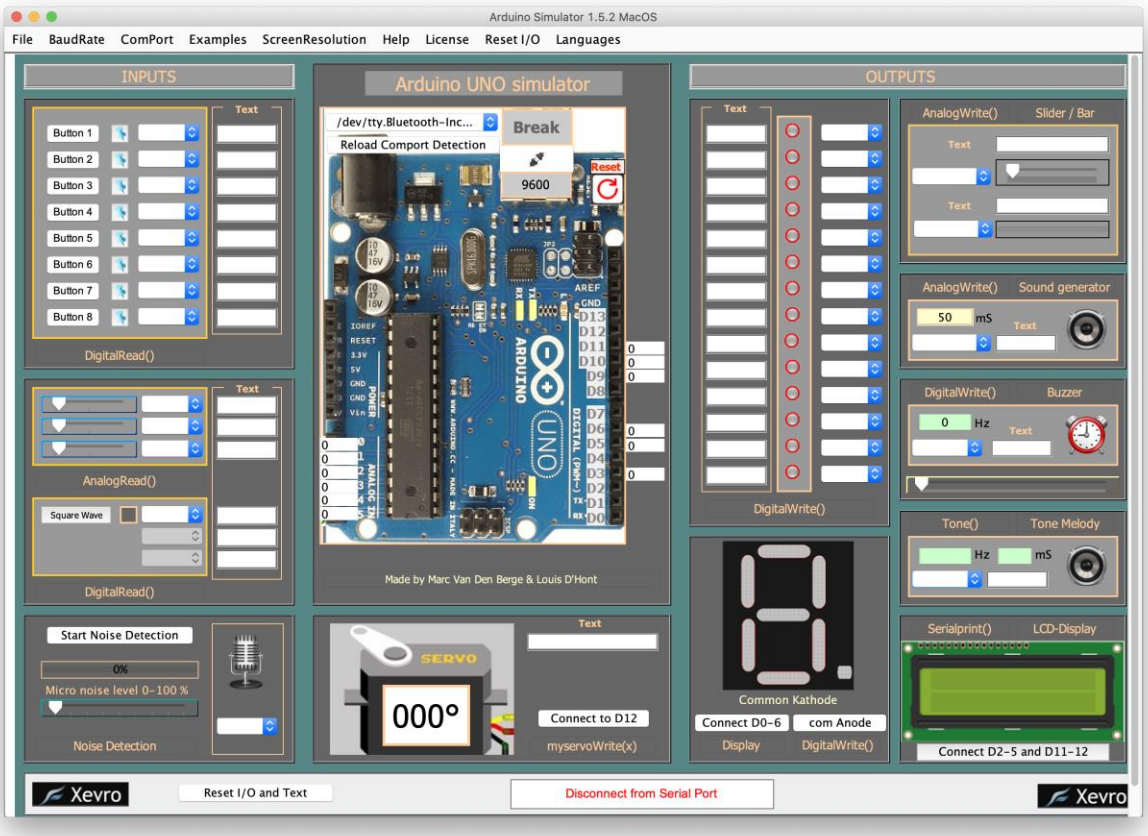
ARDUINO IO SIMULATOR MACOS
USER MANUAL

XEVRO

Version 1.5.3

Xevro© 2019

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.

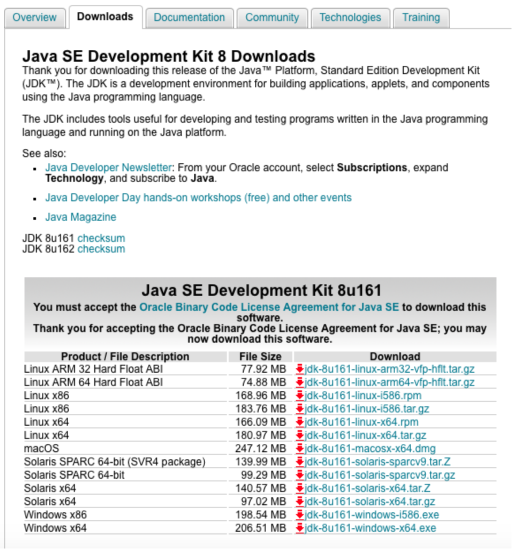
This Arduino IO Simulator is designed to test an Arduino program quickly with the Arduino board without really having connections to external IO (buttons, potentiometers, LEDs, LCD displays, ...) and add a nice custom drawing around it to get a better simulation experience.

