

ARDUINO IO SIMULATOR MACOS 1.8

USER MANUAL

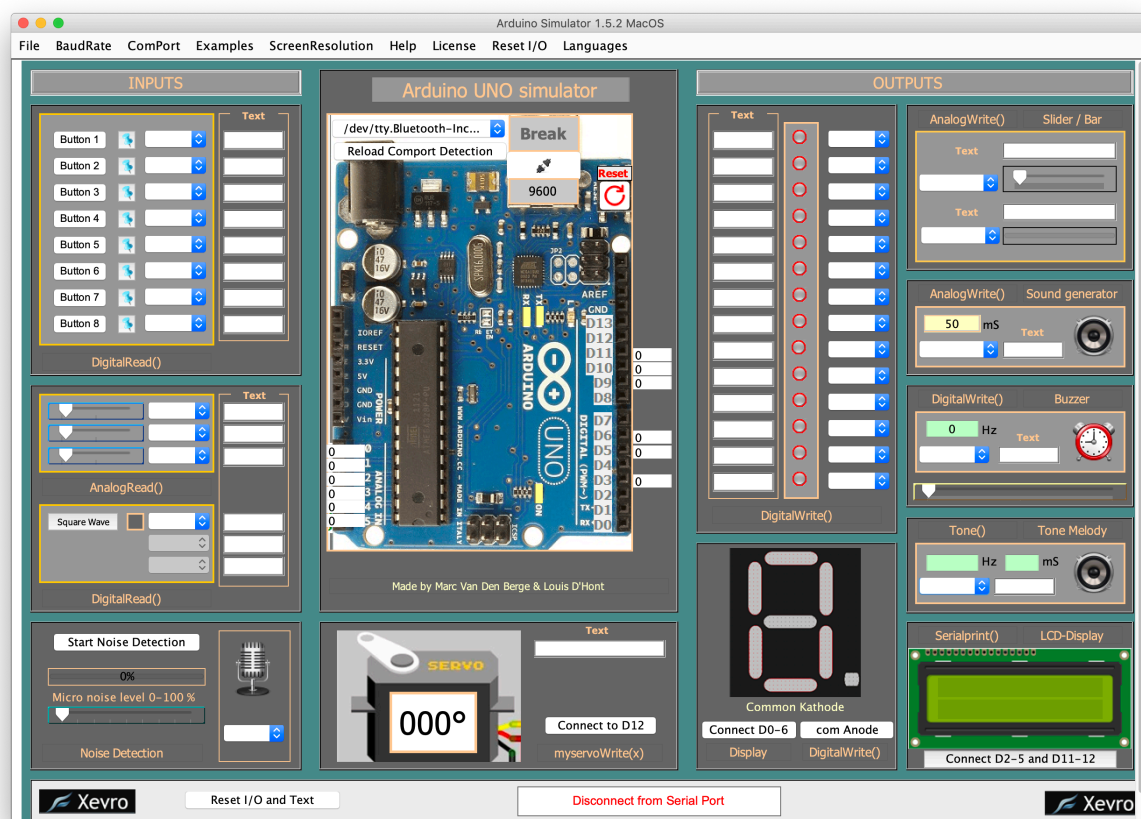


XEVRO

Version 1.8

Xevro© 2021

This manual describes all the features and capabilities of the Arduino Simulator.



INTRODUCTION

The Arduino Simulator gives you the tools and components you need to simulate your Arduino IO. It's made for quick tests and small projects and there is still further developed in order to obtain the widest possible IO functions.

This Arduino IO Simulator is designed to test an Arduino program quickly with the Arduino board without really having connections to external IO (buttons, potentiometers, LEDs, LCD displays, ...) and add a nice custom drawing around it to get a better simulation experience.

To use the simulator you need 3 programs:

- - Java JDK
- - The Arduino Simulator
- - The Arduino IDE software

In order to use the Simulator we need to download the Java JDK on our MacBook or iMac, you can find the download link on the website of Xevro. Or go to this link and download the macOS version:

www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.92 MB	jdk-8u161-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u161-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.96 MB	jdk-8u161-linux-i586.rpm
Linux x86	183.76 MB	jdk-8u161-linux-i586.tar.gz
Linux x64	166.09 MB	jdk-8u161-linux-x64.rpm
Linux x64	180.97 MB	jdk-8u161-linux-x64.tar.gz
macOS	247.12 MB	jdk-8u161-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.99 MB	jdk-8u161-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.29 MB	jdk-8u161-solaris-sparcv9.tar.gz
Solaris x64	140.57 MB	jdk-8u161-solaris-x64.tar.Z
Solaris x64	97.02 MB	jdk-8u161-solaris-x64.tar.gz
Windows x86	198.54 MB	jdk-8u161-windows-i586.exe
Windows x64	206.51 MB	jdk-8u161-windows-x64.exe

ARDUINO IDE

For we start using the Arduino Simulator we need the Arduino software, it is also freely available on the Arduino website: <http://arduino.cc/en/Main/Software>

Download the Arduino Software

ARDUINO 1.6.8

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions.

Windows Installer
[windows.zip](#) (for non-admin install)

Mac OS X for Java 6 (recommended)
Mac OS X for Java 7+ (experimental)

Linux 32 bits
Linux 64 bits

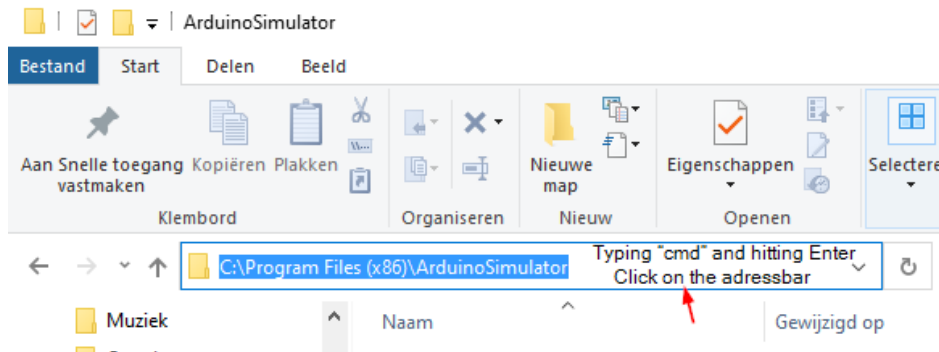
[Release Notes](#) [Source Code](#)

