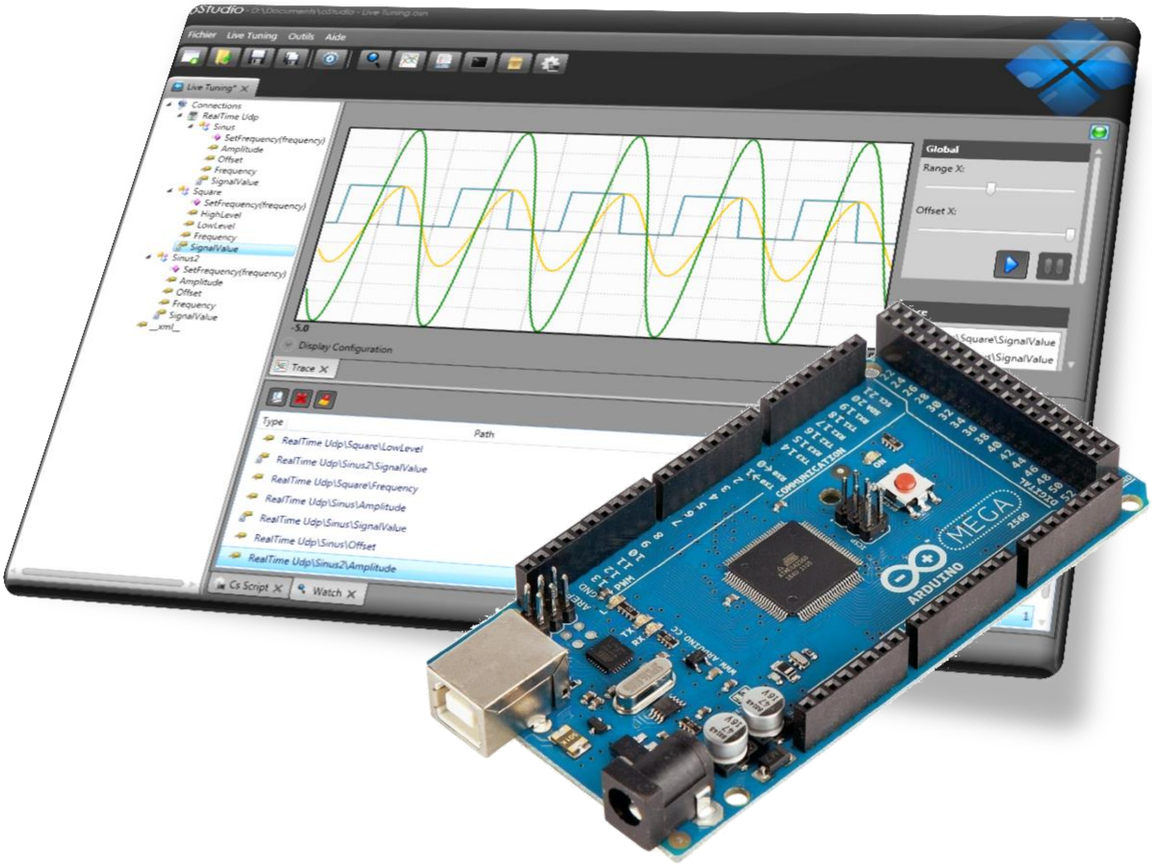
www.objectis-software.com

Live Tuning for Arduino with OcfEmbedded in 15 minutes

Application note



Rev. 1.0

Date : January 2014

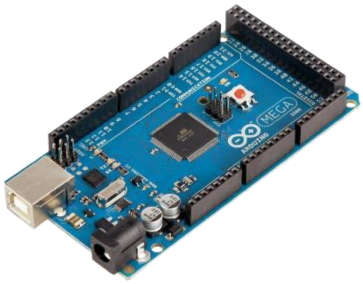
Table of contents

1 Introduction

Live tuning on embedded targets allows live interaction with embedded applications while they are running. Live debugging allows developers to watch variables states, to trace or set them. Better yet, Live Tuning allows to invoke functions. These functionalities dramatically improve the development and setup phase of any embedded application.

Live Tuning on embedded target is composed of two parts: the embedded code and the oStudio tool. The embedded part of Live Tuning is very simple code to port on the embedded target, mainly by porting the physical layer of the protocol. This application notes provide all the required elements to test OcfEmbedded on Arduino.

Please read the OcfEmbedded programmer guide for more detailed information.



Arduino board

oStudio Live Tuning as live debugger

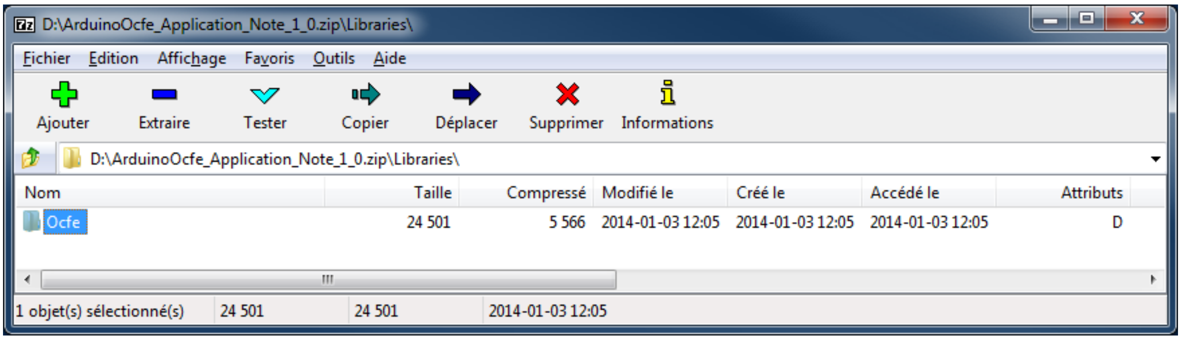
2 Step by step procedure

2.1 Download and setup

1. Download and install oStudioLive Tuning from Objectis web site www.objectis-software.com
2. Download and open the ArduinoOcfe_Application_Note.zip

2.2 Copy Ocfe files to Arduino IDE libraries

From the ArduinoOcfe_Application_Note.zip file, folder "libraries" :



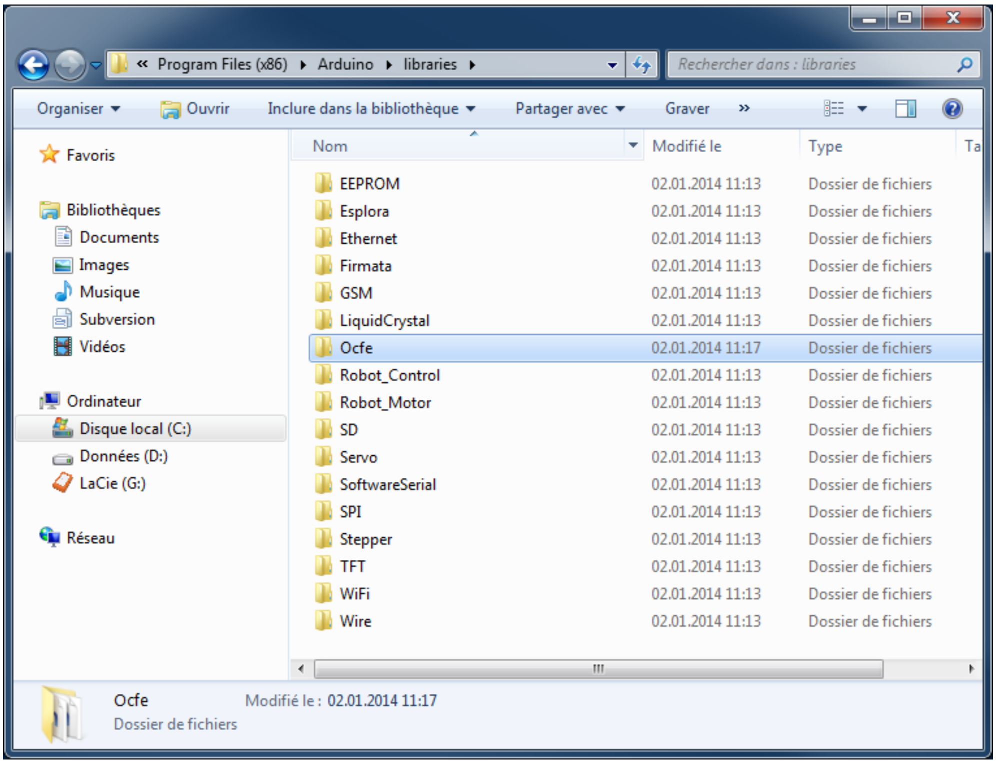
D:\ArduinoOcfe_Application_Note_1_0.zip\Libraries\

D:\ArduinoOcfe_Application_Note_1_0.zip\Libraries\

Nom	Taille	Compressé	Modifié le	Créé le	Accédé le	Attributs
Ocfe	24 501	5 566	2014-01-03 12:05	2014-01-03 12:05	2014-01-03 12:05	D

1 objet(s) sélectionné(s) 24 501 24 501 2014-01-03 12:05

extract the folder “Ocfe” to the “Arduino IDE” libraries folder (which is typically C:\Program Files (x86)\Arduino\libraries) :

