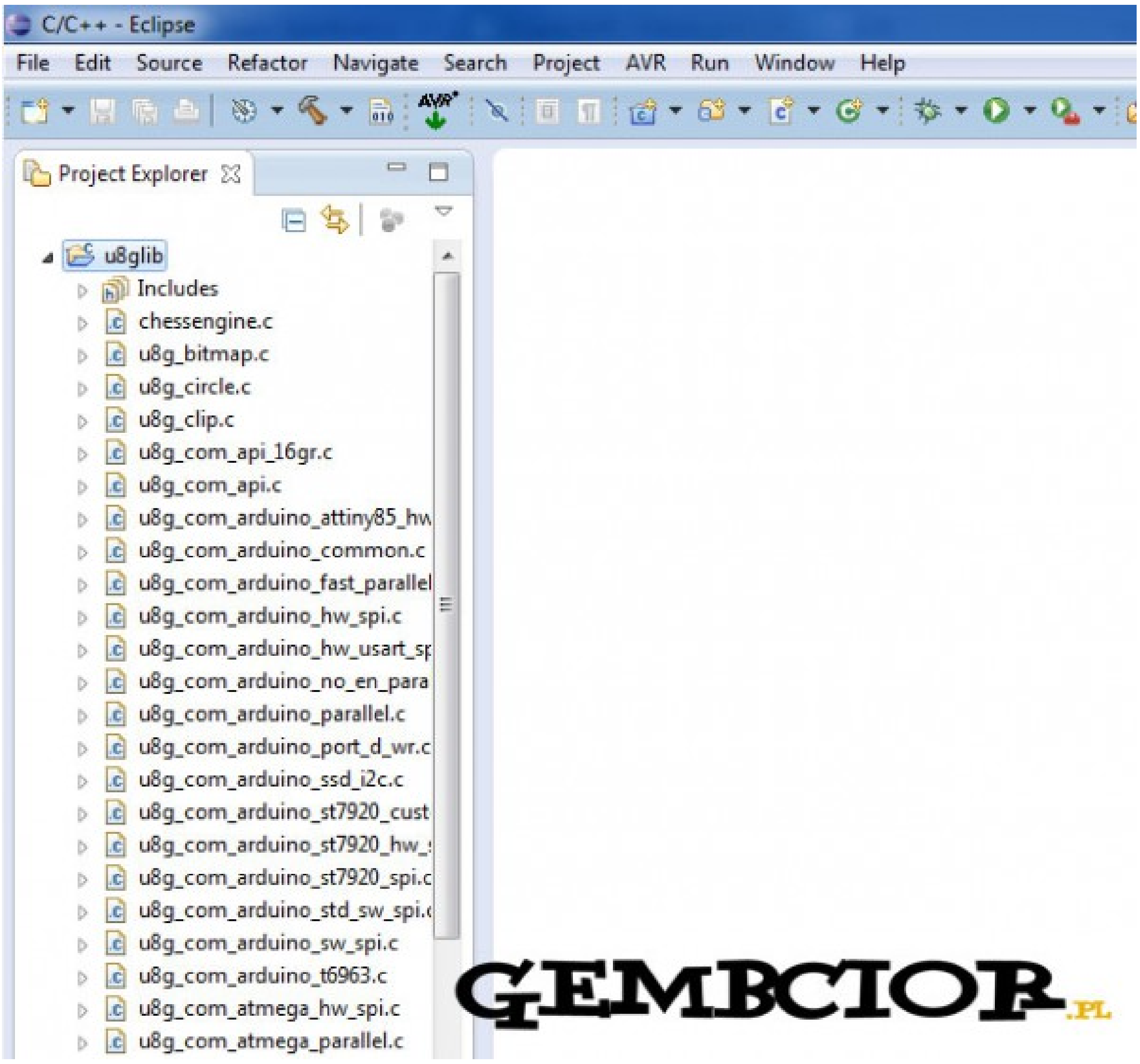
☑ src

☑ chessengine.c
☑ u8g_bitmap.c
☑ u8g_circle.c
☑ u8g_clip.c
☑ u8g_com_api_16gr.c
☑ u8g_com_api.c
☑ u8g_com_arduino_attiny85_hw_spi.c
☑ u8g_com_arduino_common.c
☑ u8g_com_arduino_fast_parallel.c

Into folder: u8glib

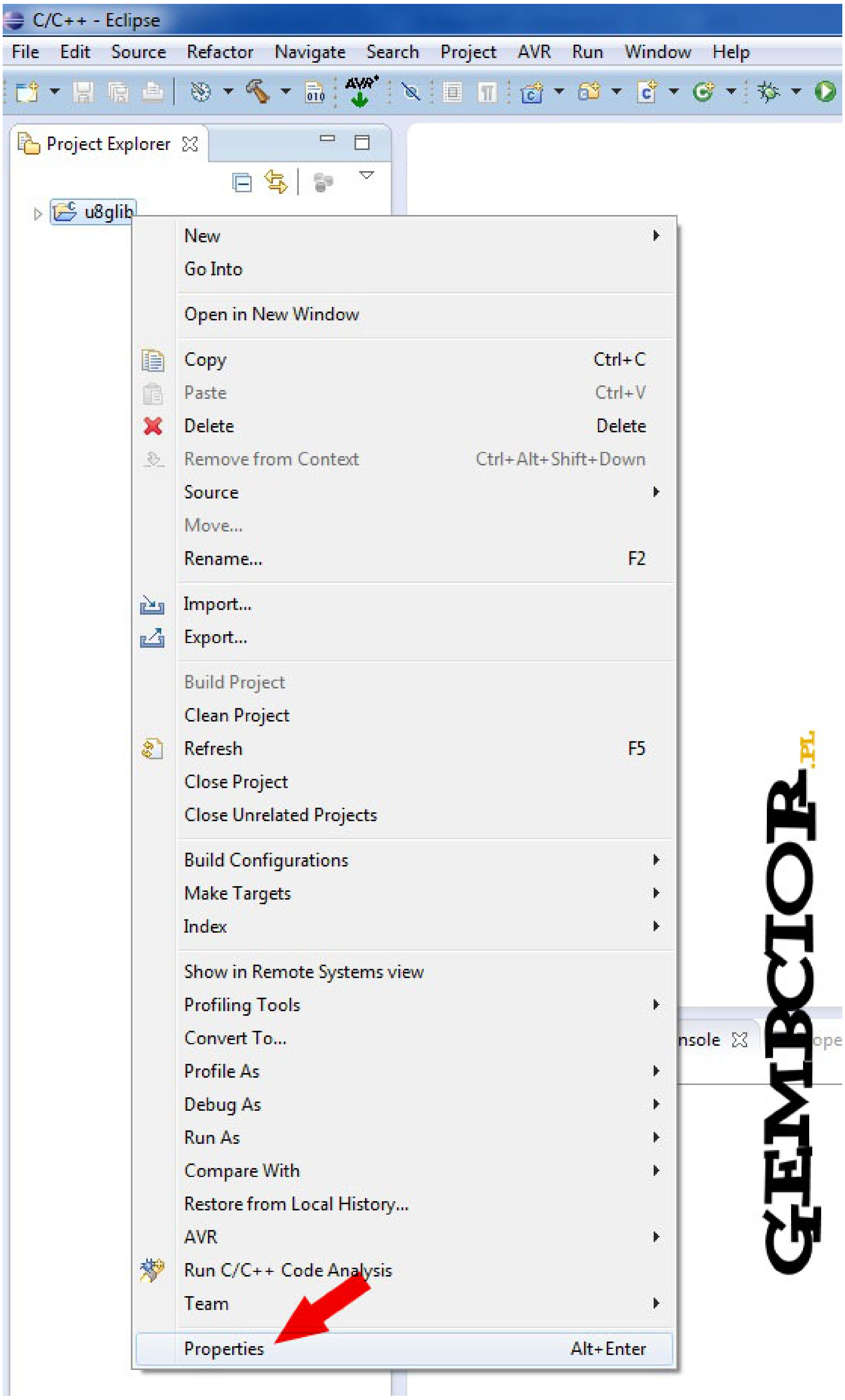
Options
☐ Overwrite existing resources without warning
☐ Create top-level folder