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



Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 8u161 checksum
JDK 8u162 checksum

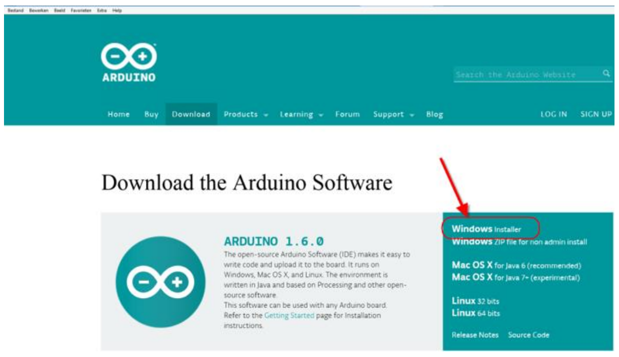
Java SE Development Kit 8u161

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

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

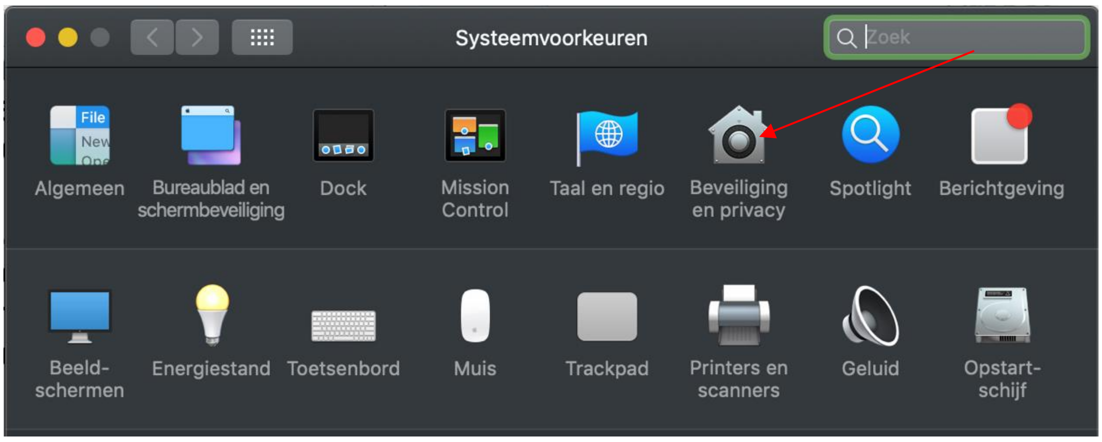
ARDUINO 1.6.0

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.

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. The first time when you start the Arduino Simulator you will get a window pop-up that asks to give your password. This is to change the permissions of the Java/extensions folder where the Serial driver needs to come. It will close automatically after type in your password. This will just happen ones.

4. Drag the Arduino Simulator 1.5 MacOS in the Programs folder.

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

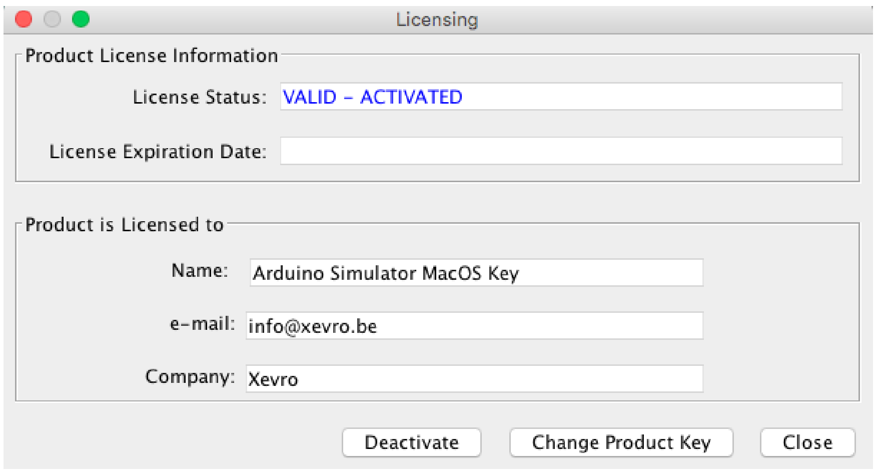
If you bump into a mistake or problem, please send me an e-mail to info@xevro.be

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'.