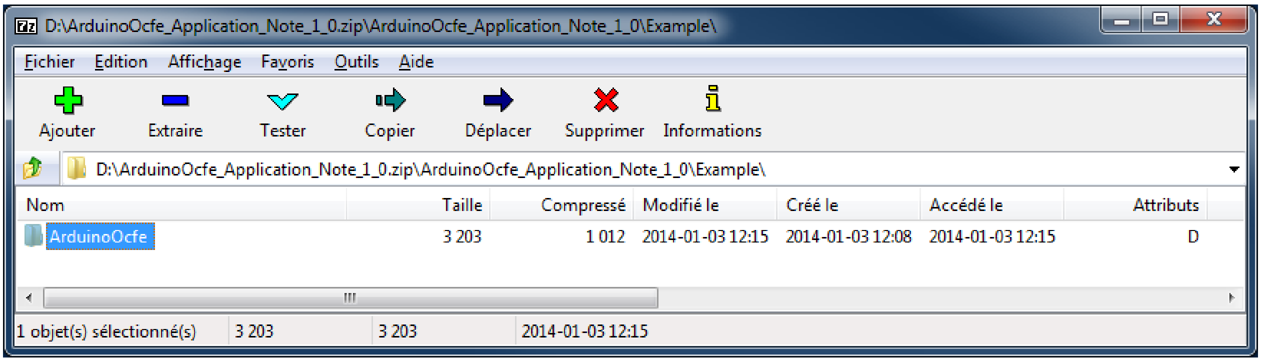


Nom	Modifié le	Type
EEPROM	02.01.2014 11:13	Dossier de fichiers
Esplora	02.01.2014 11:13	Dossier de fichiers
Ethernet	02.01.2014 11:13	Dossier de fichiers
Firmata	02.01.2014 11:13	Dossier de fichiers
GSM	02.01.2014 11:13	Dossier de fichiers
LiquidCrystal	02.01.2014 11:13	Dossier de fichiers
Ocfe	02.01.2014 11:17	Dossier de fichiers
Robot_Control	02.01.2014 11:13	Dossier de fichiers
Robot_Motor	02.01.2014 11:13	Dossier de fichiers
SD	02.01.2014 11:13	Dossier de fichiers
Servo	02.01.2014 11:13	Dossier de fichiers
SoftwareSerial	02.01.2014 11:13	Dossier de fichiers
SPI	02.01.2014 11:13	Dossier de fichiers
Stepper	02.01.2014 11:13	Dossier de fichiers
TFT	02.01.2014 11:13	Dossier de fichiers
WiFi	02.01.2014 11:13	Dossier de fichiers
Wire	02.01.2014 11:13	Dossier de fichiers

Ocfe Modifié le : 02.01.2014 11:17
Dossier de fichiers

2.3 Extract the example

From the ArduinoOcfe_Application_Note.zip file, folder “Example”:



D:\ArduinoOcfe_Application_Note_1_0.zip\ArduinoOcfe_Application_Note_1_0\Example\

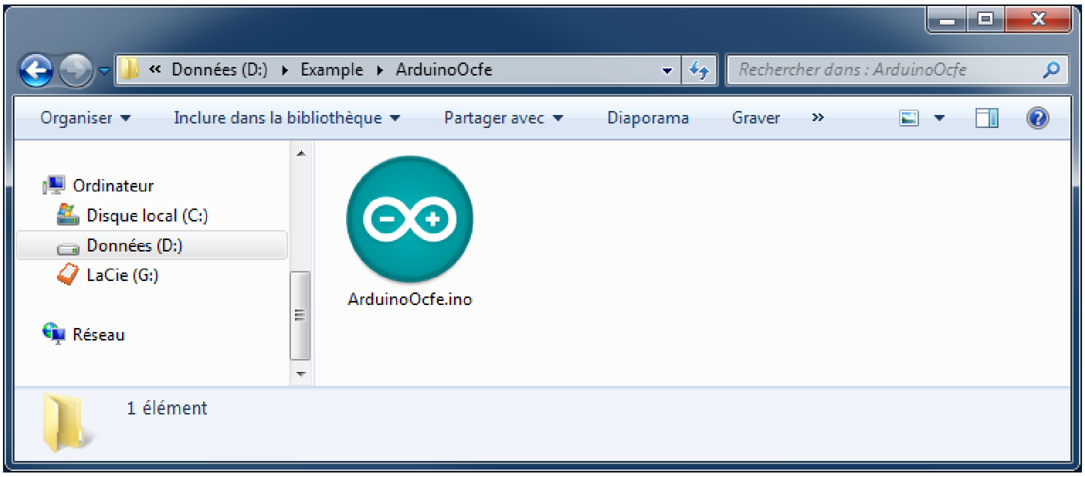
Nom	Taille	Compressé	Modifié le	Créé le	Accédé le	Attributs
ArduinoOcfe	3 203	1 012	2014-01-03 12:15	2014-01-03 12:08	2014-01-03 12:15	D

1 objet(s) sélectionné(s) 3 203 3 203 2014-01-03 12:15

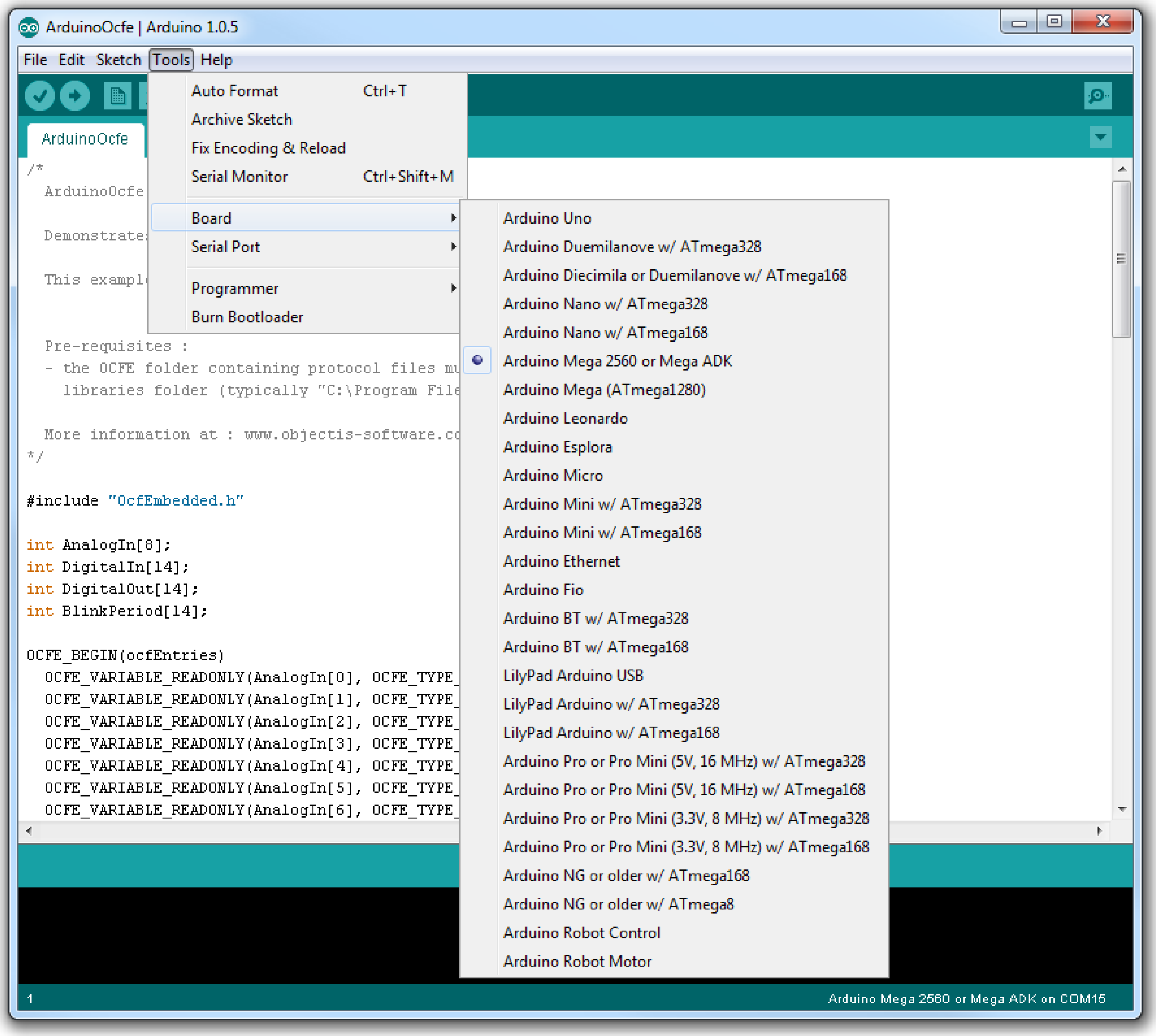
Extract the ArduinoOcfe folder to the folder where you usually store your Arduino projects.