- Project Explorer should look like this:


C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
u8glib
Includes
chessengine.c
u8g_bitmap.c
u8g_circle.c
u8g_clip.c
u8g_com_api_16gr.c
u8g_com_api.c
u8g_com_arduino_attiny85_hw
u8g_com_arduino_common.c
u8g_com_arduino_fast_parallel
u8g_com_arduino_hw_spi.c
u8g_com_arduino_hw_usart_sp
u8g_com_arduino_no_en_para
u8g_com_arduino_parallel.c
u8g_com_arduino_port_d_wr.c
u8g_com_arduino_ssd_i2c.c
u8g_com_arduino_st7920_cust
u8g_com_arduino_st7920_hw_
u8g_com_arduino_st7920_spi.c
u8g_com_arduino_std_sw_spi.
u8g_com_arduino_sw_spi.c
u8g_com_arduino_t6963.c
u8g_com_atmega_hw_spi.c
u8g_com_atmega_parallel.c
GEMBCIOR.PL

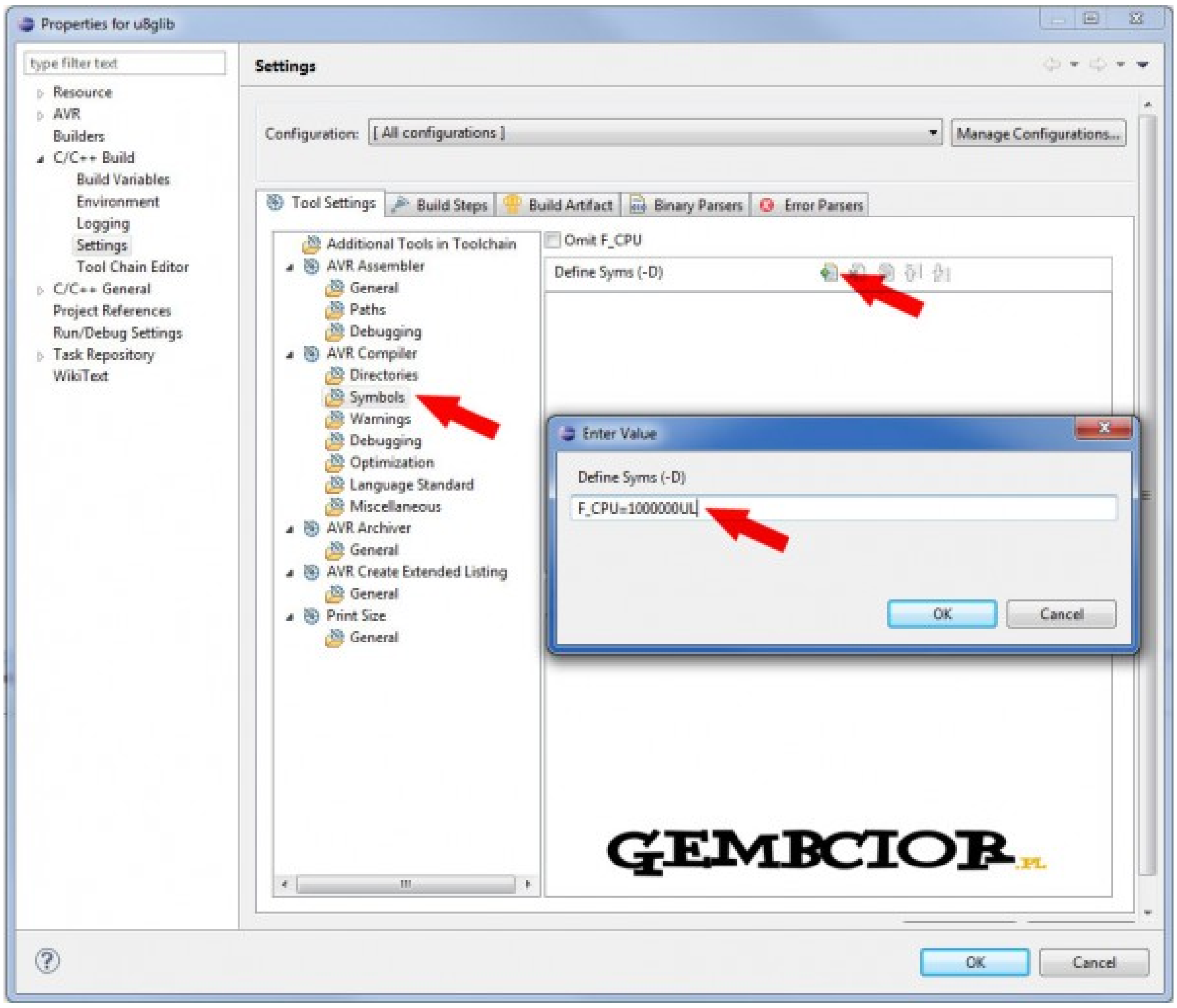
4. Editing project properties

- Right click our project in Project Explorer and choose Properties from the bottom of the list.