Licensing

Product License Information

License Status: VALID – ACTIVATED

License Expiration Date:

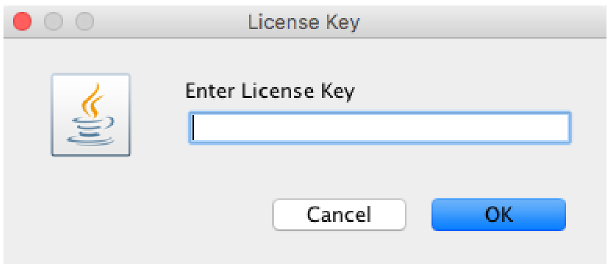
Product is Licensed to

Name: Arduino Simulator MacOS Key

e-mail: info@xevro.be

Company: Xevro

License key input field



License Key

Enter License Key

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 serial port

1. Connect the Arduino Board

The Arduino Simulator works with a lot of Arduino boards:

- Arduino UNO
- Arduino Mega
- Arduino Nano
- Arduino Micro
- Arduino Leonardo
- Arduino ...

Only the digital and analog pins that are available on the Simulator can be used!
Disconnect the Arduino Simulator before uploading the Arduino code.

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 an Arduino (ino) file with the corresponding library and important code in it.

3. Change the original Arduino code

In order to let the Simulator understand the code, we have devised our own instructions. To maintain usability, we have decided to change the current instructions a little bit and replace only the first letter with a capital letter. We have modified the instructions with a point so that the point may be omitted.

```
digitalWrite(13, HIGH); -> DigitalWrite(13, HIGH);

lcd.print("x"); -> lcdprint("x");
```

4. Select the used in-outputs in the Arduino 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 Simulator to the Arduino board with the right serial port

The Arduino Simulator knows which port is the Arduino board.
Make sure the Arduino is disconnected while uploading the Arduino code.

CODE CHANGES

The Arduino IDE works with instructions that the IO read and write, we must convert this to serial communication, now we need to write new instructions for this.

We don't want to change the real instructions so we decided to make the first character as a capital, the instructions with a point in it are changed with no point.

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

Example 1:

```
         digitalWrite(12, HIGH);
Become   DigitalWrite(12, HIGH);
```

```
Example 2:  lcd.print("Hello world");
Become      lcdprint("Hello world");
```

The first letter is now a case of removing the tip:

Instructions Arduino		**Instructions Arduino IO Simulator**	
digitalWrite(x,y);	=	DigitalWrite(x,y);	attention: capital letter
digitalRead(x);	=	DigitalRead(x);	attention: capital letter
analogWrite(x,y);	=	AnalogWrite(x,y);	attention: capital letter
analogRead(x);	=	AnalogRead(x);	attention: capital letter
myservo.write(x);	=	myservowrite(x);	attention: the '.' remove
Serial.print(x);	=	Serialprint(x);	attention: the '.' remove
tone(x,y ,z);	=	Tone(x,y ,z);	attention: capital letter
noTone (x);	=	NoTone(x);	attention: capital letter
lcd.setCursor(x,y);	=	lcdsetCursor(x,y);	attention: the '.' remove
lcd.print(long x);	=	lcdprint(x);	attention: the '.' remove
lcd.print("x");	=	lcdprint("x");	attention: the '.' remove
lcd.autoscroll();	=	lcdautoscroll();	attention: the '.' remove
lcd.noAutoscroll();	=	lcdnoAutoscroll();	attention: the '.' remove
lcd.clear();	=	lcdclear();	attention: the '.' remove
lcd.display();	=	lcddisplay();	attention: the '.' remove
lcd.noDisplay();	=	lcdnoDisplay();	attention: the '.' remove
lcd.blink();	=	lcdblink();	attention: the '.' remove

lcd.noBlink();	=	lcdnoBlink();	attention: the '.' remove
lcd.write(x);	=	lcdwrite(x);	attention: the '.' remove
lcd.rightToLeft();	=	lcdrightToLeft();	attention: the '.' remove
lcd.leftToRight();	=	lcdleftToRight();	attention: the '.' remove
lcd.home();	=	lcdhome();	attention: the '.' remove
lcd.cursor();	=	lcdcursor();	attention: the '.' remove
lcd.scrollDisplayLeft();	=	lcdscrollDisplayLeft();	attention: the '.' remove
lcd.scrollDisplayRight();	=	lcdscrollDisplayRight();	attention: the '.' Remove

What to do:

1. Open a new Arduino sketch
2. Add the serial connections
3. Add the 'ArduinoSimulator.h' library
4. Change your code with the right instructions (see above)
5. Upload your code and connect the simulator to the Arduino!