HOW TO START THE SIMULATOR FROM A COMMAND PROMPT

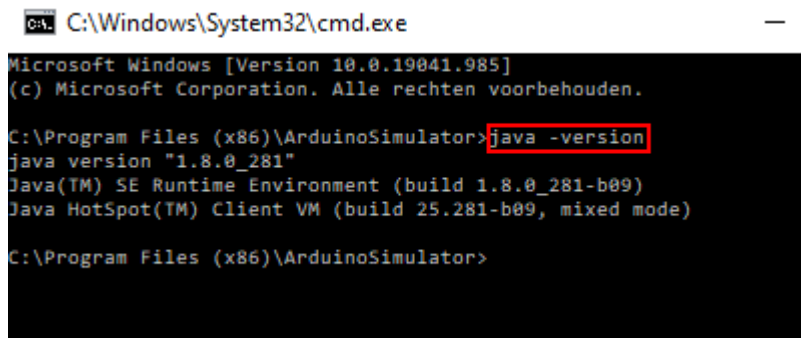
You can open a Command Prompt window directly from inside a Windows Explorer window. Taking you directly to that folder location!

If you click on this address bar, you can type in text. By typing 'cmd' and hitting Enter, you'll open up the command prompt at that location.

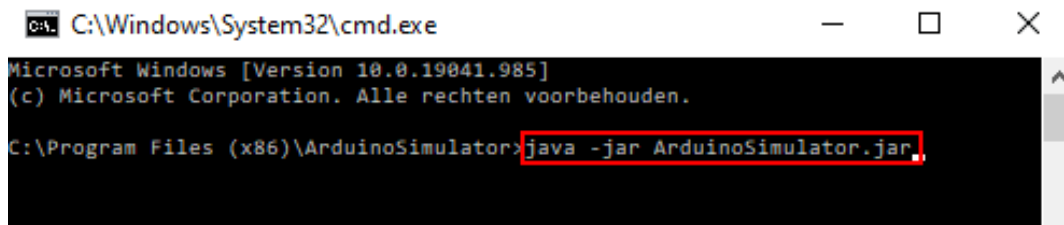
Go to the directory C:\Program Files (x86)\ArduinoUNOSimulator



Check java version commando: java -version + ENTER



Start simulator commando: java -jar ArduinoSimulator.jar + ENTER



If java doesn't work in command prompt

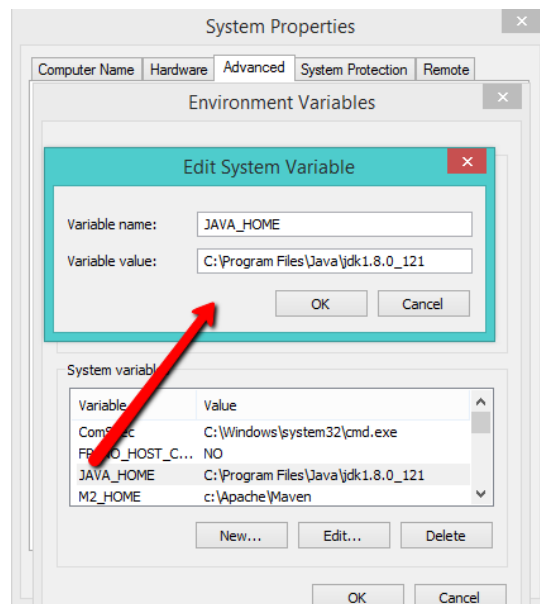
Press simultaneously the "windows" and "pause" buttons on your keyboard, this will bring up the System Preferences dialog. In the Advanced tab, find Environment Variables.

In the User (upper) section, create or update the following two variables :

- JAVA_HOME = where you put your JDK, eg. C:/Java/SDK
- PATH = %JAVA_HOME%/bin
-

Close the dialogs.

Then, in a new command-line console, try "java -version" and see if it's detected. It's important that you use a new console, because environment variables are read only when the console is launched.



INSTALLATION GUIDE

1. Before you can use the Arduino Simulator you will need to install the Java JDK. On Windows computers, it's enough to install the basic Java JRE.

The Java JDK isn't automatically installed and the simulator will not run without the Java JDK. www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

2. Xevro is an unauthorized company so your Mac will give a message to give us permissions. In order to do that you will go to 'System Preferences' and go to the 'Security & Privacy'. Click on the tab 'General' and press on the 'Open Anyway' button, and the Arduino Simulator will start.



3. Drag the Arduino Simulator 1.8 MacOS in the Programs folder.

4. Copy the license key in the Arduino Simulator of the website page.

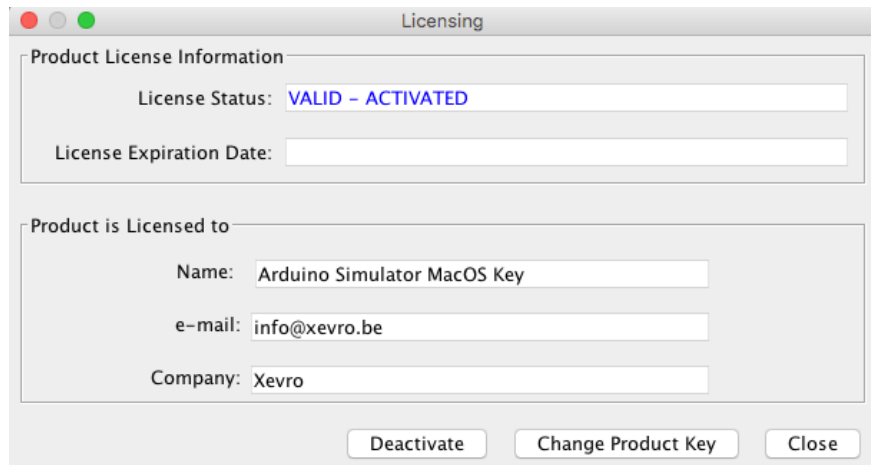
If you bump into a mistake or problem, please send me an e-mail to info@xevro.be

More information on the Xevro support page: <https://xevro.be/support/simulator-manual-support.html#SimulatorIO>

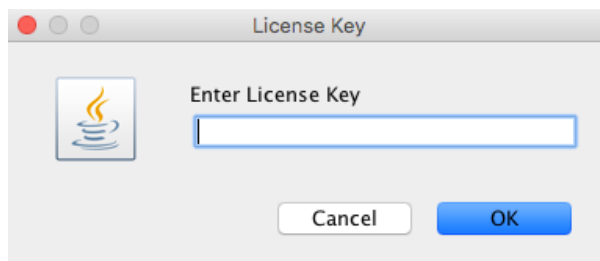
LICENSE ACTIVATION

The Arduino Simulator free available but we secured it with a license key. The first time you opens the program there will be an activation screen pops up where you can put in the license and activate it.