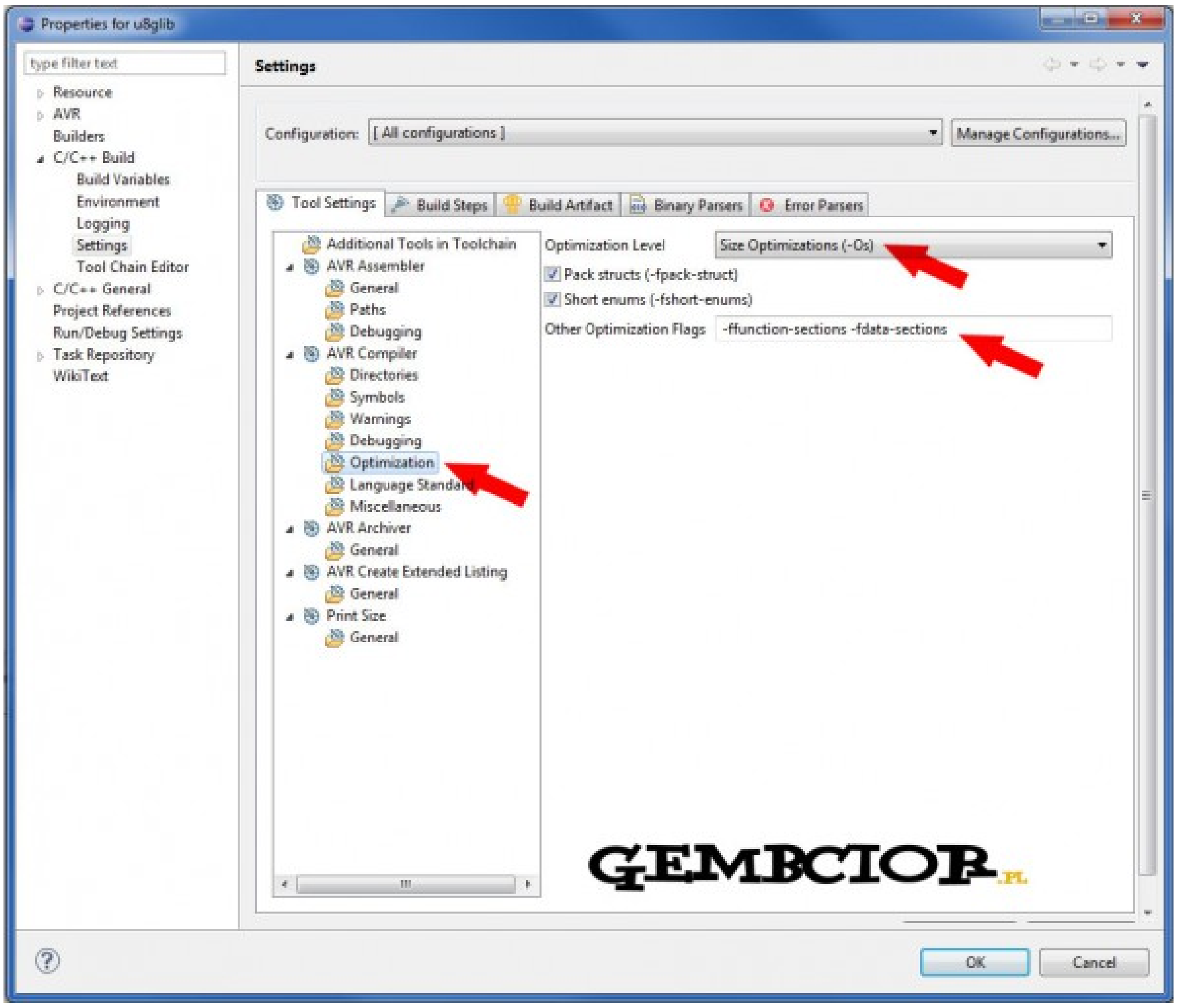
- Choose C/C++ Build -> Settings from the new window.
- Then, choose All configurations from Configuration listbox at the top of the window.
- Then, select Tool Settings -> AVR Compiler -> Symbols and define new symbol **F_CPU=1000000UL** which is responsible for our microcontroller clock frequency. As I

mentioned earlier, I used 1MHz. This value is dependent on what frequency we plan to use in our project.



Define Syms (-D): F_CPU=1000000UL

- After that, choose Optimization from the tree on the left.
- Select Size Optimizations (-Os) from the Optimization Level listbox.
- We also need to add additional flags. In Other Optimization Flags type in: **-ffunction-sections -fdata-sections**

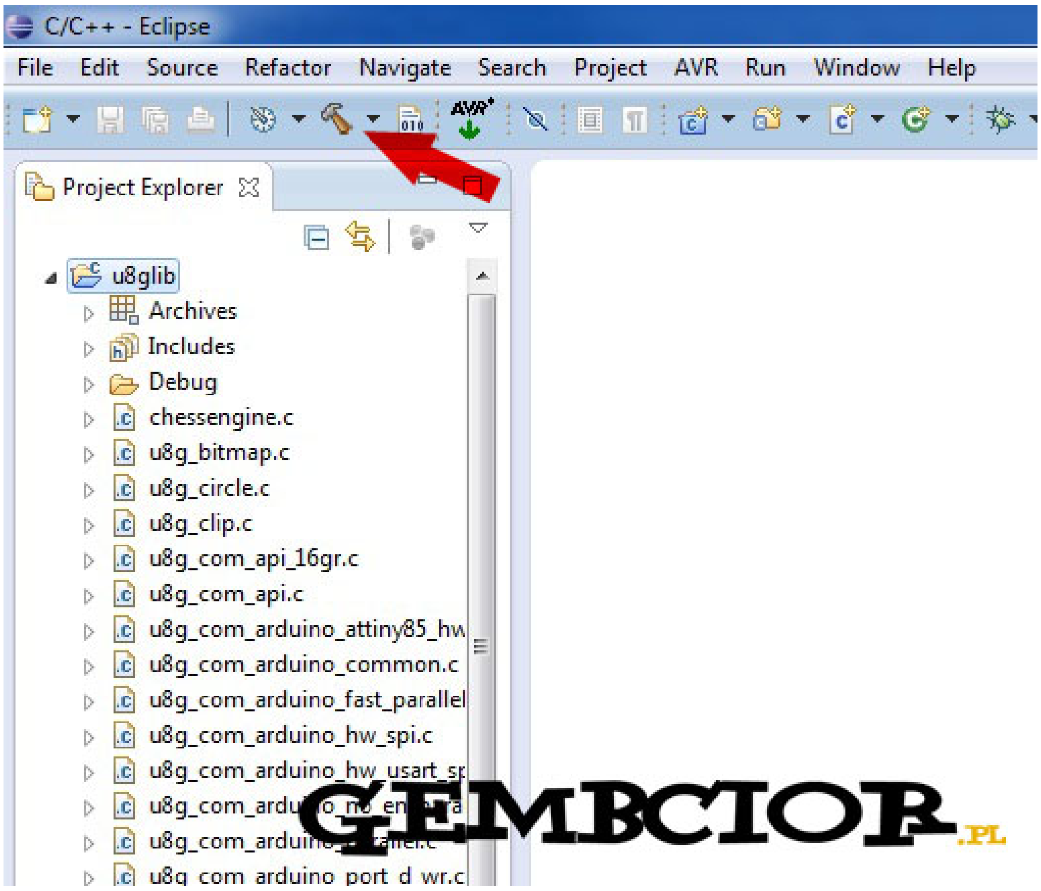


Configuration: [All configurations]