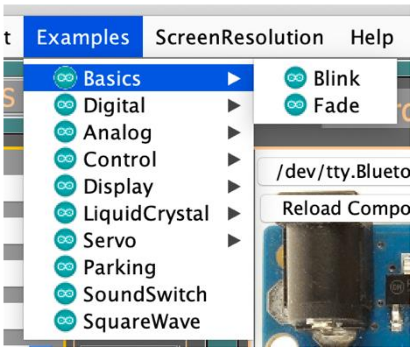
attention:
Remember to adjust the 'Void Setup to initialize the real IO!!

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:

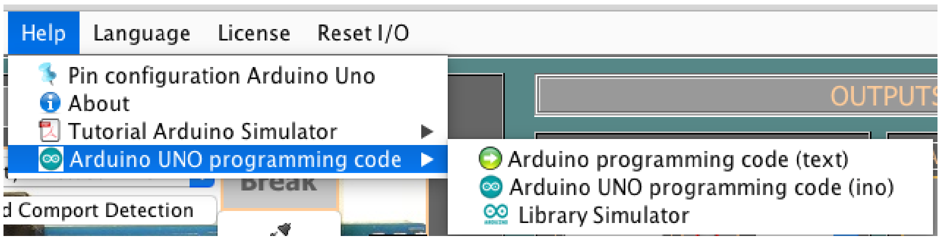
1. Open an example sketch and upload it into the Arduino board.
2. Add the used components (I/O) in the worksheet. (there are some examples that are already saved in the restore settings)
3. Connect the Arduino IO Simulator with the board
4. Simulate your Arduino in and outputs on the simulator.



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:


Help
Language
License
Reset I/O
Pin configuration Arduino Uno
About
Tutorial Arduino Simulator
Arduino UNO programming code
Arduino programming code (text)
Arduino UNO programming code (ino)
Library Simulator
OUTPUTS
Break
d Comport Detection

Start the “Arduino UNO programming code” application
Now you can set your own code into the Arduino if it’s upload in the Arduino you can test it with the Simulator.

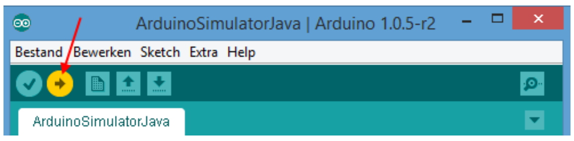
Attention: The library “SimulatorProgram.h” stand by the Simulator.

Uploading of a new program to the Arduino board

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

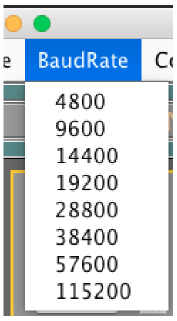
- Select board “Arduino UNO”
- Select the serial port
- Upload the program into the Arduino UNO

Attention: The BaudRate on the simulator is 9600.



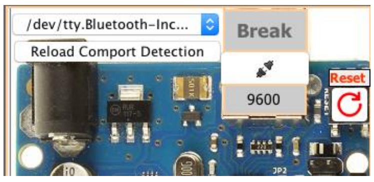
CONFIGURE THE SERIAL PORT

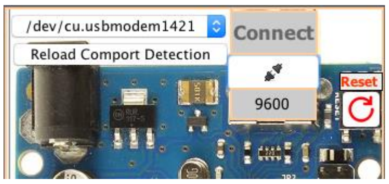
Set the BaudRate
The BaudRate is set by default at 9600 or chance the BaudRate in the Arduino code and also in the Simulator.



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'.