Click on the 'Change Product Key' to insert the license key you copied on the website, after entering this you need to click on 'activate'.



License key input field



CODE CHANGES

The Arduino IDE works with instructions that the IO read and write, by adding the libraries to your project it is possible to simulate the project.

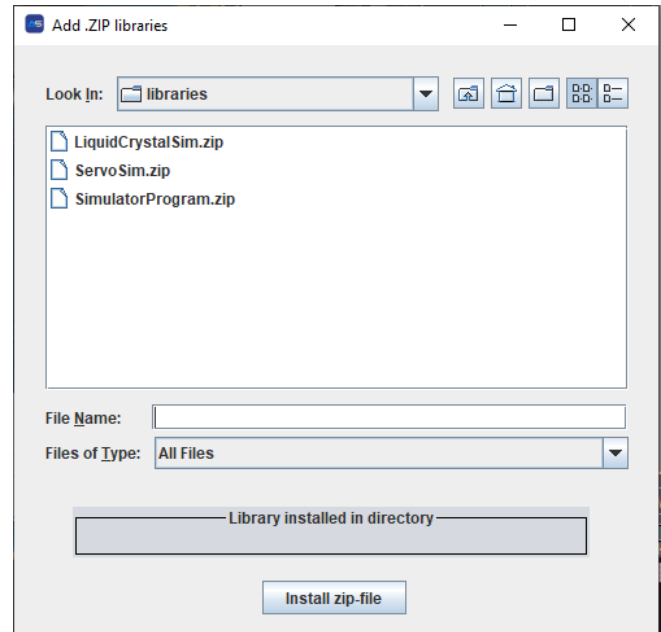
We don't want to change the real instructions, we decided to edit the libraries so they are compatible with our software. The core SimulatorProgram library will be added to the Arduino/libraries folder in the documents folder the very first time you open the simulator. In the Simulator software is a function available to add a library to add more libraries to the Arduino IDE.

All available Simulator libraries

.:) > Program Files (x86) > ArduinoUNC

Naam

ⓧ LiquidCrystalSim
ⓧ ServoSim
ⓧ SimulatorProgram



Go inside a folder and select the .zip file, click on the 'install zip-file' button to install the library. The library will be automatically recognized by the Arduino IDE. by adding the Arduino include statement to your project the library will be used in your project and you can make simulations with the Arduino board.

Use the SerialInput("x") to read a value out of the serial communication line.

```
if (SerialInput.equalsIgnoreCase("START")){  
    digitalWrite(relais,HIGH);  
}
```

attention:

Remember to adjust the 'Void Setup to initialize the real IO!!

Example: `pinMode (0, OUTPUT);`

`pinMode (1, INPUT);`

In each sketch are the instructions changed, so you only need to connect the inputs and outputs through the screen and the sketch should work.

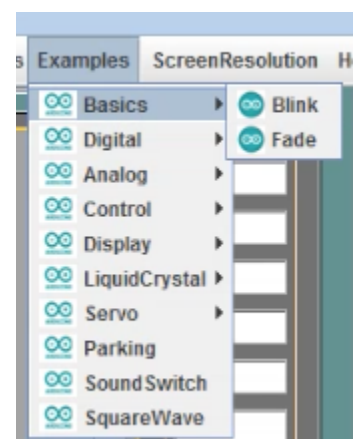
How to use an example:

1. Open an example sketch and upload it into the Arduino board.
2. Add the used components (I/O) in the worksheet. (there are some examples that are already saved in the restore settings)
3. Connect the Arduino IO Simulator with the board
4. Simulate your Arduino in and outputs on the simulator.

HOW TO USE IT

The Arduino Simulator is very easy to understand. The simulator needs 5 simple things in order to work correctly.

1. Connect the Arduino board
2. Upload your custom Arduino code with the corresponding library file
3. Change the original Arduino code
4. Select the used in-outputs in the Arduino Simulator
5. Connect the Arduino Simulator to the Arduino board with the right COM port



1. Connect the Arduino Board

The Arduino IO Simulator is very easy to understand and work with. The Simulator requires 5 simple steps in order to simulate a project.

1. **Connect the Arduino board**
2. **Upload your custom Arduino code with the corresponding library file**
3. **Add the used libraries**
4. **Select the used in-outputs in the Arduino IO Simulator**
5. **Connect the Arduino IO Simulator to the Arduino board with the right serial port**

1. Connect the Arduino Board

The Arduino IO Simulator works with a lot of Arduino boards:

- Arduino UNO
- Arduino Mega
- Arduino Leonardo
- Arduino...

Attention: Only the digital and analog pins that are available on the simulator can be used! Disconnect the Arduino IO Simulator before uploading the Arduino code with the IDE.

2. Upload your custom Arduino code with the corresponding library file

Open the simulator and go to 'Help -> Arduino UNO programming code -> Arduino UNO programming code (ino)'.

This will open a Arduino (ino) file with the corresponding library and important code in it.

3. Add the used libraries

In order to let the Simulator understand the code, we have created our own libraries. To maintain the usability, we have decided to keep the instructions as they are but we changed the libraries a bit so they are compatible with our software.

There are a few libraries available to use. The simulator program library is necessary for the digitalWrite... instructions. To use the 16x2 LCD display you have to add our liquidCrystalSim library in order to use it with the simulator. All the instructions are the same.

4. Select the used in-outputs in the Arduino IO Simulator

Each input and output on the Simulator has a selection box where the used digital or analog pin can be connected.

5. Connect the Arduino IO Simulator to the Arduino board with the right serial port

The Arduino IO Simulator knows which port is the Arduino board.