Optimization Level: Size Optimizations (-Os)
[✓] Pack structs (-fpack-struct)
[✓] Short enums (-fshort-enums)
Other Optimization Flags: -ffunction-sections -fdata-sections

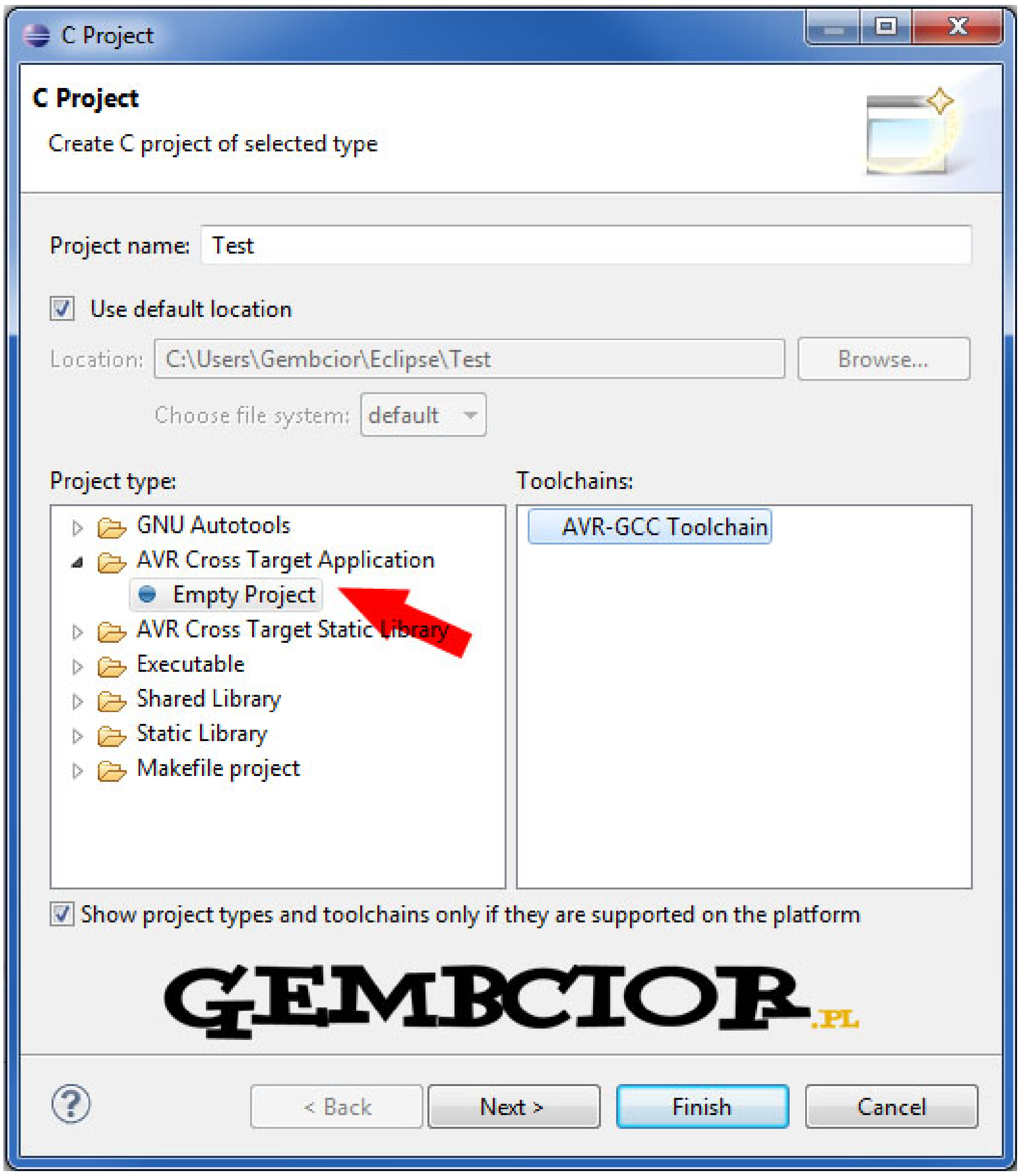
5. Building a library

- Now we have to build our library, so we can use it.
- Click on a hammer icon and wait. It should compile in about 30 seconds.


C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
u8glib
Archives
Includes
Debug
chessengine.c
u8g_bitmap.c
u8g_circle.c
u8g_clip.c
u8g_com_api_16gr.c
u8g_com_api.c
u8g_com_arduino_attiny85_hw
u8g_com_arduino_common.c
u8g_com_arduino_fast_parallel
u8g_com_arduino_hw_spi.c
u8g_com_arduino_hw_usart_sp
u8g_com_arduino_no_en_para
u8g_com_arduino_parallel.c
u8g_com_arduino_port_d_wr.c
GEMBCIOR.PL

6. Creating a project

- We create the project in the same way we did at the beginning of the tutorial, the only difference is Project Type, this time it is AVR Cross Target Application.