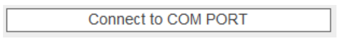
Before the selection **After the selection**



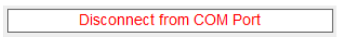


Attention:

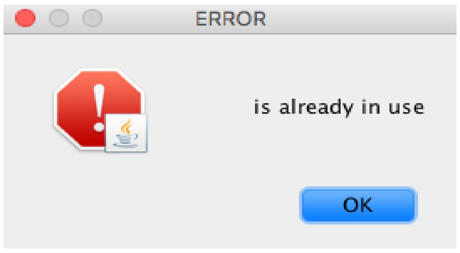
- At start-up, we also see the state of the simulator at the bottom of the serial port:



- Once you have selected the correct serial port changes to this text:



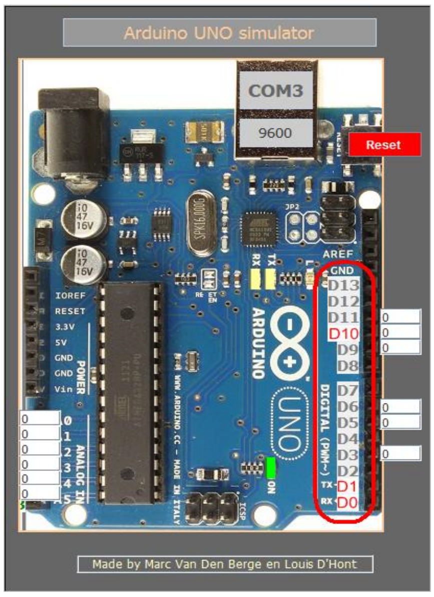
You will get an error message when you want to connect with a pin that is already used.



ERROR

is already in use

USE THE ANALOG & DIGITAL IO

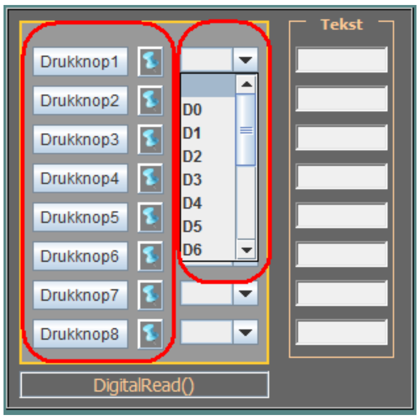


Digital Inputs

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.

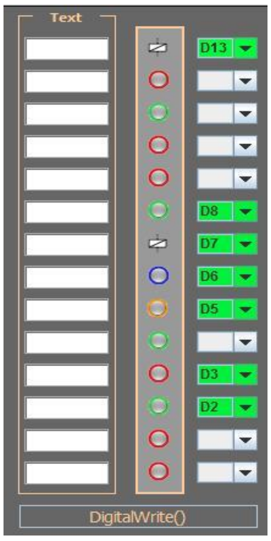


The light blue pin can be used to hold down the button while doing other things, the border changes to red when it's pressed.

The buttons can be controlled with the **D**igitalRead() 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.

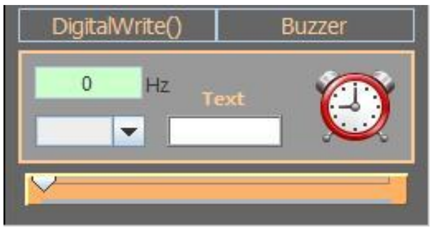


The LEDs can be controlled with the **D**igitalWrite() function.

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 **D**igitalWrite(); 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).

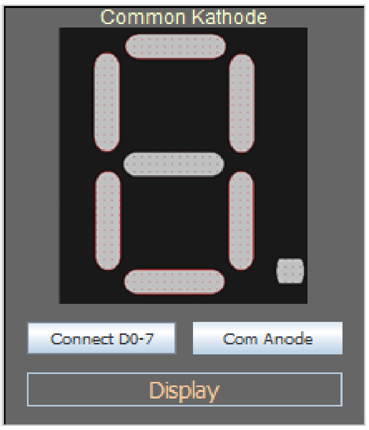


7 SEGMENT DISPLAY

The 7 segment display has 7 digital pins that can connect to D0-6 on the Arduino. The display can be connected in common anode or common cathode.

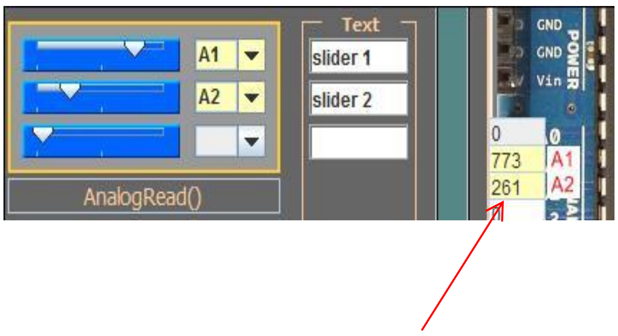
To light up the display use **D**igitalWrite(D0-6);

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 **A**nalogRead() function. On the Arduino you have a white box where the slider value is shown.