MAKE SURE THE ARDUINO IS DISCONNECT WHILE UPLOADING THE ARDUINO CODE.

CHECK YOUR PROGRAM WITH CHECKVAR VARIABLES

With Checkvars it is possible to check the status of your variables in the Arduino sketch. You can insert a variable on a line in your sketch and through the serial monitor you can follow the values of the variables.

You can enter many different variables as :

Int, long int, long unsigned int, word, double, float ,char, string, Boolean

Instruction:

CheckVar(num , var);

num: integer from 0 to 32768

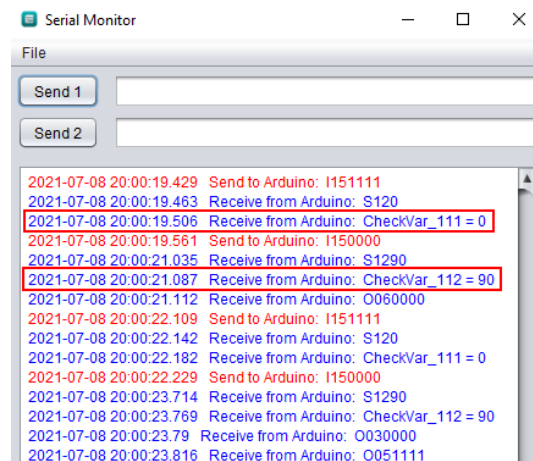
var: Int, long int, long unsigned int, word, double, float, char, string, Boolean

Example sketch parking:

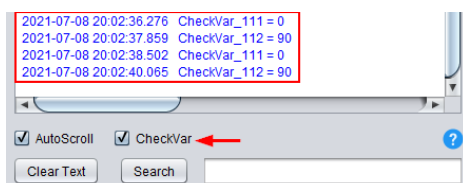
I want to check the variables 'BarUp' and 'BarLow' in the sketch, and I give the CheckVars the numbers 111 and 112 for BarUp and BarLow.

In the serial monitor you can follow the variables CheckVar_111 and CheckVar_112 and their values.

```
if (digitalRead(Exit) == 1)
{
  if (Available != CAPACITY) {
    Available++;
    myservo.write(BarUp);
    CheckVar(111, BarUp);
    delay(1500);
    myservo.write(BarLow);
    CheckVar(112, BarLow);
  }
}
```



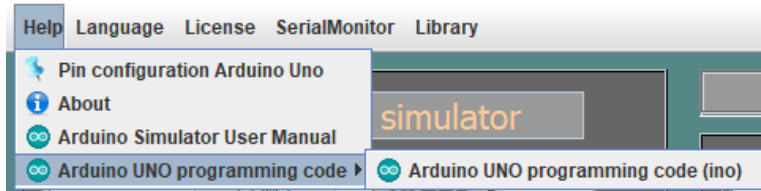
It is also possible to see only the CheckVars by check the box 'CheckVar'.



PREPARING THE ARDUINO UNO PROGRAM

Open a new sketch (xx.ino)

The Simulator UNO-program (.ino) and the Simulator library "SimulatorProgram.h" can found under Help:



Start the "Arduino UNO programming code" application

Now you can set your own code into the Arduino, if its upload in the Arduino you can test it with the Simulator.

Attention: The library "SimulatorProgram.h" stand by the Simulator.

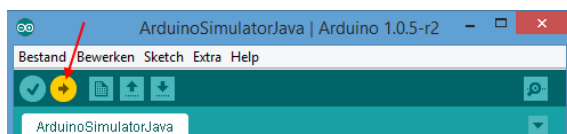
Uploading of a new program to the Arduino board

- Start the Arduino application
- Open the sketch
- Arduino UNO connecting with the pc:



- Select board "Arduino UNO"
- Select the serial port
- Upload the program into the Arduino UNO

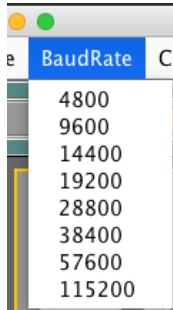
Attention: The BaudRate on the simulator is 9600.



CONFIGURE THE SERIAL PORT

Set the BaudRate

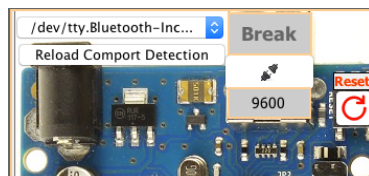
The BaudRate is set by default at 9600 or change the BaudRate in the Arduino code and also in the Simulator.



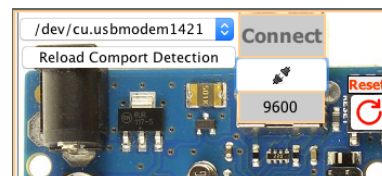
Set the Com port

First, you need to select the serial port, the USB port that is used by the Arduino. The Simulator auto detects the Arduino and turns 'red'.

Before the selection



After the selection



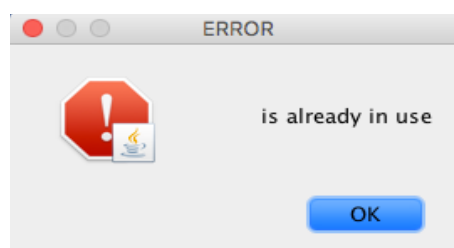
Attention:

- At start-up, we also see the state of the simulator at the bottom of the serial port:
- Once you have selected the correct serial port changes to this text:

Connect to COM PORT

Disconnect from COM Port

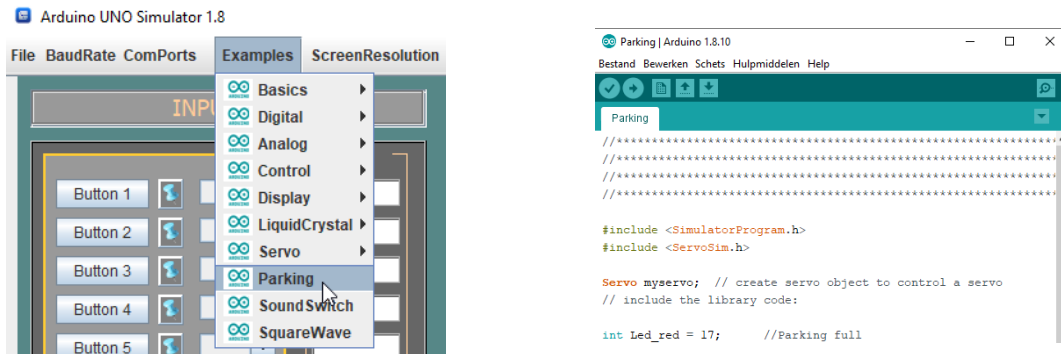
You will get an error message when you want to connect with a pin that is already used.