2.4 Compile and upload the example to your Arduino target

Open the ArduinoOcfe.ino file with “Arduino Ide”



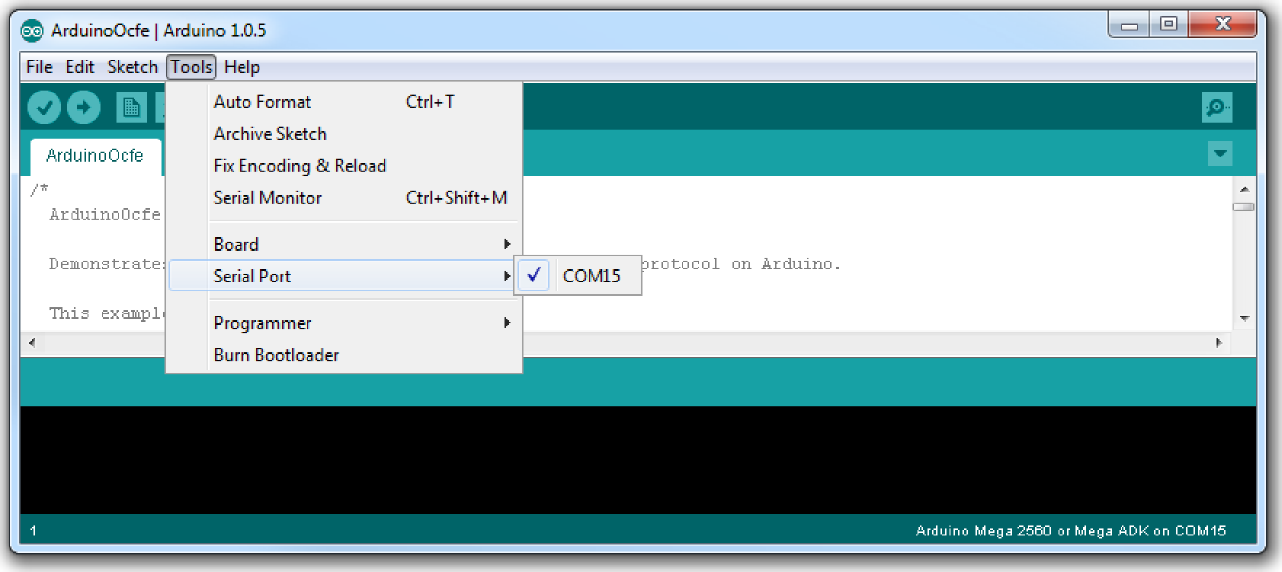
Select the Board model matching your Arduino


ArduinoOcfe | Arduino 1.0.5
File
Edit
Sketch
Tools
Help
ArduinoOcfe
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
● Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor

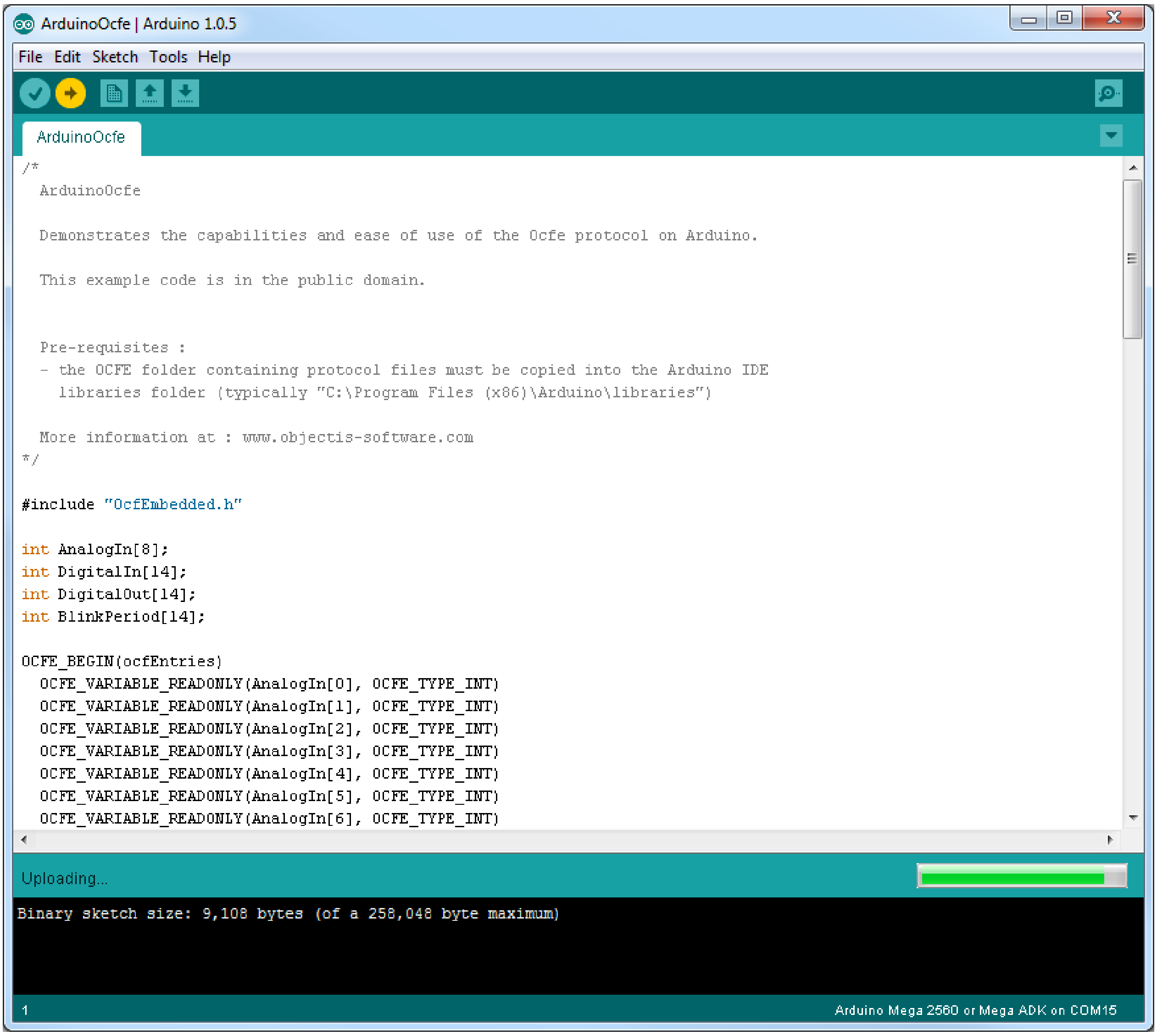
```
/*
  ArduinoOcfe

  Demonstrate

  This exampl


  Pre-requisites :
  - the OCFE folder containing protocol files mu
    libraries folder (typically "C:\Program File

  More information at : www.objectis-software.co
*/

#include "OcfEmbedded.h"

int AnalogIn[8];
int DigitalIn[14];
int DigitalOut[14];
int BlinkPeriod[14];

OCFE_BEGIN(ocfEntries)
  OCFE_VARIABLE_READONLY(AnalogIn[0], OCFE_TYPE_
  OCFE_VARIABLE_READONLY(AnalogIn[1], OCFE_TYPE_
  OCFE_VARIABLE_READONLY(AnalogIn[2], OCFE_TYPE_
  OCFE_VARIABLE_READONLY(AnalogIn[3], OCFE_TYPE_
  OCFE_VARIABLE_READONLY(AnalogIn[4], OCFE_TYPE_
  OCFE_VARIABLE_READONLY(AnalogIn[5], OCFE_TYPE_
  OCFE_VARIABLE_READONLY(AnalogIn[6], OCFE_TYPE_
```

1 Arduino Mega 2560 or Mega ADK on COM15

Connect your Arduino to USB port and select the corresponding communication port:



Compile and upload to the Arduino



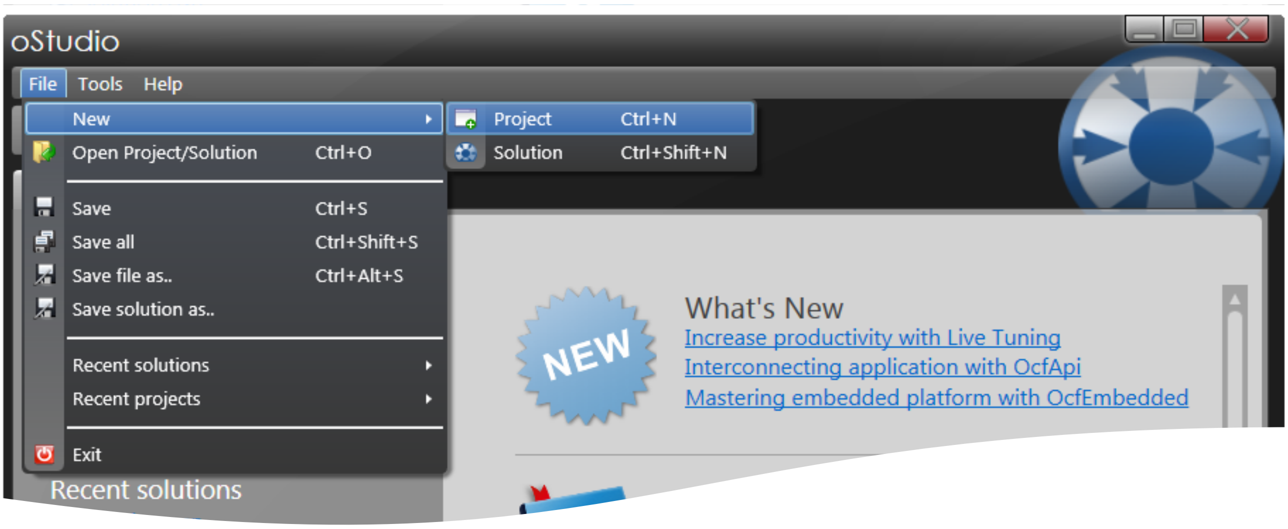
```
/*
  ArduinoOcfe

  Demonstrates the capabilities and ease of use of the Ocfe protocol on Arduino.

  This example code is in the public domain.


  Pre-requisites :
  - the OCFE folder containing protocol files must be copied into the Arduino IDE
    libraries folder (typically "C:\Program Files (x86)\Arduino\libraries")

  More information at : www.objectis-software.com
*/

#include "OcfEmbedded.h"

int AnalogIn[8];
int DigitalIn[14];
int DigitalOut[14];
int BlinkPeriod[14];

OCFE_BEGIN(ocfEntries)
  OCFE_VARIABLE_READONLY(AnalogIn[0], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[1], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[2], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[3], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[4], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[5], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[6], OCFE_TYPE_INT)
```

Uploading...