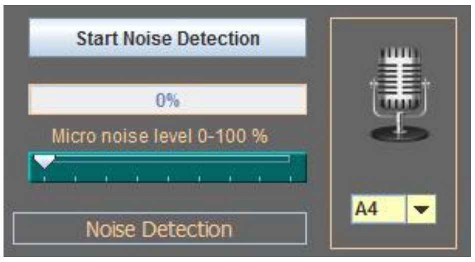


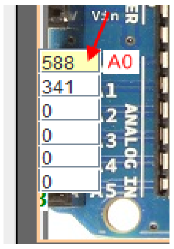
NOISE DETECTION

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 send 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 sent when the noise is detected.

The noise detection can be controlled with the **A**nalogRead() function.

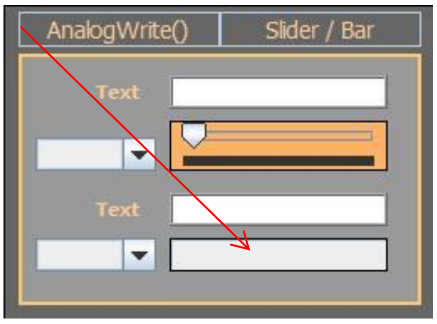


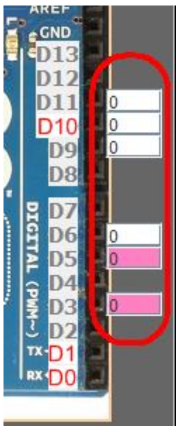


BARGRAPH

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



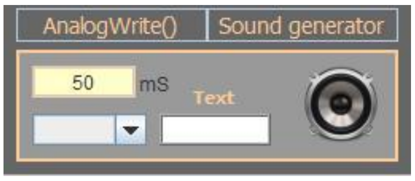


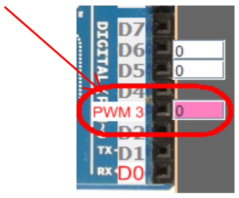
SOUND GENERATOR

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.

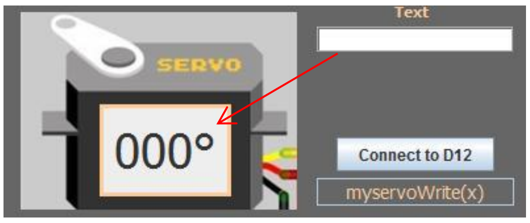
Use **A**nalogWrite(pin, value); to control the sound generator.





SERVO

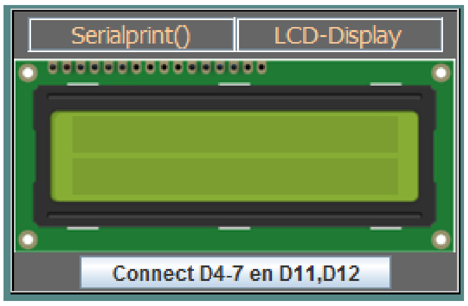
The servo can be connected to one digital pin (D12) of the Arduino. The amount of degrees (°) are visible in the servo. With the MyservoWrite() you can control servo.



LCD DISPLAY

The LCD display can be connected to the Arduino by connecting D2-5, D11 and D12. All the instruction that controls the LCD are changed by removing the '.' point.

example: lcd.print() = lcdprint().