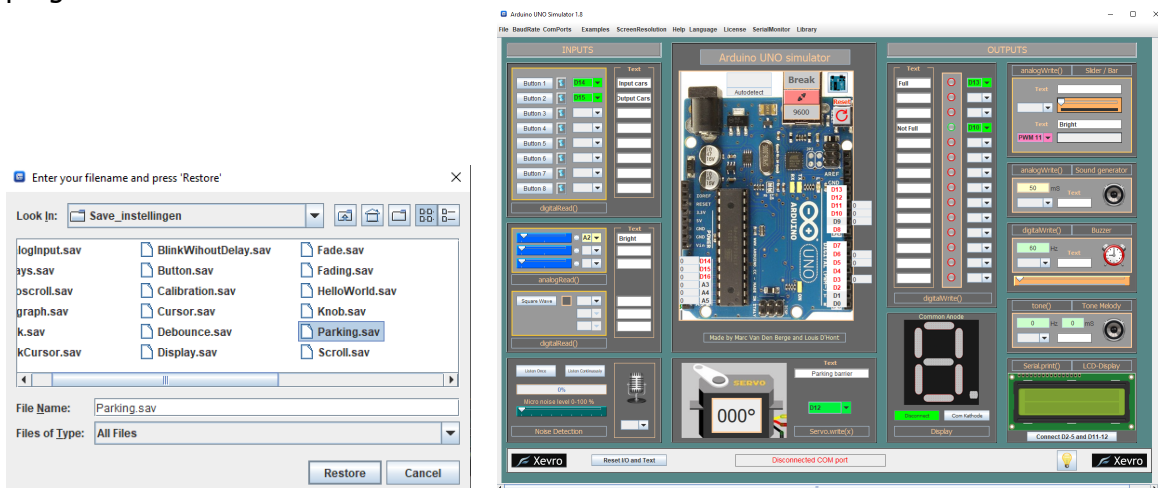
HOW TO RUN A PRE-PROGRAMMED SKETCH

Example: Parking

Select 'Examples' and select 'Parking', the Arduino sketch start up.



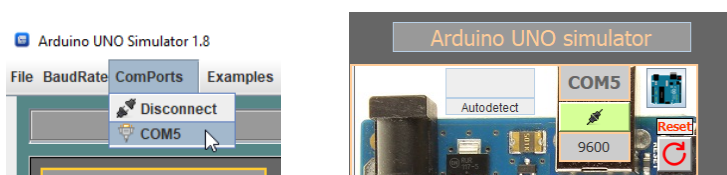
Go to the menu 'File' and select 'Restore Settings', choose the file 'parking.sav' and the pre-programmed tools come on the screen.



The BaudRate is normally 9600, check this in the Arduino sketch.

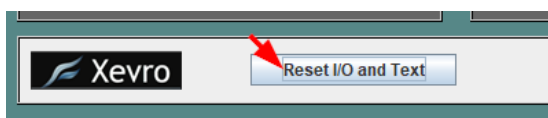
```
Serial.begin(9600);
```

Select the COM port, the connection between your Arduino and the simulator is established (green button) and the program is running



If you want to start a new sketch you do the following moves:

Push the button 'Reset I/O and Text'.



ARDUINO BOARD VIEWER

By clicking on the Arduino board button a window will open in which you will see an Arduino UNO board with all the used IO pins on it. The Reset IO button will clear all the IO pins to start with selecting IO again. The RX/TX LEDs are also visually simulated.

The Arduino UNO board window is set as a top-level window to make sure you can always see the Arduino board while you are simulating projects.

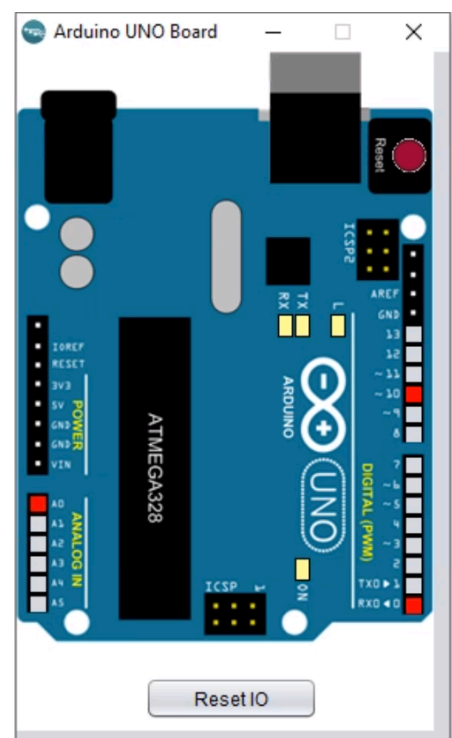
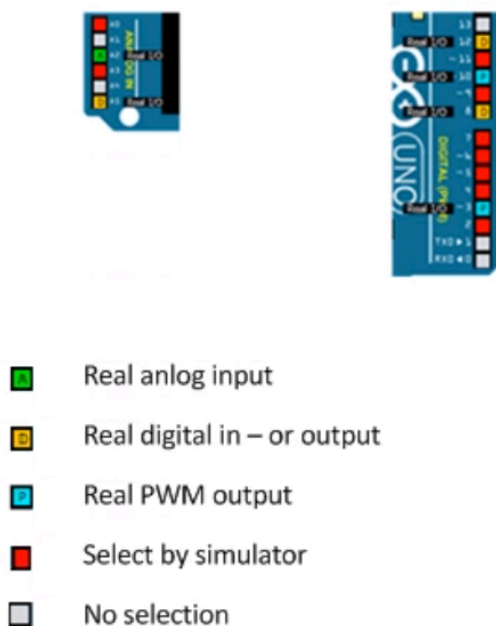
The real IO connect method in the setup will allow the simulator library to use simulated IO together with real IO pins if you want to use connected components on . When you perform a hardware reset on the real Arduino board or a reset from the simulator Arduino board you will lose all real IO connections, to get these connections back you have to enter the following instruction in the setup.