C Project

C Project

Create C project of selected type

Project name: Test

[✓] Use default location

Location: C:\Users\Gembcior\Eclipse\Test

Choose file system: default

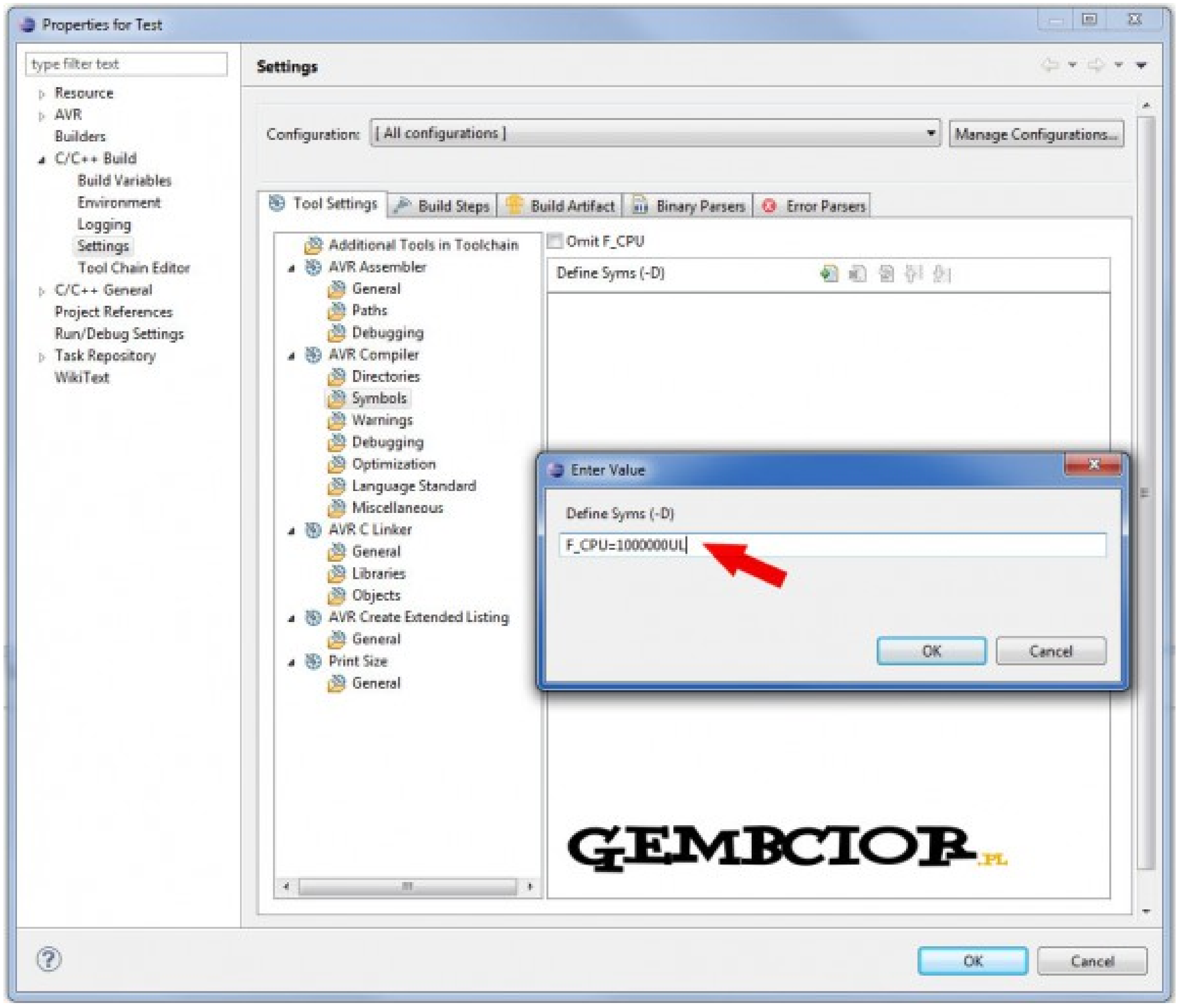
Project type:
- GNU Autotools
- AVR Cross Target Application
  - Empty Project
- AVR Cross Target Static Library
- Executable
- Shared Library
- Static Library
- Makefile project