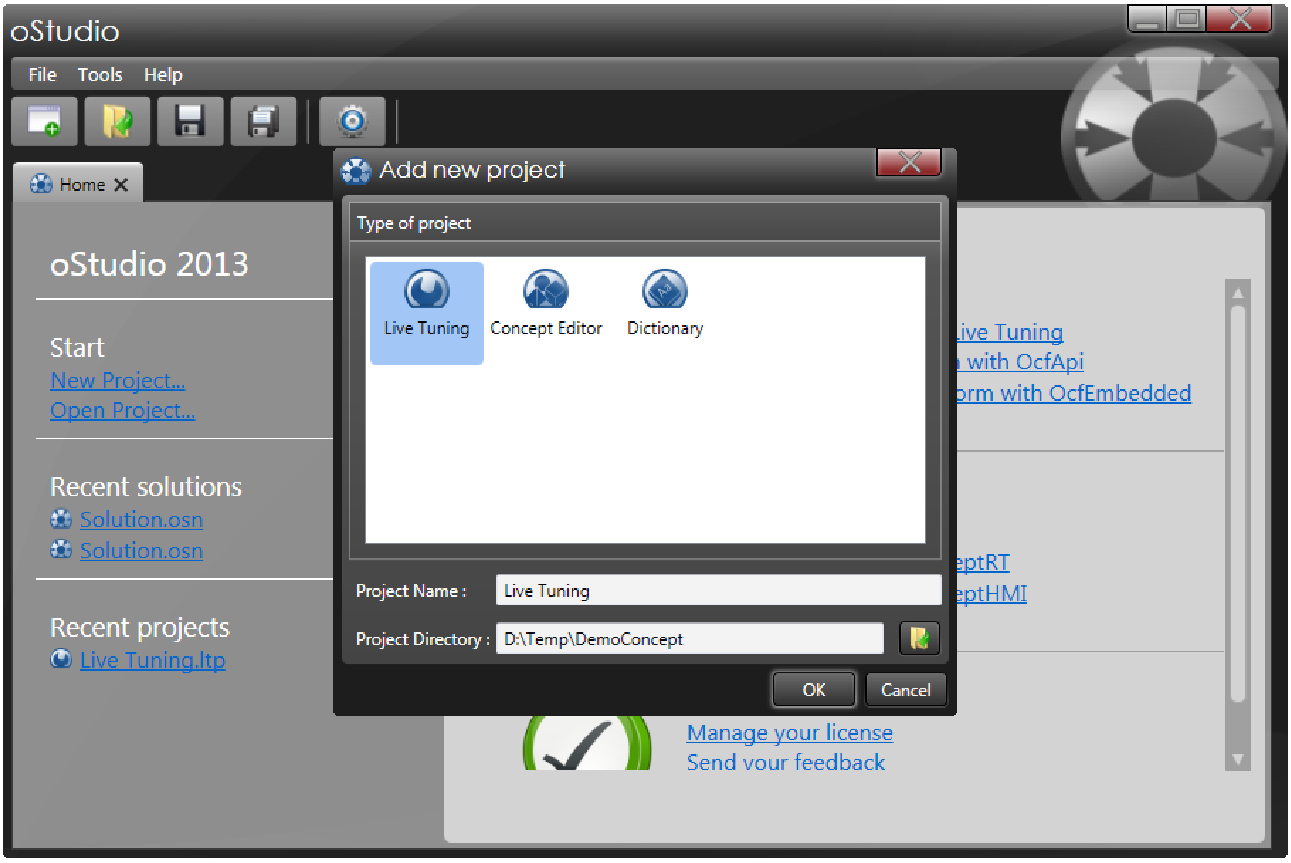
```
Binary sketch size: 9,108 bytes (of a 258,048 byte maximum)
```

2.5 Test and enjoy with oStudio LiveTuning

1. Once the example is uploaded, the Output 13 Led should blink with a 1 second period.
2. From the Windows start menu, open “oStudio”
3. Select menu File/New/Project


oStudio
File
Tools
Help
New
Open Project/Solution Ctrl+O
Save Ctrl+S
Save all Ctrl+Shift+S
Save file as.. Ctrl+Alt+S
Save solution as..
Recent solutions
Recent projects
Exit
Project Ctrl+N
Solution Ctrl+Shift+N
NEW
What's New
Increase productivity with Live Tuning
Interconnecting application with OcfApi
Mastering embedded platform with OcfEmbedded
Recent solutions

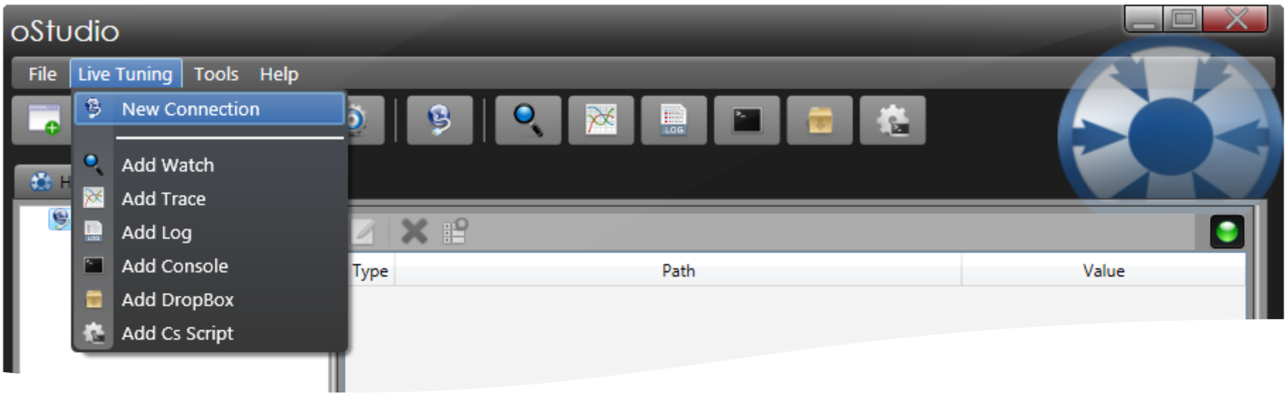
4. In the “Add new project” dialog, select the “Live Tuning” project type, fill in the project name and select the desired folder, then click ok.