When working with TCP you have to place the instruction 'RealIO_Connect()' after the delay(5000).

```
void setup() {  
  //***** code for SimulatorTCP *****  
  Serial.begin(9600);           // Simulator Serial Com  
  inString.reserve(25);  
  
  while (!Serial) {  
    ; // wait for serial port to connect. Needed for native USB port only  
  }  
  Ethernet.begin(mac, ip, gateway, subnet);  
  server.begin();               // Start the server.  
  Serial.print("Init");  
  delay(5000);  
  RealIO_Connect(); // Used for connection between arduino and real IO , place  
  //lcd.port(portname); // The lcd library needs this portname  
  //myservo.port(portname); // The servo library needs this portname  
  //stepper.port(portname); // The stepper library needs this portname  
  //irrecv.port(portname); // The IRremote library needs this portname  
  //iirsend.port(portname); // The IRremote library needs this portname  
  //***** end code SimulatorTCP *****  
}
```

```
void setup() {  
  // Simulator Serial Connection code  
  Serial.begin(9600);  
  inString.reserve(10);  
  RealIO_Connect(); // Used for connection between
```

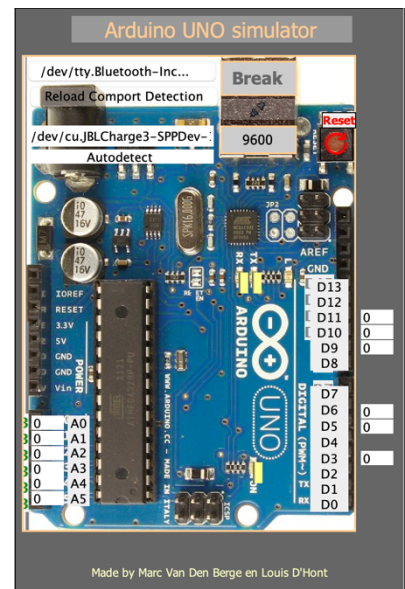
When you click on a pin you can change the status of the pin. To use an analog or digital pin as a real in or output you will have to click ones and it will show a 'D' OR 'A' in it. A PWM pin will show 'P'. Each time you click on a pin, the status will be send to the Simulator library on the Arduino board.



USE THE ANALOG & DIGITAL IO

Digital Inputs

The Arduino UNO has 14 digital IO pins that we can configure into inputs or outputs (IO). These pins get symbolic images as D0 to D13 and the text change red if it's select.

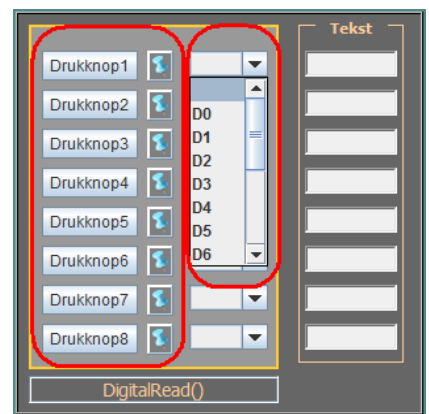


BUTTONS

There are 8 buttons available. The combobox is used to connect the button to one of the 14 IO pins.

The light blue pin can be used to hold down the button while doing other things, the border changes to red when it's pressed.

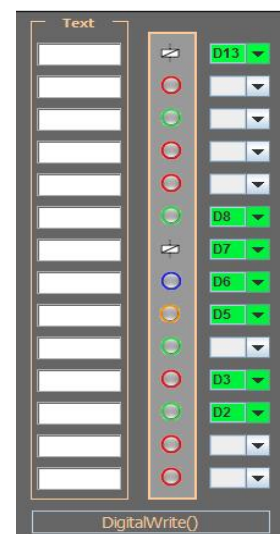
The buttons can be controlled with the **DigitalRead()** function.



LEDS

There are 14 LEDs available, for every pin of the Arduino 1 LED. Use the combobox to connect it with the Arduino. By clicking on the LED you can change the color.

The LEDs can be controlled with the **DigitalWrite()** function.



BUZZER

The buzzer is used to make a noise with a custom frequency. The combobox is used to connect the buzzer with the Arduino.

The buzzer can be controlled with the `digitalWrite();` function. By sending out a `digitalWrite(pin, HIGH);` signal in the Arduino code, the buzzer will make a noise with the adjustable frequency (use the slider to change the frequency).

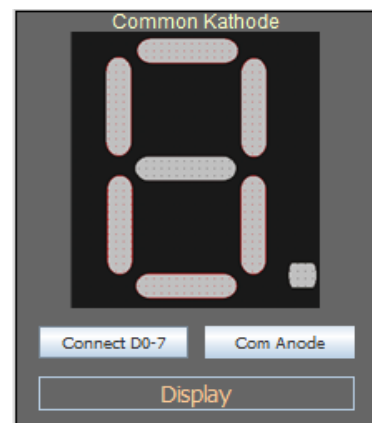


7 SEGMENT DISPLAY

The 7 segment display has 6 digital pins that can connect to D2-8 on the Arduino. The display can be connected in common anode or common cathode.

To light up the display use `digitalWrite(D2-8);`

See the example: Parking.



SLIDERS

There are 3 sliders to connect with one of the 6 analog pins (A0-A5). The sliders can be read by the Arduino with the `analogRead()` function. On the Arduino you have a white box where the slider value is shown.

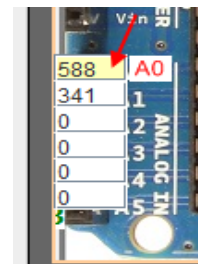
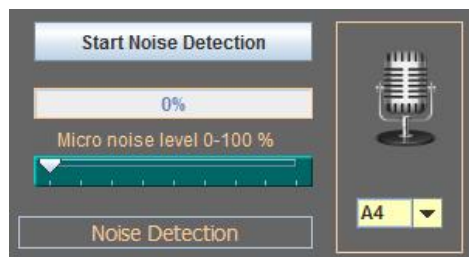


NOISE DETECTION

The noise detection is used to send an analog (0-1023) signal to the Arduino depends on the noise level. The combobox is used to connect the noise detector to one of the 6 analog pins (A0-A5).