Toolchains:
- AVR-GCC Toolchain

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

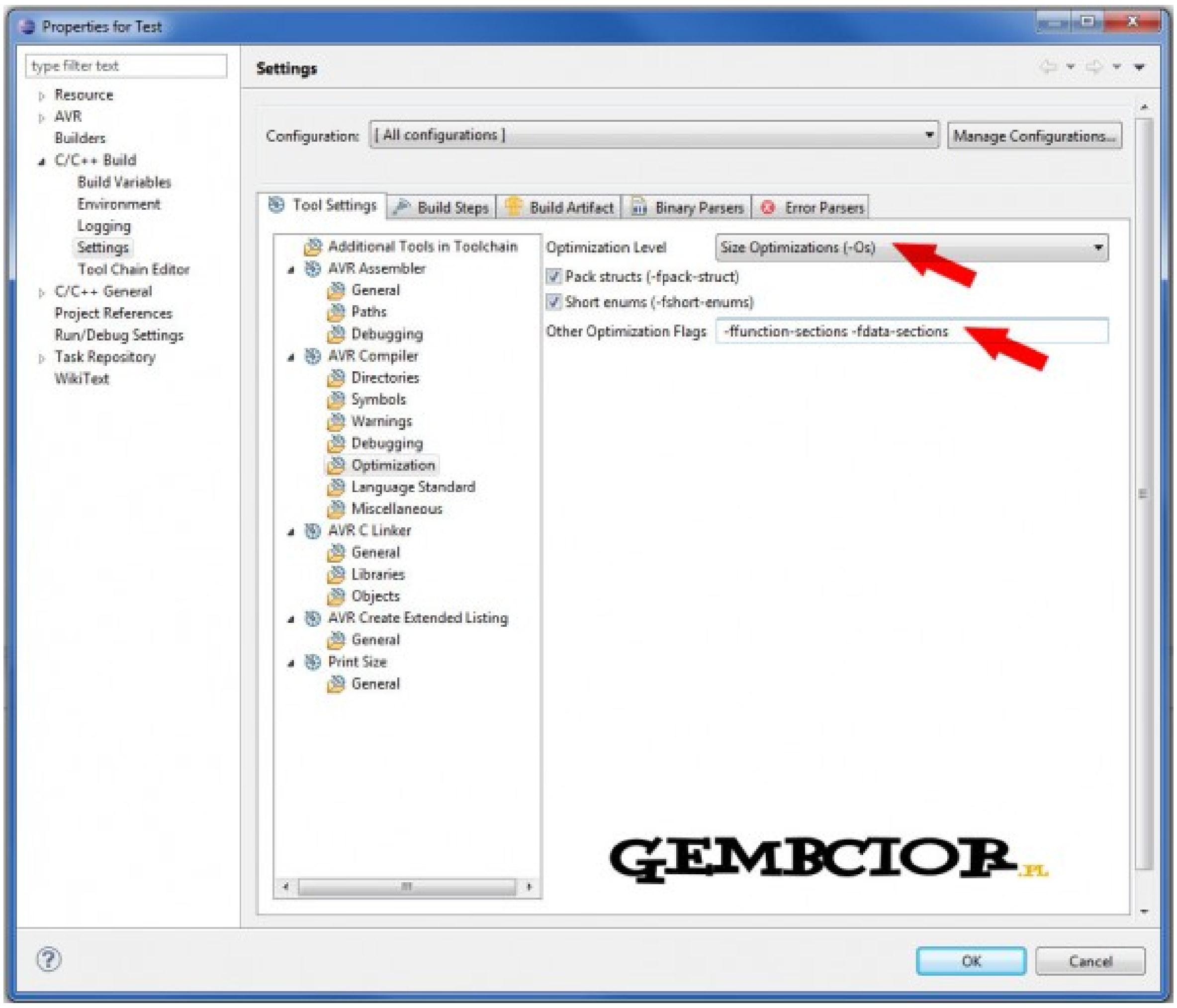
7. Editing a project settings

- As we did before, define a symbol **F_CPU=1000000UL**



Define Syms (-D): F_CPU=1000000UL

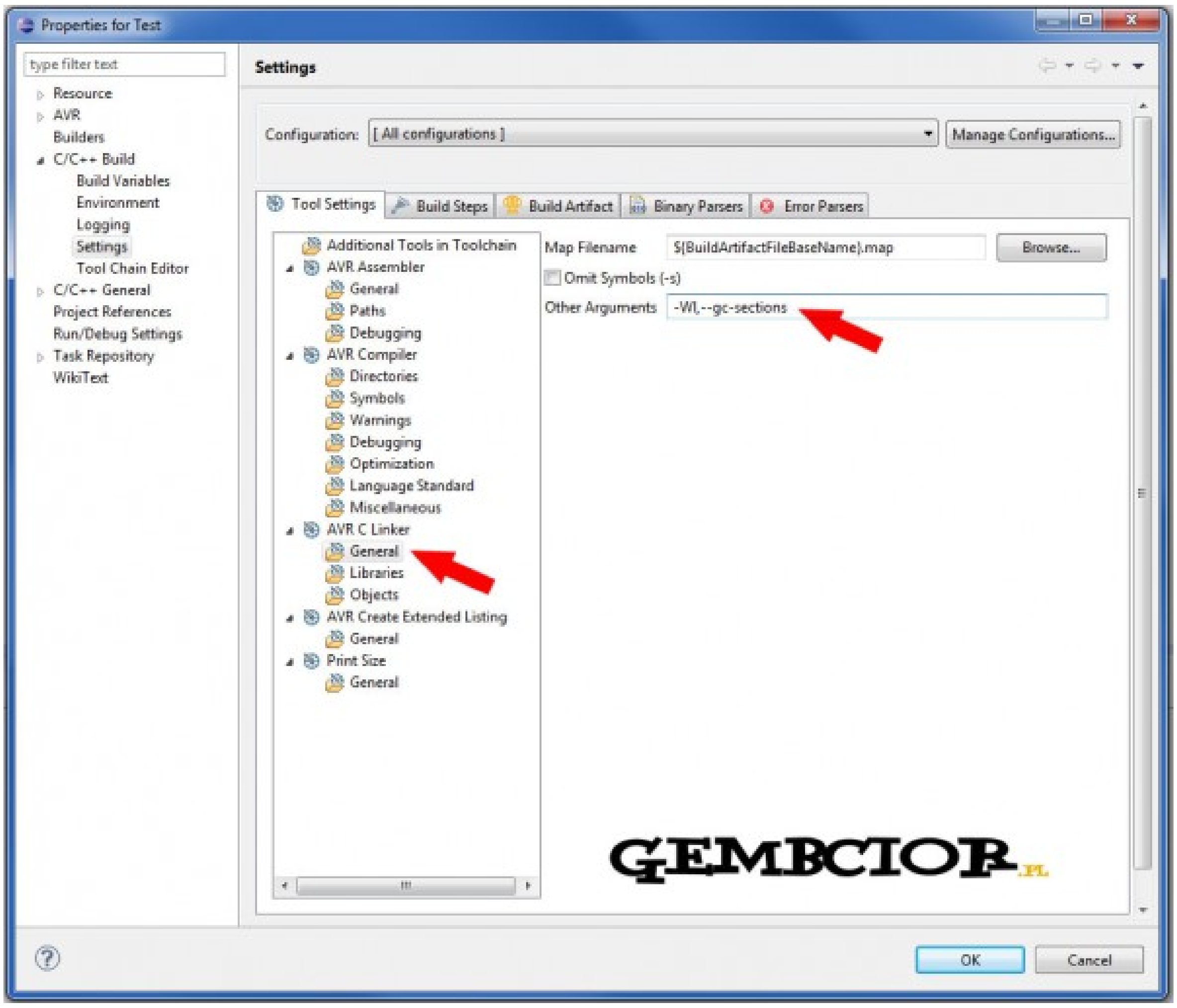
- Now change the optimization options, once again select Size Optimizations (-Os) from the Optimization Level listboxand add additional flags.
- In Other Optimization Flags type in: **-ffunction-sections -fdata-sections**



Configuration: [All configurations]

Optimization Level: Size Optimizations (-Os)
[✓] Pack structs (-fpack-struct)
[✓] Short enums (-fshort-enums)
Other Optimization Flags: -ffunction-sections -fdata-sections

- Then go to AVR C Linker -> Libraries and in Other Arguments type in: **-Wl,--gc-sections**



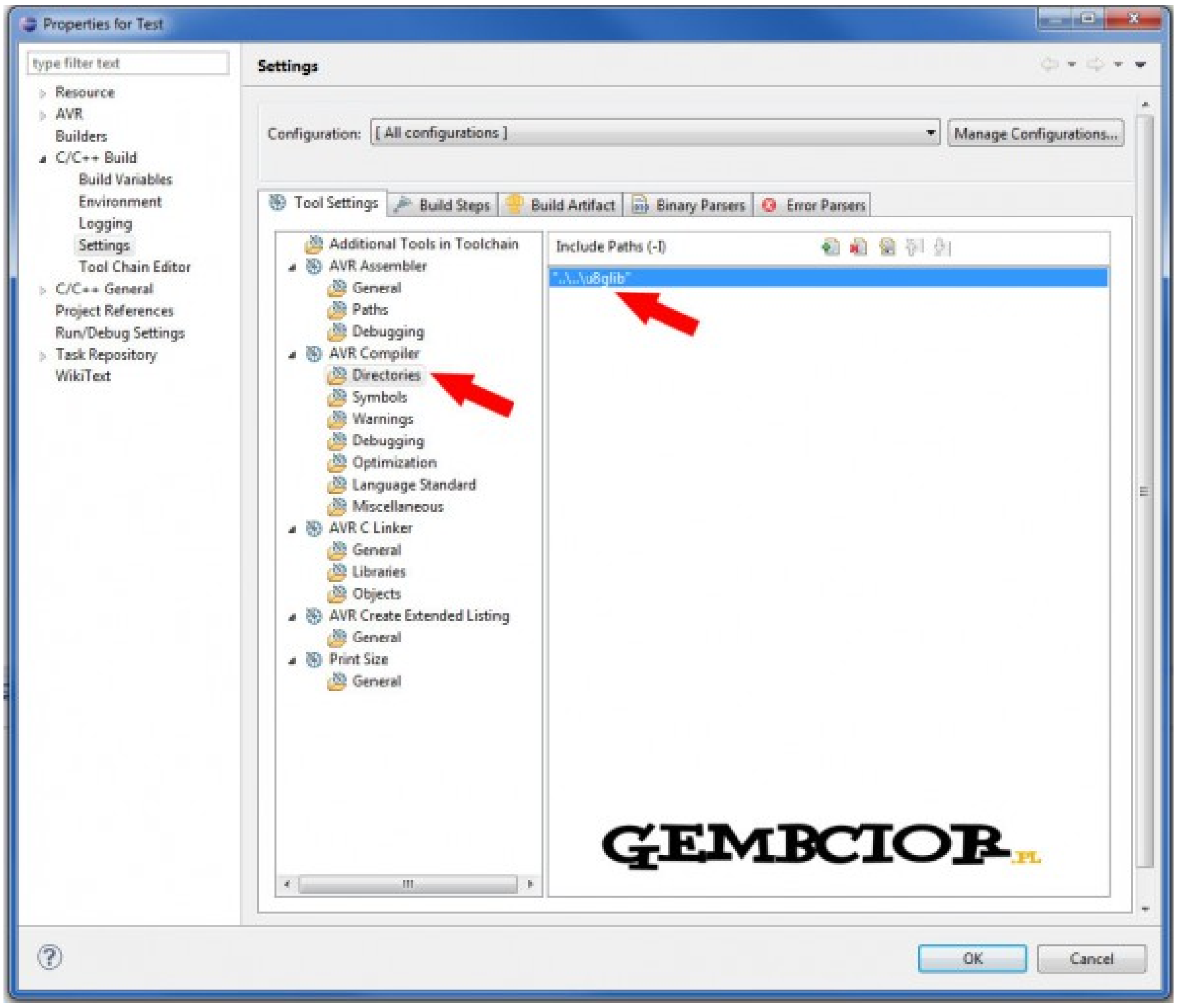
Configuration: [All configurations]