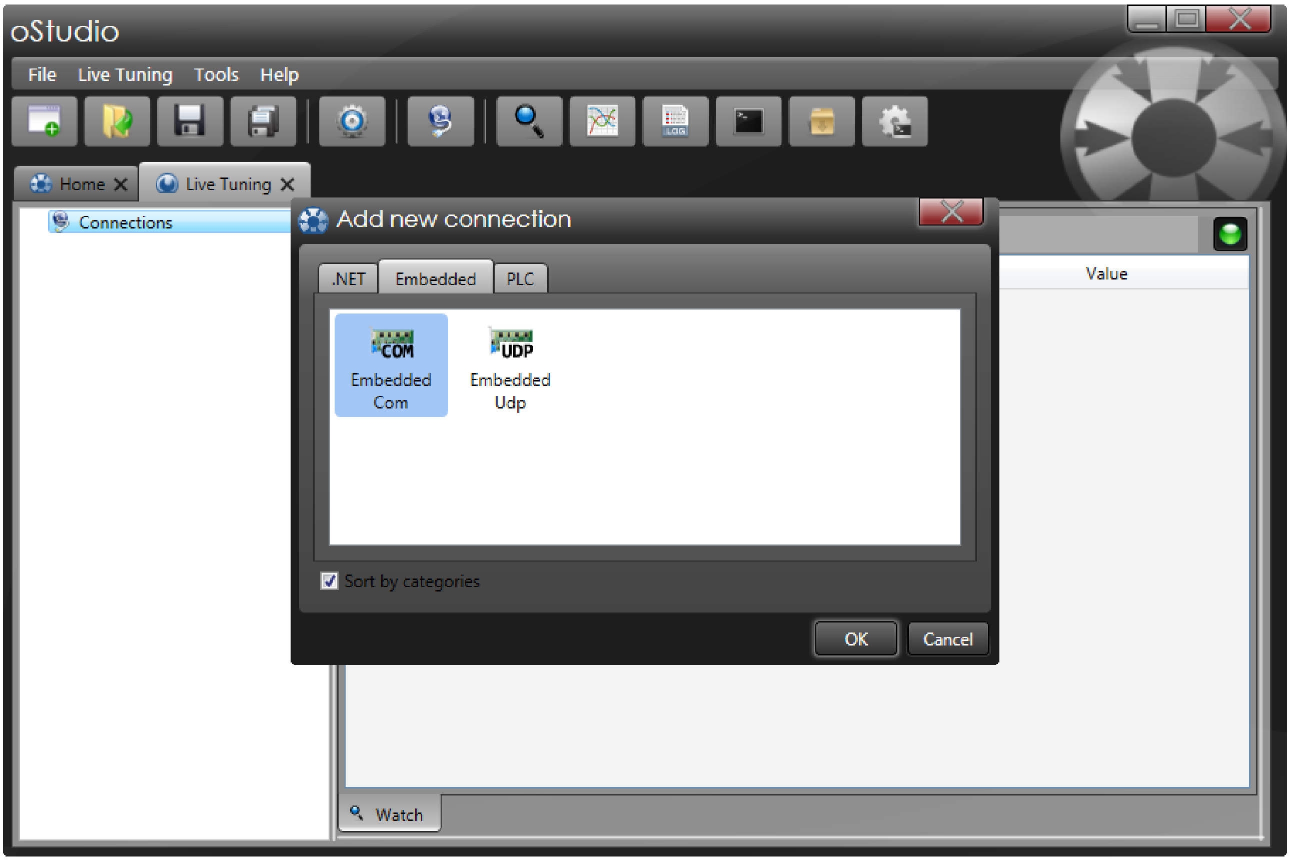
Project Name : Live Tuning

Project Directory : D:\Temp\DemoConcept

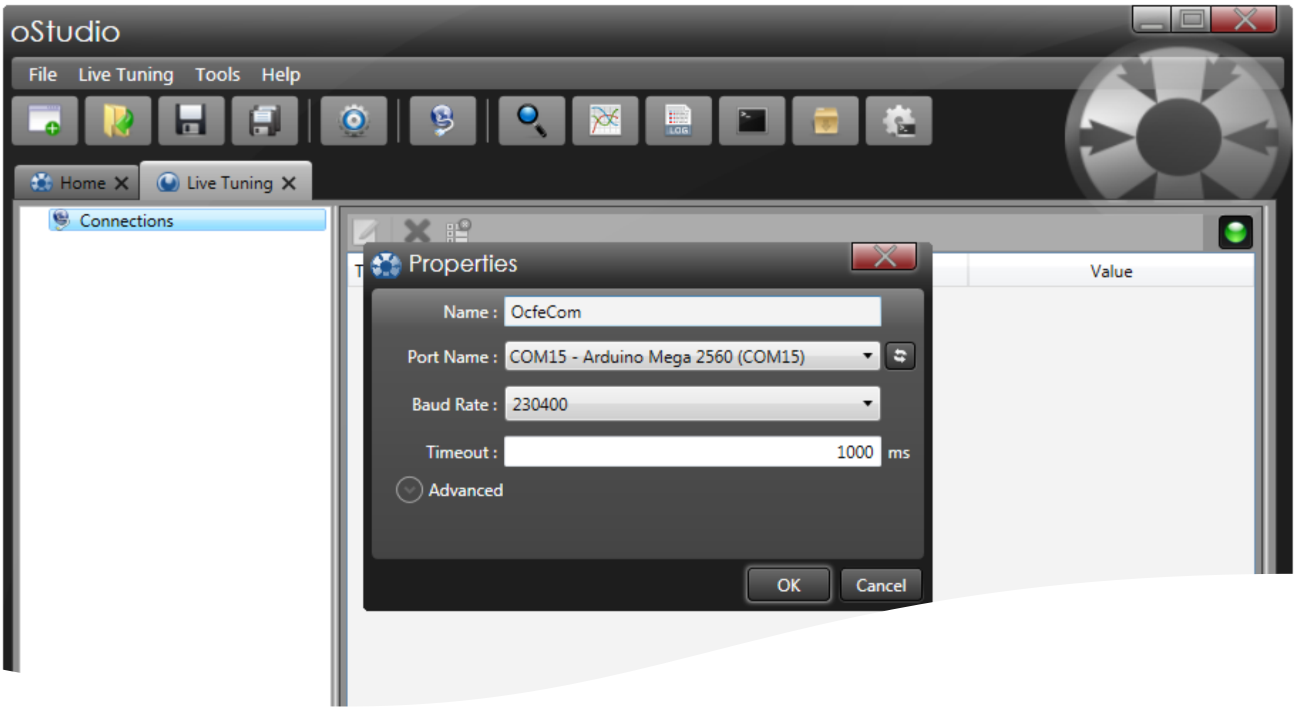
5. From the menu “Live Tuning”, select “New connection” :



6. In the “Add New connection” dialog, select “Embedded Com”, this is the communication protocol for Ocf Embedded through a serial “Com” port.



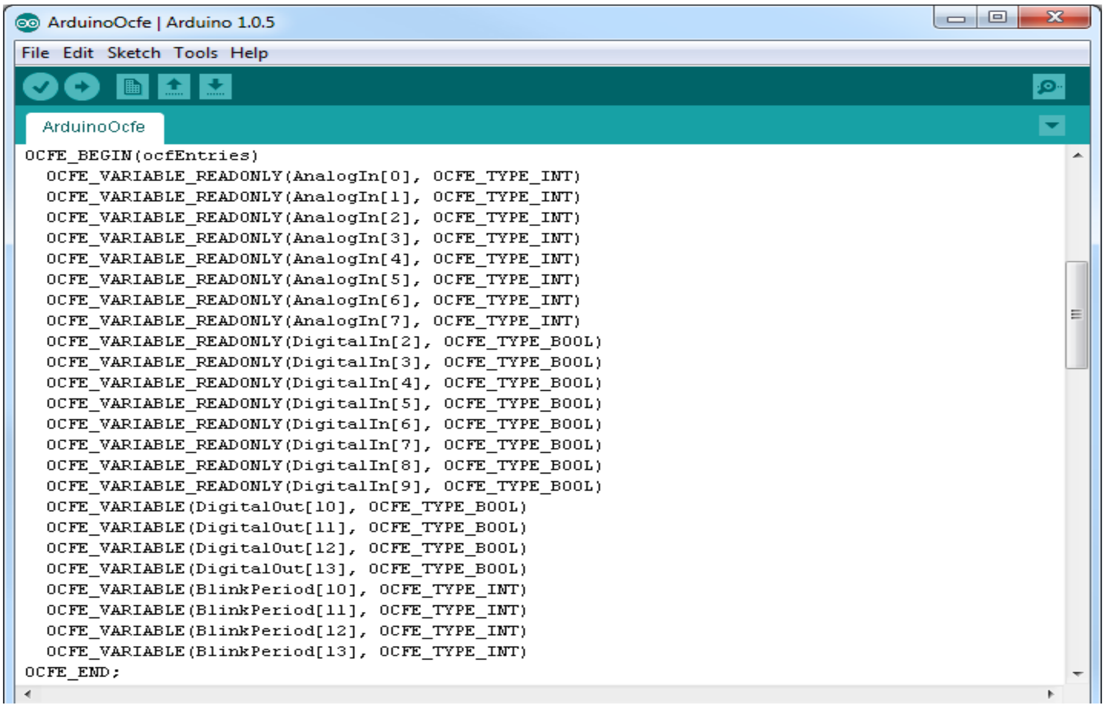
7. In the “Properties” dialog, fill in the appropriate communication parameters :
   - Select the “Com” port corresponding to your Arduino.
   - Select the baud rate “230400” to match the settings in the example program.
   - Click ok.



Name : OcfeCom
Port Name : COM15 - Arduino Mega 2560 (COM15)
Baud Rate : 230400
Timeout : 1000 ms

8. Browse the published variables

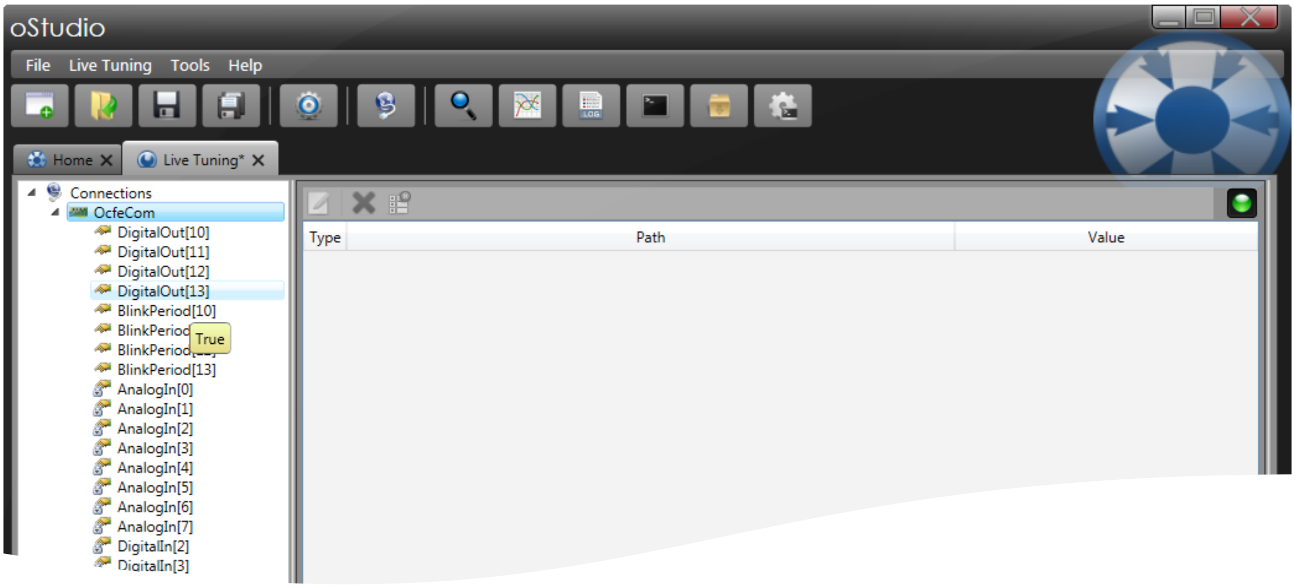
   All the variables published by the example program through the OCFE protocol are exported in the following section of the source code:



```
OCFE_BEGIN(ocfEntries)
  OCFE_VARIABLE_READONLY(AnalogIn[0], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[1], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[2], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[3], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[4], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[5], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[6], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(AnalogIn[7], OCFE_TYPE_INT)
  OCFE_VARIABLE_READONLY(DigitalIn[2], OCFE_TYPE_BOOL)
  OCFE_VARIABLE_READONLY(DigitalIn[3], OCFE_TYPE_BOOL)
  OCFE_VARIABLE_READONLY(DigitalIn[4], OCFE_TYPE_BOOL)
  OCFE_VARIABLE_READONLY(DigitalIn[5], OCFE_TYPE_BOOL)
  OCFE_VARIABLE_READONLY(DigitalIn[6], OCFE_TYPE_BOOL)
  OCFE_VARIABLE_READONLY(DigitalIn[7], OCFE_TYPE_BOOL)
  OCFE_VARIABLE_READONLY(DigitalIn[8], OCFE_TYPE_BOOL)
  OCFE_VARIABLE_READONLY(DigitalIn[9], OCFE_TYPE_BOOL)
  OCFE_VARIABLE(DigitalOut[10], OCFE_TYPE_BOOL)
  OCFE_VARIABLE(DigitalOut[11], OCFE_TYPE_BOOL)
  OCFE_VARIABLE(DigitalOut[12], OCFE_TYPE_BOOL)
  OCFE_VARIABLE(DigitalOut[13], OCFE_TYPE_BOOL)
  OCFE_VARIABLE(BlinkPeriod[10], OCFE_TYPE_INT)
  OCFE_VARIABLE(BlinkPeriod[11], OCFE_TYPE_INT)
  OCFE_VARIABLE(BlinkPeriod[12], OCFE_TYPE_INT)
  OCFE_VARIABLE(BlinkPeriod[13], OCFE_TYPE_INT)
OCFE_END;
```