When you click on the 'Start Noise Detection' the detection starts listening to the microphone noise level. When the noise level exceeded the slider value then it will sends the signal (0-1023) to the Arduino. The limit value in the Arduino code needs to be lower than the noise detection slider because the signal will be send when the noise is detected.

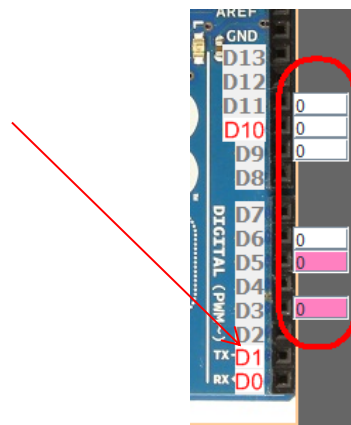
The noise detection can be controlled with the `analogRead()` function.



BARGRAPH

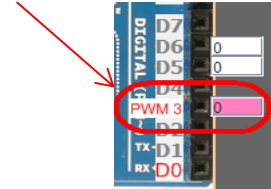
The bargraph can be connected to one of the 6 digital PWM pins of the Arduino. The bargraph shows the % of your value (0-1023), this can be used to simulate a PWM signal as a % bar.

Use `analogWrite(pin, value);` to control the bargraph (See example: sound switch).



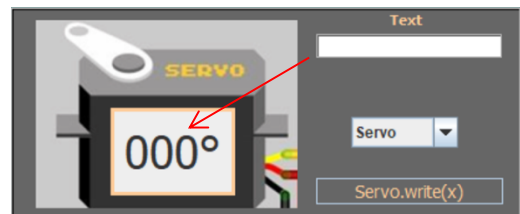
SOUND GENERATOR

The sound generator can be connected to one of the 6 digital PWM pins of the Arduino. By changing the time (ms) you change the duration that the sound goes off (1ms – 10 000ms). The frequency can go from 10hz to 10Khz. Use `analogWrite(pin, value);` to control the sound generator.



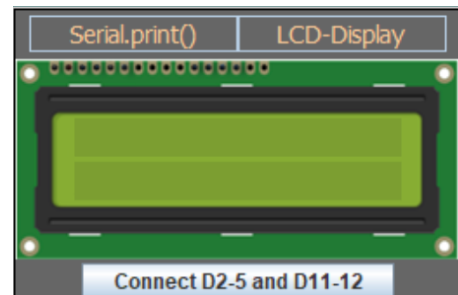
SERVO

The servo can be connected to one digital pin (D2-13) of the Arduino. The number of degrees (°) is visible in the servo. Click on the servo to make the servo smaller through removing the background and combobox. Use `servo.write()` and `servo.attach()`. Add the servo simulator library to use it.



LCD DISPLAY

The LCD display can be connected to the Arduino by connecting D2-5, D11 and D12. Add the simulator LiquidCrystalSim library to make it work with the simulator.



TONE MELODY

The tone melody can be connected to digital pin D8 of the Arduino. The frequency and time of the sound (milliseconds) are present in the light green boxes. Use `tone(8, f, d);` and `noTone(8);` (See example: Tone Melody)

`noTone()` stops playing sound.

f = frequency

d = Duration

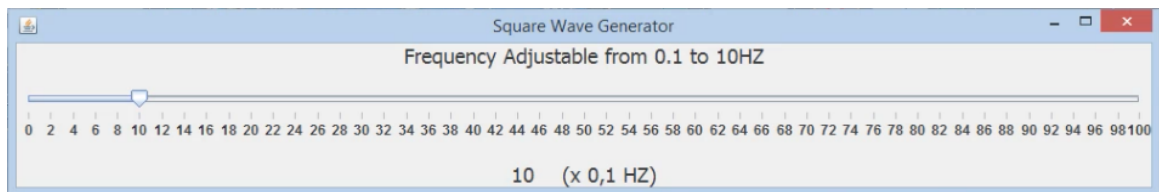
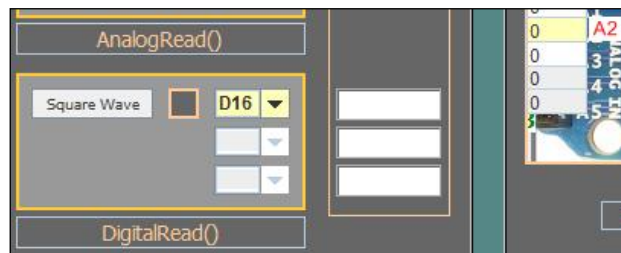


SQUARE WAVE GENERATOR

The square wave sends pulls signals to the Arduino, when the signal is high you see the grey square lights up 'red'. The combobox is used to connect the squarewave to one of the 6 analog pins (A0-A5 = D14-19).

When you click on the 'SquareWave' button there opens a second window with a slider to change the frequency.

The square wave can be controlled with the digitalRead() function.



A FEW THINGS WHILE PREPARING THE ARDUINO PROGRAM

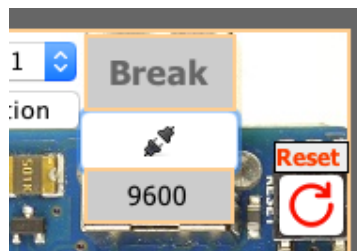
Always upload the sketch to the Arduino UNO.

if the simulator is connected with the Arduino you can't upload the Arduino program. We made a tool 'Disconnect' which closes the connection with the serial port of the simulator so that you can upload the sketch to the Arduino.

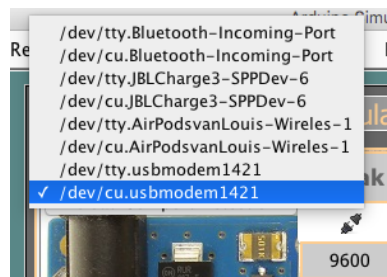
The great advantage of this is that we don't need to shut down the Simulator whenever we want to upload the sketch simulator.

After downloading the simulator, we connect again with the serial port.