Map Filename: ${BuildArtifactFileBaseName}.map

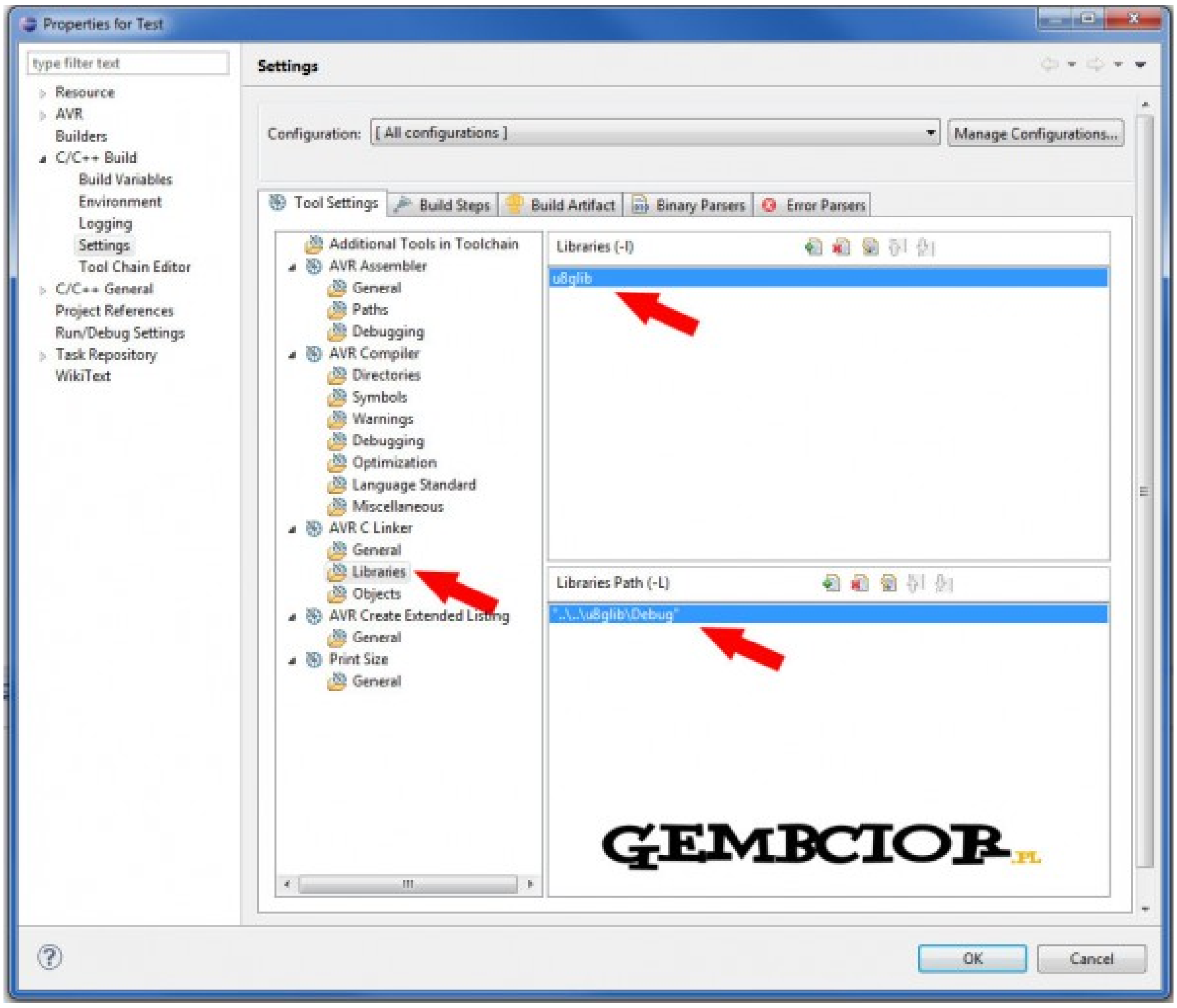
Omit Symbols (-s)

Other Arguments: -Wl,--gc-sections

- Go to AVR Compiler -> Directories and add a directory **..\..\u8glib** by clicking on a green plus icon.