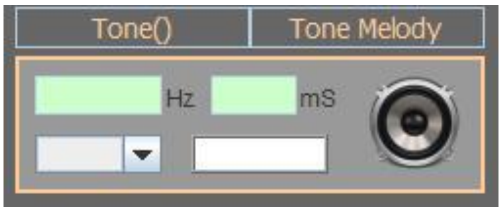
TONE MELODY

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

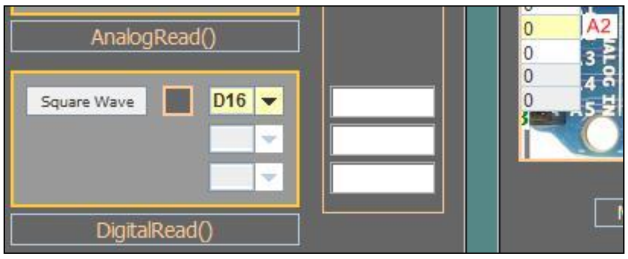


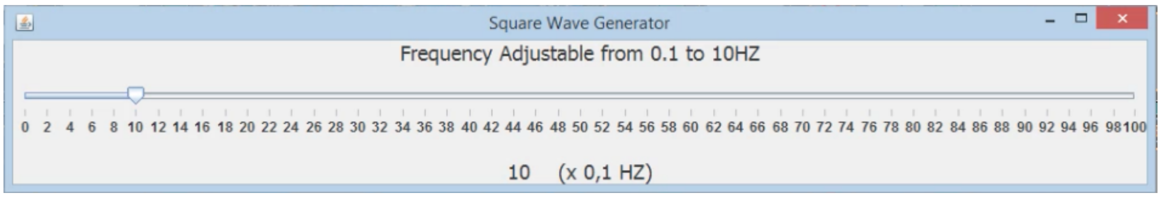
SQUARE WAVE GENERATOR

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 **D**igitalRead() function.





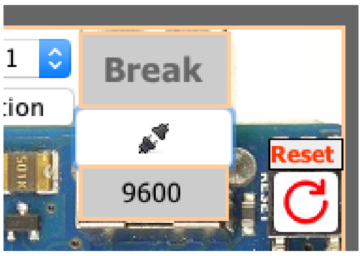
A FEW THINGS WHILE PREPARING THE ARDUINO PROGRAM

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.

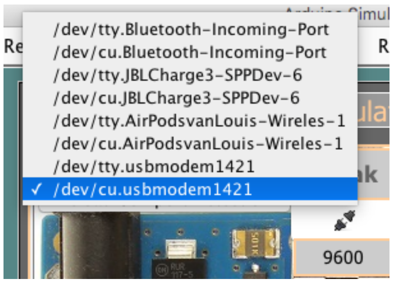
The great advantage of this is that we don’t need to shut down the Simulator whenever we want to upload the sketch simulator.

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

Disconnect serial port

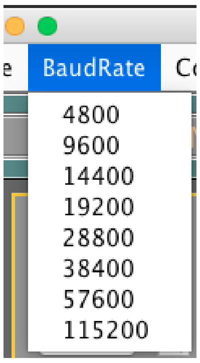


Choose serial port



```
/dev/tty.Bluetooth-Incoming-Port
/dev/cu.Bluetooth-Incoming-Port
/dev/tty.JBLCharge3-SPPDev-6
/dev/cu.JBLCharge3-SPPDev-6
/dev/tty.AirPodsvanLouis-Wireles-1
/dev/cu.AirPodsvanLouis-Wireles-1
/dev/tty.usbmodem1421
✓ /dev/cu.usbmodem1421
```

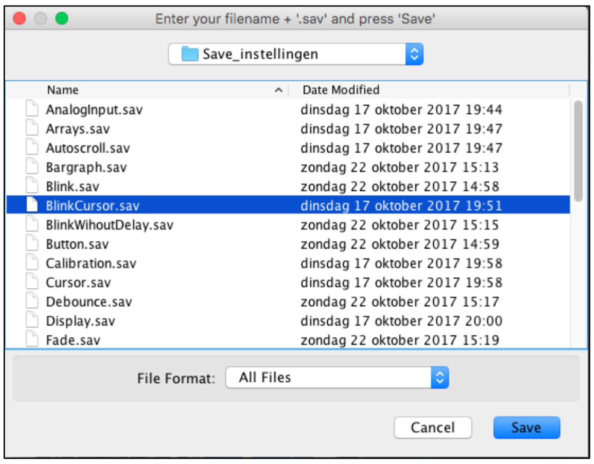
BaudRate



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



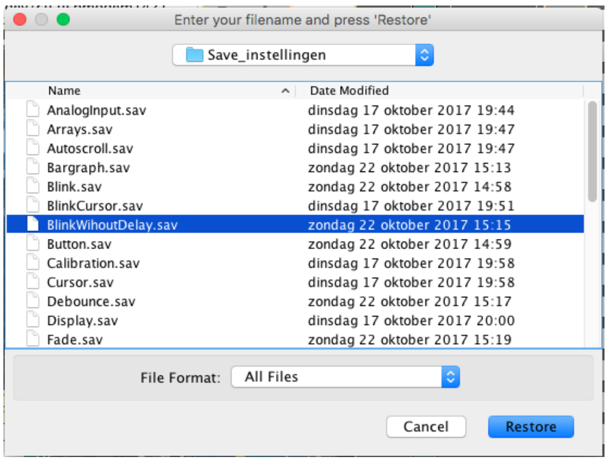
Enter your filename + '.sav' and press 'Save'