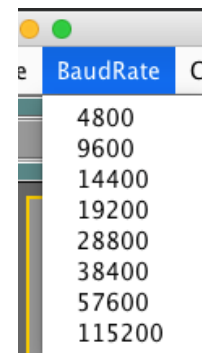
Disconnect serial port



Choose serial port



BaudRate



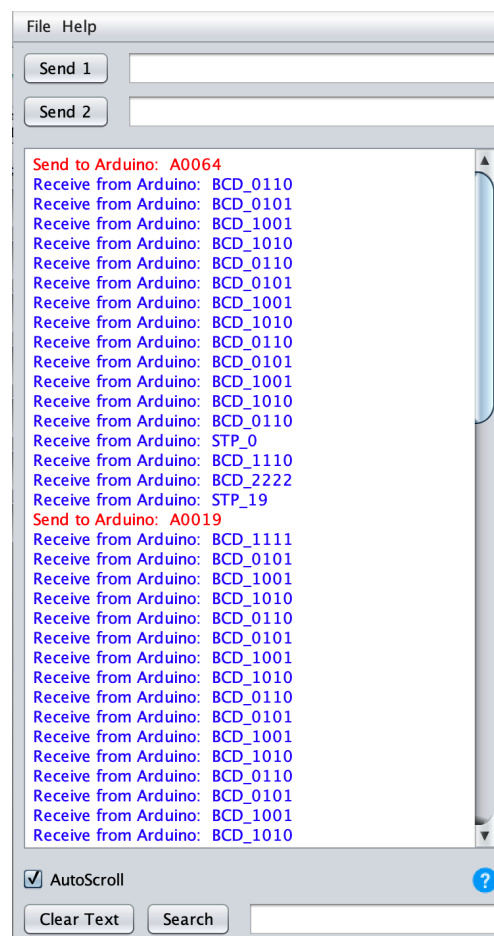
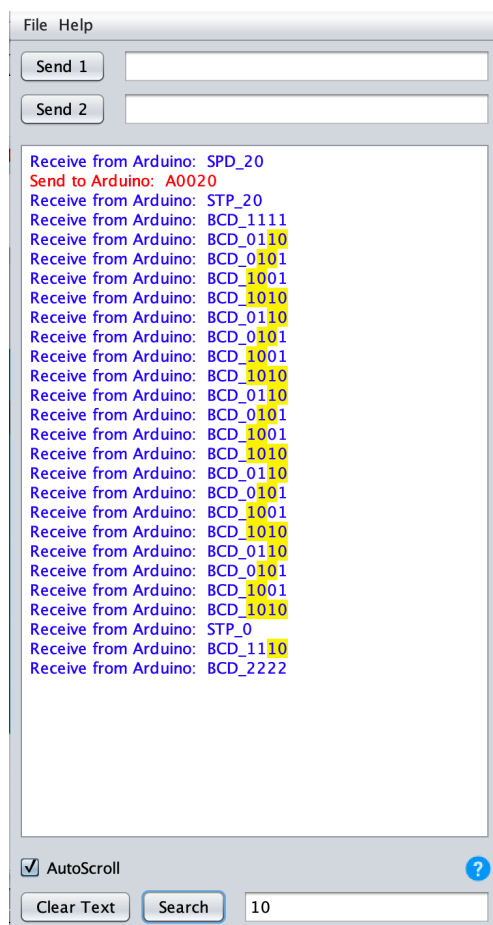
SERIAL MONITOR

In the simulator, it's now also possible to view the serial data. There are 2 buttons that you can use to send a signal to the Arduino. The monitor shows you a 'Receive from Arduino' line when the Arduino sends data to the simulator and a 'Send to Arduino' line will be shown when you send something to the Arduino with the simulator.

If you want to save the serial monitor output you can save the whole text or a selected area. With the search function, it's possible to search for a specific word or character. if the word or character is found, it will be highlighted in yellow. By clicking on the blue question mark you get the explanation of all the simulator codes.

By using Serialprint(); in the Arduino IDE, you can send a serial message.

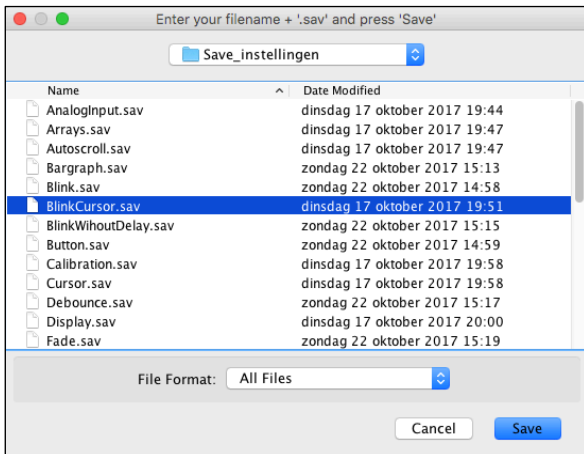
The serial message will start with 'txt_' in the serial monitor.



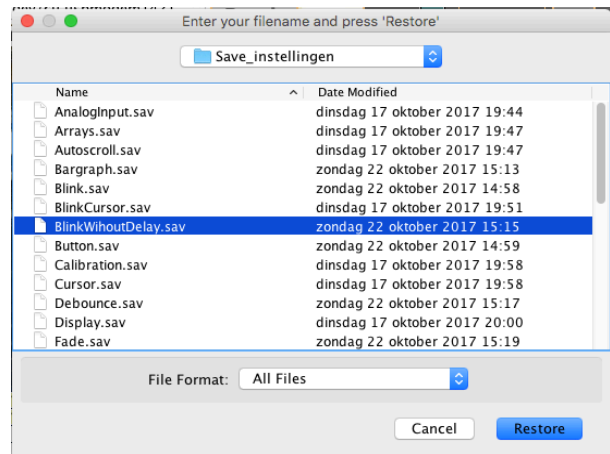
SAVE AND RESTORE OF SETTINGS

With 'Save' you can save your selected I/O and dictated texts.
the 'Restore' button restores the settings to make it easy to use.
We can save the filename of the extension with *.sav or *.txt.
You find the 'Save and Restore' function under 'File'.

Save



Restore



SCREEN RESOLUTION

We have 6 options:

- 13 inch MacBook pro size
- Resolution: 1024 x 768
- Resolution: 1336 x 768
- Resolution: 1920 x 1080
- Full Screen (iMac)
- Variable resolution (min: 500, max: 2000)