- After that click AVR C Linker -> Libraries and add a name of the library which is u8glib. **u8glib**
- In Libraries Path add a directory for the library ..\..\u8glib\Debug



Configuration: [All configurations]

Libraries (-l): u8glib

Libraries Path (-L): "..\..\u8glib\Debug"

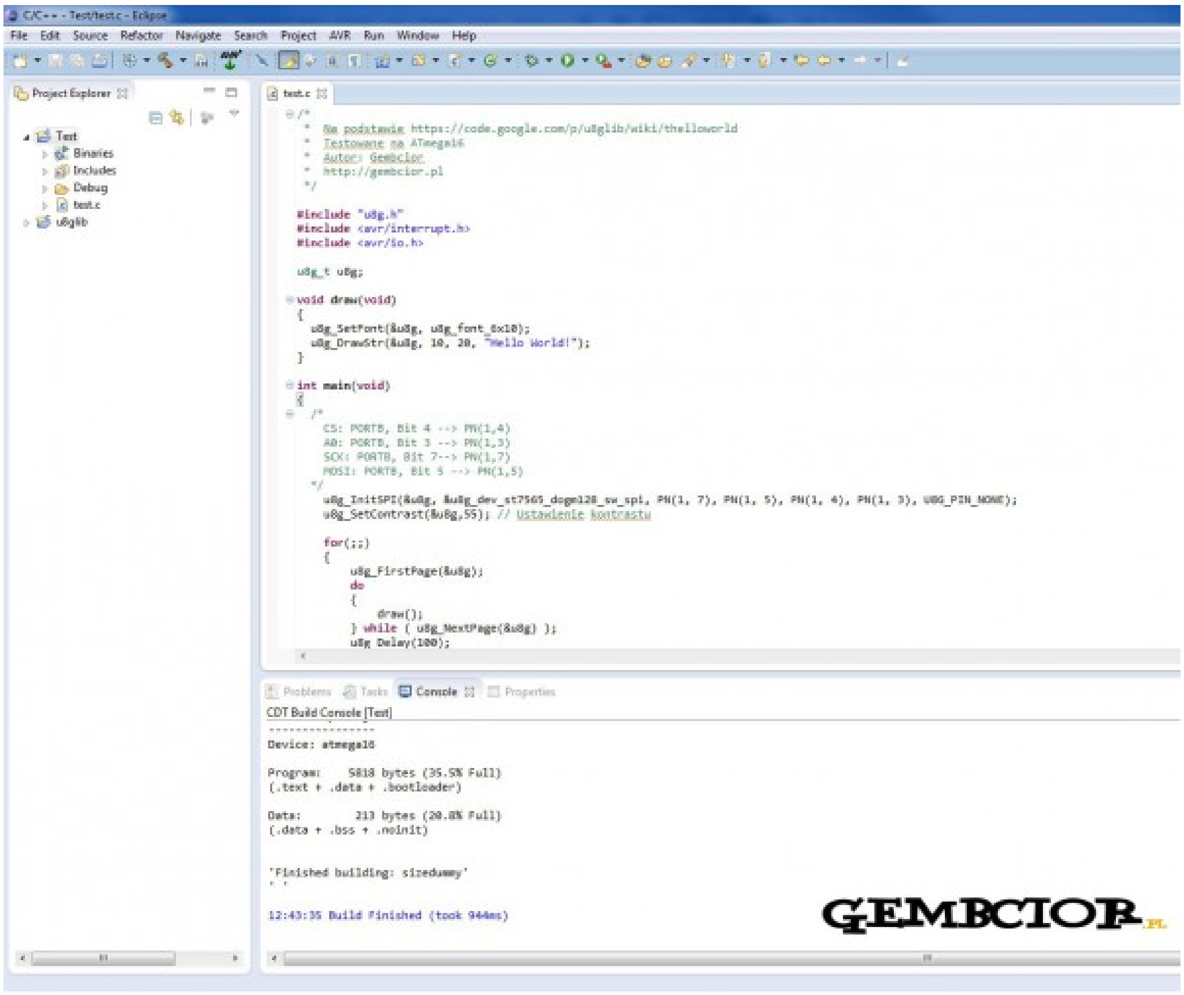
- All of the given directories are relative paths and points to the folder in which our u8glib library files are located.

8. Creating a sample program

- There's nothing left for us than to add a new file with .c extension to our project.
- We can now copy and paste a sample code there, like a classic "Hello world".

```c
/*
 *  Na podstawie https://code.google.com/p/u8glib/wiki/thelloworld
 *  Testowane na ATmega16
 *  Autor: Gembcior
 *  https://gembcior.pl
 */

#include "u8g.h"
#include <avr/interrupt.h>
#include <avr/io.h>

u8g_t u8g;

void draw(void)
{
  u8g_SetFont(&u8g, u8g_font_6x10);
  u8g_DrawStr(&u8g, 10, 20, "Hello World!");
}

int main(void)
{
  /*
    CS: PORTB, Bit 4 --> PN(1,4)
    A0: PORTB, Bit 3 --> PN(1,3)
    SCK: PORTB, Bit 7--> PN(1,7)
    MOSI: PORTB, Bit 5 --> PN(1,5)
  */
    u8g_InitSPI(&u8g, &u8g_dev_st7565_dogm128_sw_spi, PN(1, 7), PN(1,
    u8g_SetContrast(&u8g,55); // Ustawienie kontrastu

    for(;;)
    {
        u8g_FirstPage(&u8g);
        do
        {
            draw();
        } while ( u8g_NextPage(&u8g) );
        u8g_Delay(100);
    }
}
```

- Compile it and program the microcontroller.


C/C++ - Test/test.c - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
Test
Binaries
Includes
Debug
test.c
u8glib
test.c
Problems Tasks Console Properties
GEMBCIOR.PL

```
/*
 * Na podstawie https://code.google.com/p/u8glib/wiki/thelloworld
 * Testowane na ATmega16
 * Autor: Gembcior
 * http://gembcior.pl
 */

#include "u8g.h"
#include <avr/interrupt.h>
#include <avr/io.h>

u8g_t u8g;

void draw(void)
{
  u8g_SetFont(&u8g, u8g_font_6x10);
  u8g_DrawStr(&u8g, 10, 20, "Hello World!");
}

int main(void)
{
  /*
    CS: PORTB, Bit 4 --> PN(1,4)
    A0: PORTB, Bit 3 --> PN(1,3)
    SCK: PORTB, Bit 7--> PN(1,7)
    MOSI: PORTB, Bit 5 --> PN(1,5)
  */
    u8g_InitSPI(&u8g, &u8g_dev_st7565_dogm128_sw_spi, PN(1, 7), PN(1, 5), PN(1, 4), PN(1, 3), U8G_PIN_NONE);
    u8g_SetContrast(&u8g,55); // Ustawienie kontrastu

    for(;;)
    {
        u8g_FirstPage(&u8g);
        do
        {
            draw();
        } while ( u8g_NextPage(&u8g) );
        u8g_Delay(100);
```

CDT Build Console [Test]

```
-----------------
Device: atmega16

Program:    5818 bytes (35.5% Full)
(.text + .data + .bootloader)

Data:        213 bytes (20.8% Full)
(.data + .bss + .noinit)


'Finished building: sizedummy'
' '

12:43:35 Build Finished (took 944ms)
```

In this tutorial I was using Eclipse Kepler in 32bit version with AVR plugin in 2.4 Version and Atmel Toolchain 3.4.2.1573