The list of published variables can easily be changed to match your own application. See the Ocfe programmer guide for more information.

All those metadata are read by oStudio when connecting to the Arduino target (or when F5 key is pressed to refresh the tree). So, once connected to the Arduino, oStudio let you immediately browse all those variables with the left tree view.

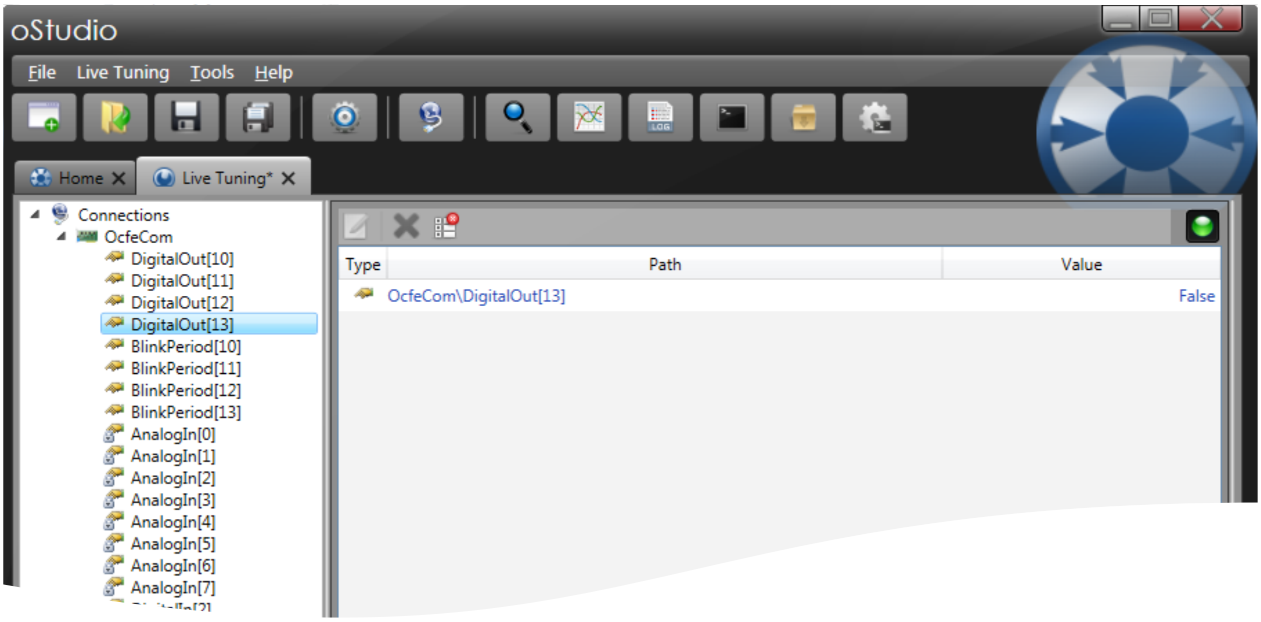


While the mouse cursor is moving around a variable name, a tooltip appears displaying the current value of the variable, retrieved through OCFE protocol.

9. Monitor the variables

Simply drag drop a variable from the tree on the left to the watch pane on the right.

The variable is monitored, and the value is continuously updated.

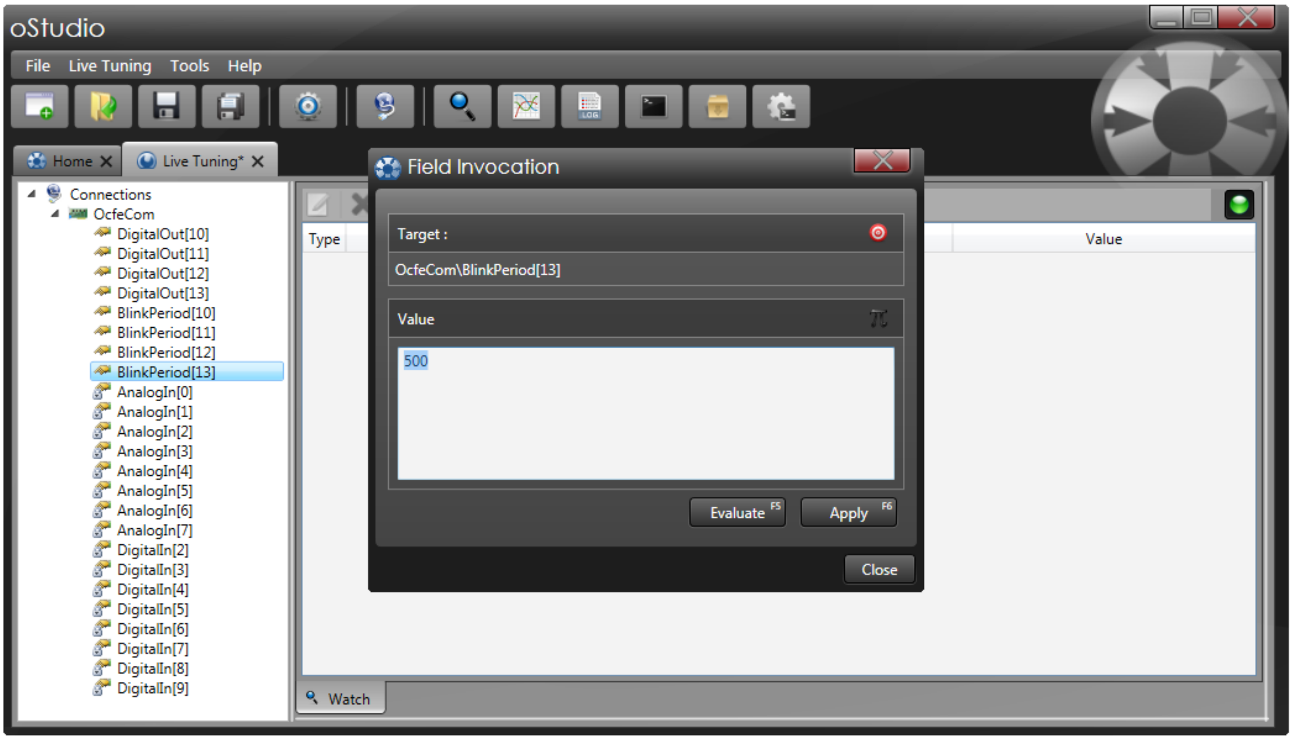


Type	Path	Value
	OcfeCom\DigitalOut[13]	False

Doing so with the DigitalOutput[13] that is blinking, you will see the toggling of the variable in the watch pane.

10. Remotely change the value of a variable

    A double click on any variable will open the “Field invocation” dialog. This dialog will let you type a new value for the variable that will be applied with the “Apply” button.