Save_instellingen

Name	Date Modified
AnalogInput.sav	dinsdag 17 oktober 2017 19:44
Arrays.sav	dinsdag 17 oktober 2017 19:47
Autoscroll.sav	dinsdag 17 oktober 2017 19:47
Bargraph.sav	zondag 22 oktober 2017 15:13
Blink.sav	zondag 22 oktober 2017 14:58
BlinkCursor.sav	dinsdag 17 oktober 2017 19:51
BlinkWihoutDelay.sav	zondag 22 oktober 2017 15:15
Button.sav	zondag 22 oktober 2017 14:59
Calibration.sav	dinsdag 17 oktober 2017 19:58
Cursor.sav	dinsdag 17 oktober 2017 19:58
Debounce.sav	zondag 22 oktober 2017 15:17
Display.sav	dinsdag 17 oktober 2017 20:00
Fade.sav	zondag 22 oktober 2017 15:19

File Format: All Files

Restore



Enter your filename and press 'Restore'

Save_instellingen

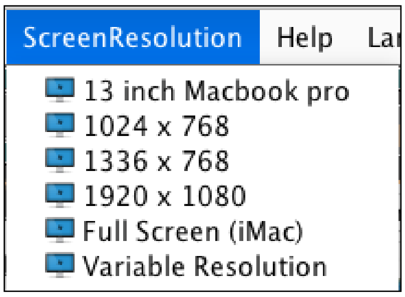
Name	Date Modified
AnalogInput.sav	dinsdag 17 oktober 2017 19:44
Arrays.sav	dinsdag 17 oktober 2017 19:47
Autoscroll.sav	dinsdag 17 oktober 2017 19:47
Bargraph.sav	zondag 22 oktober 2017 15:13
Blink.sav	zondag 22 oktober 2017 14:58
BlinkCursor.sav	dinsdag 17 oktober 2017 19:51
BlinkWihoutDelay.sav	zondag 22 oktober 2017 15:15
Button.sav	zondag 22 oktober 2017 14:59
Calibration.sav	dinsdag 17 oktober 2017 19:58
Cursor.sav	dinsdag 17 oktober 2017 19:58
Debounce.sav	zondag 22 oktober 2017 15:17
Display.sav	dinsdag 17 oktober 2017 20:00
Fade.sav	zondag 22 oktober 2017 15:19

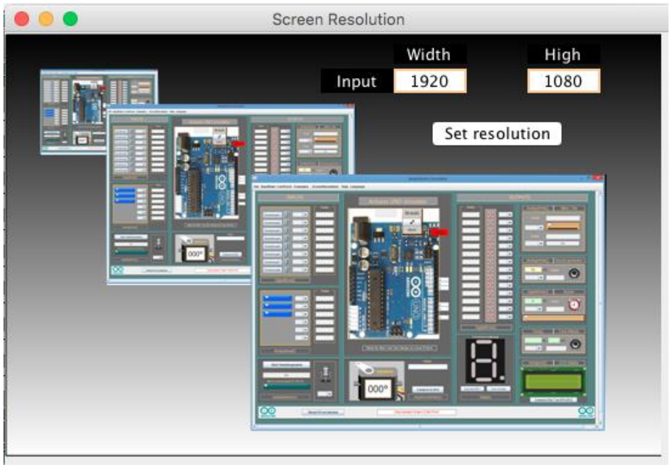
File Format: All Files

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)





LANGUAGES

We have 20 languages available, 5 languages are in the list but will be added to the program. The languages that will be available soon are: Ukrainian, Polish, Hindi, Italian and Russian

- Arabic
- Chinese (traditional)
- German
- English
- French
- Hungarian
- Hindi
- Italian
- Japanese
- Dutch
- Ukrainian
- Polish
- Portuguese
- Russian
- Slovak
- Spanish
- Thai
- Czech
- Turkish
- Vietnamese