Target : OcfeCom\BlinkPeriod[13]

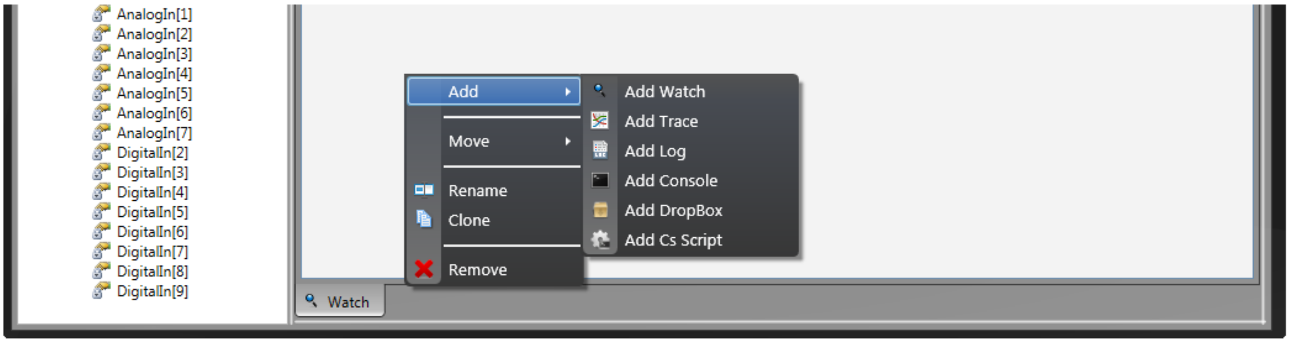
Value 500

To test this functionality, you can change the blink period of the digital output 13 by setting a new value to BlinkPeriod[13] (for example, set the value 500 to set the blink period to 500 ms). In this Arduino ocfe example application, setting the blink period to 0 will stop blinking.

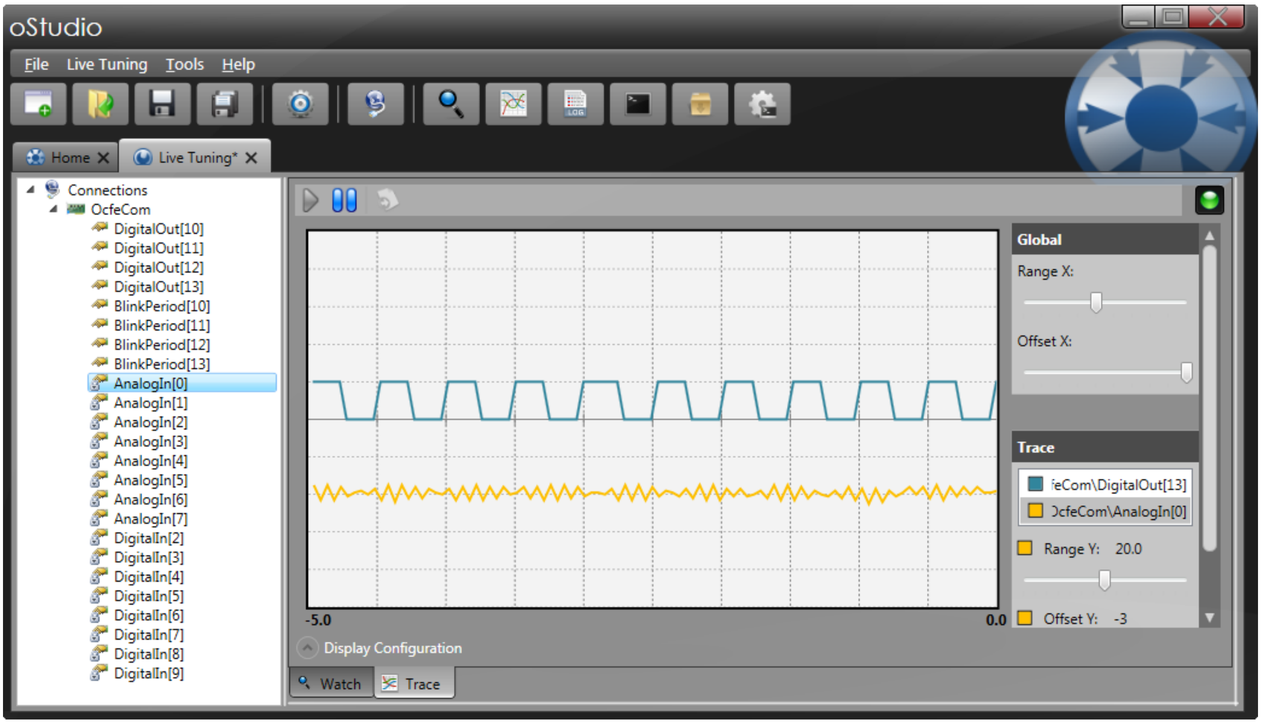
In the same way, you can write the value of a DigitalOutput, by writing the corresponding variable.

11. Tracing a variable

    A right click in the tabs will let you create new panes.

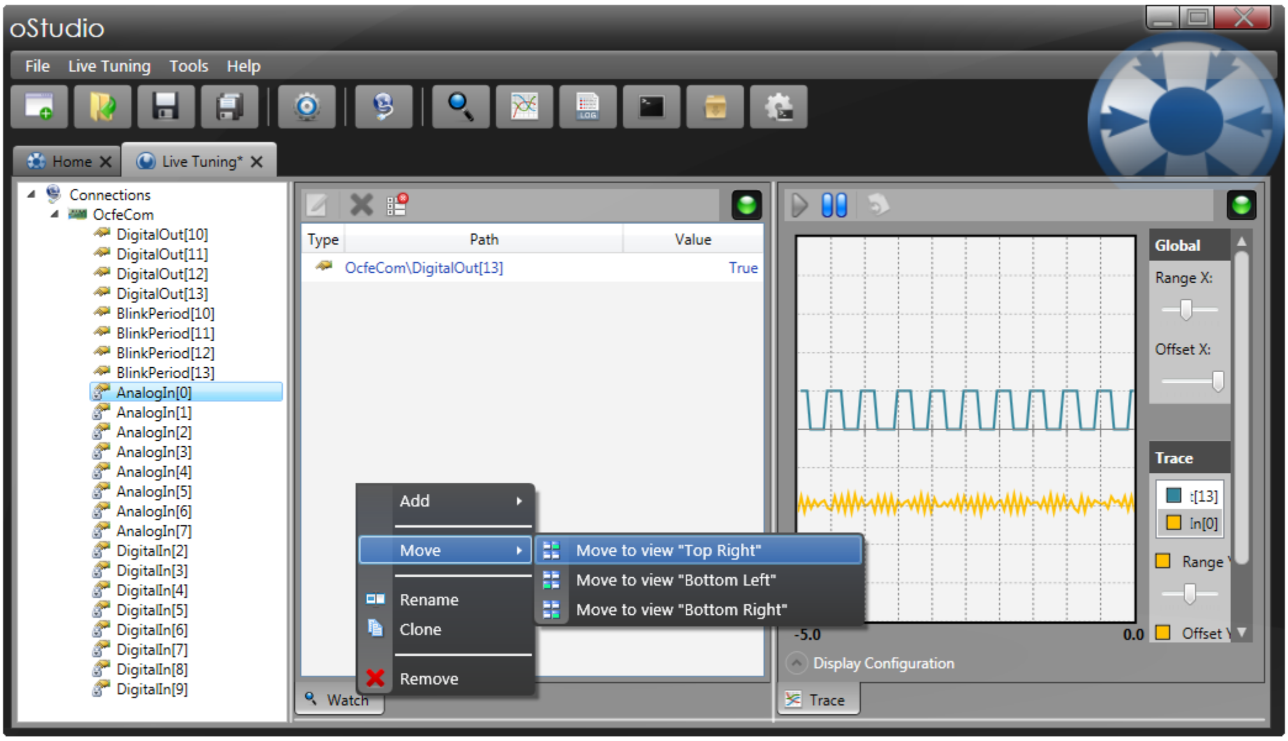


Create a new trace pane, and simply drag-drop the variables to trace into this new pane.



In the picture above, the variables DigitalOut[13] and AnalogIn[0] are plot.

oStudio let you organize your panes in multiple views :



Type	Path	Value
	OcfeCom\DigitalOut[13]	True

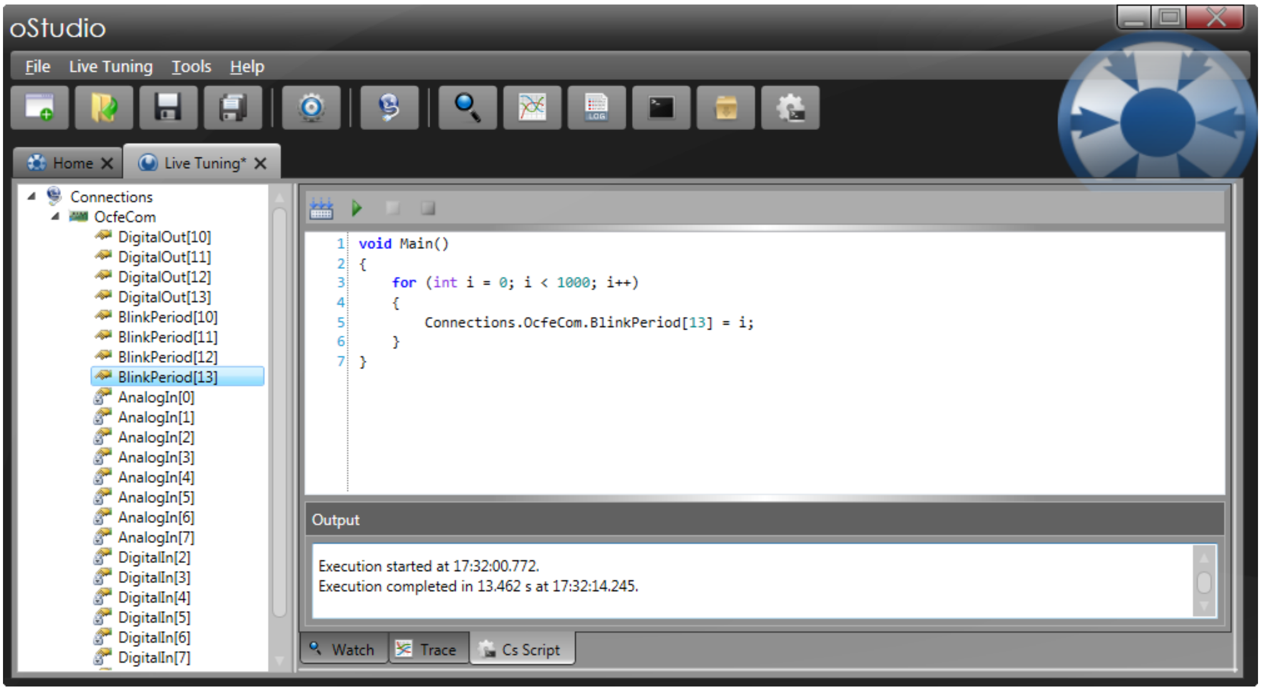
12. Scripting

oStudio offers many additional functionalities, that are extremely useful to tune an application. We will finish by writing a small C# script demonstrating how to write a variable.

Simply create a new pane “CsScript”, which let you write some C# code that will be compiled on the fly.

To read or write a variable into the Arduino using Ocfe, simply use the path of the variable and read it or write it as usual with the "=" operator. To make this more easy, you can also simply drag drop a variable from the tree to the script pane.

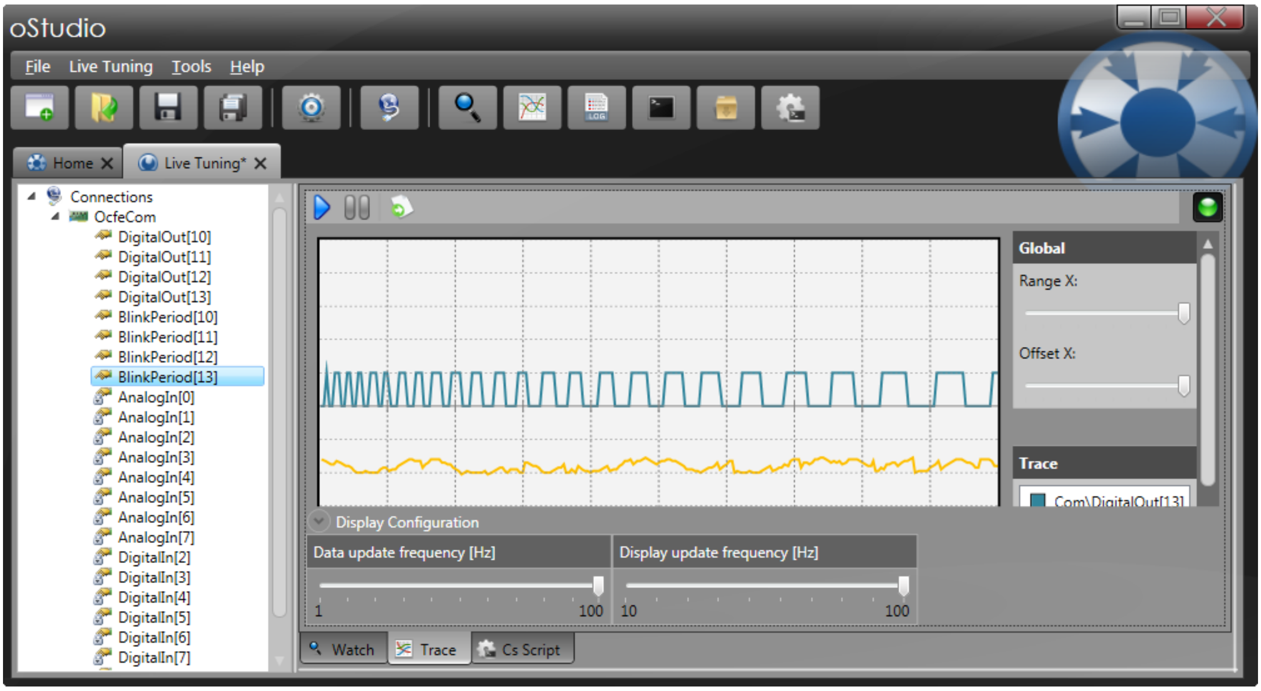
As an example, we will write a script that slowly changes the blinking period from 0 to 500 ms for output 13.



```
void Main()
{
    for (int i = 0; i < 1000; i++)
    {
        Connections.OcfeCom.BlinkPeriod[13] = i;
    }
}
```

```
Execution started at 17:32:00.772.
Execution completed in 13.462 s at 17:32:14.245.
```

Click on the run button. The blink period will slowly increase from very fast to normal blinking. The result is cleary visible on the Trace below :



Keep in mind that Ocfe is dynamic typed, so it is sometimes better or even required with C# scripts to use type cast to ensure proper execution.

The code written as scripts in oStudio can also be integrated into your own plain .Net applications.

To do so, you simply need to download the OcfApi, an extremely easy to use library.

3 More to do with oStudio and LiveTuning

Our experience is that oStudio, the live debugging tool, dramatically improves the development efficiency and the tuning of embedded systems.

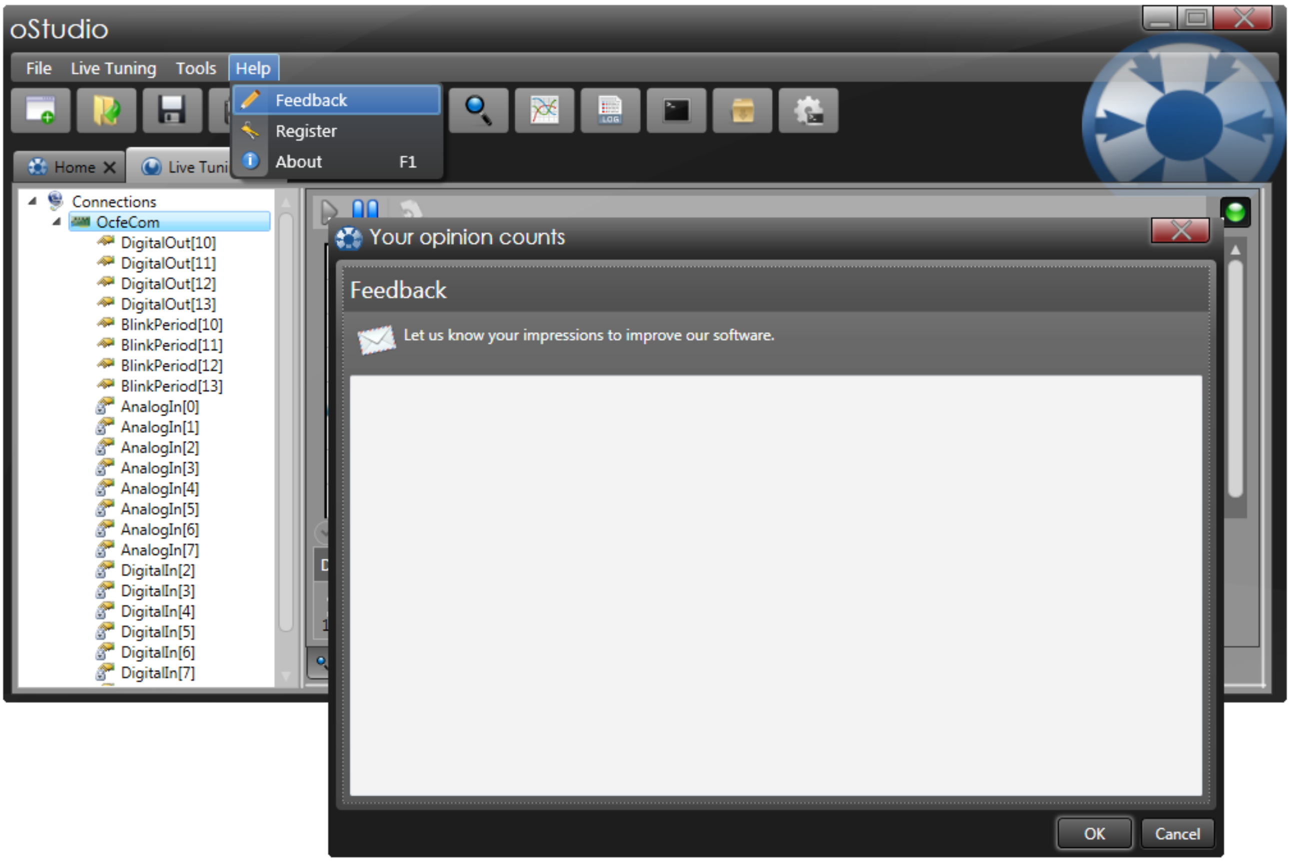
We wish that you will also get a lot of benefits from oStudio in your own projects.

You can go further with Ocf solutions :

- OcfNative is a communication protocol that let you monitor in the same easy way any of your plain .Net applications. Simply publish your objects with a few lines of code and oStudio let you grab into your application.
- “OcfNative AES encrypted“ let you achieve the same results with a high level of security on open networks.
- OcfApi let you very easily integrate Ocf protocols into your own .Net applications in a snap. That way, Ocf is not only a debugging facility, but a first class communication solution.

Ocf is a simple and efficient object oriented communication solution that simplifies to the extreme the interoperability of applications.

OBJECTIS is continuously working on improving oStudio and Ocf to deliver excellent solutions to the community. We will appreciate your feedbacks.



Your opinion counts

Feedback

Let us know your impressions to improve our software.