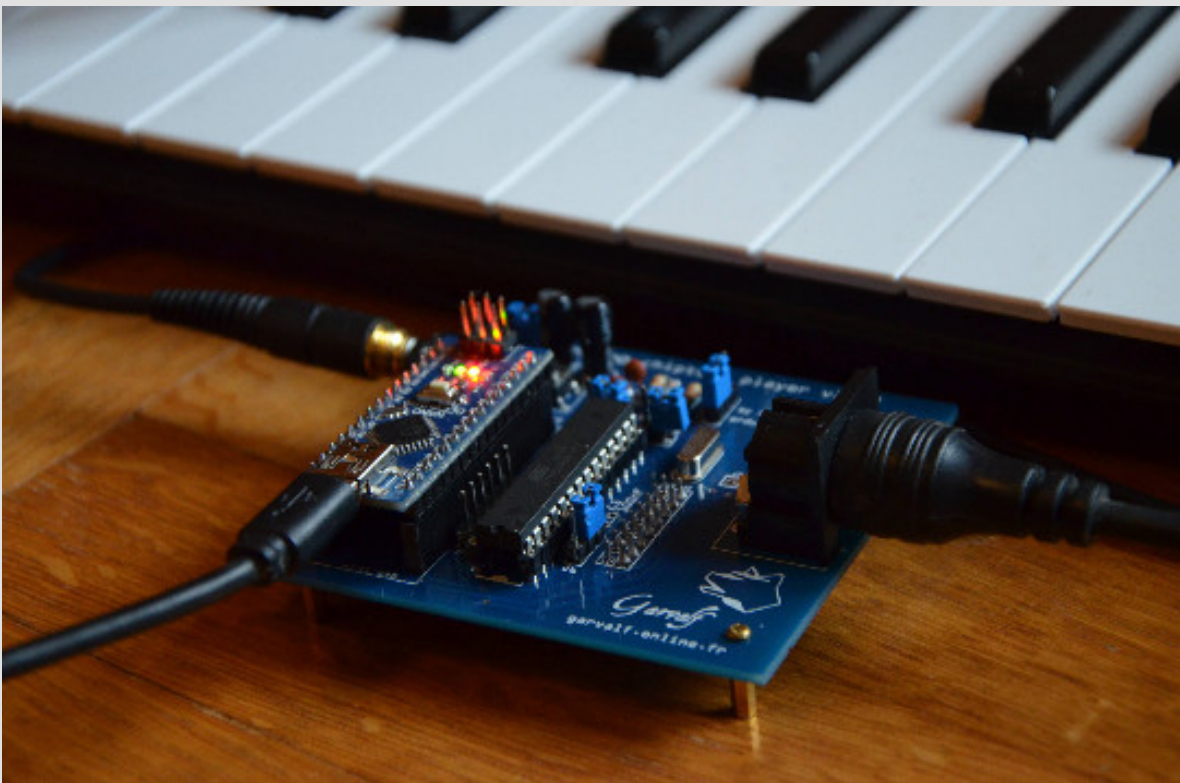
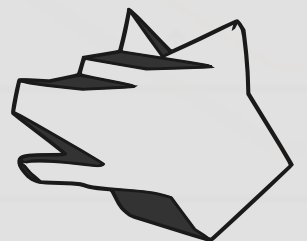


# Garvino Manual

music, sounds and chiptunes creator



Garvalf



GARVUINO

March 20, 2019

# Contents

|   |                 |
|---|-----------------|
| <b>Presentation of the board</b>        | <b>iv</b>       |
| AY-3-8910 emulator . . . . .            | iv              |
| Sid emulator . . . . .                  | iv              |
| 1-bit music . . . . .                   | iv              |
| Mozzi synth . . . . .                   | v               |
| Hardware . . . . .                      | v               |
| <br><b>Flashing the Garvuino board</b>  | <br><b>ix</b>   |
| Flashing for AY music . . . . .         | x               |
| Sid emulator . . . . .                  | x               |
| Flashing for 1-bit music . . . . .      | x               |
| Flashing for Mozzi synth . . . . .      | xi              |
| Flashing other sketches . . . . .       | xi              |
| <br><b>Assembly the Garvurino board</b> | <br><b>xiii</b> |
| Bill of Materials . . . . .             | xiii            |
| Atmega8 setup and flashing . . . . .    | xiv             |
| Breadboard . . . . .                    | xv              |
| <br><b>Expansion board</b>              | <br><b>xvi</b>  |

Thank you for your interest in Garvuino! The Garvuino is a PCB board for creating music, sounds and chiptunes, either through programming, soundtracker or live recording (midi).

As the hardware is open-source, you can modify, hack and have fun with it without restriction.

The PCB, schematics, code and further instructions can be found on <https://github.com/farvardin/garvuino/>



# Presentation of the board

Several modes can be used:

## AY-3-8910 emulator

On Atmega8 chip, driven by the Arduino nano, from the [AVR-AY project](#)

- It can replay Sinclair ZX Spectrum, Atari ST, Amstrad chiptunes from the sd card. You can create thoses chiptunes with Vortex Tracker II or Arkos Tracker.
- It's possible to use it as a (very simple and dirty) AY soundbox, driven by some external captors (light sensor). But it might not be possible to controle it with MIDI.

## Sid emulator

On Arduino, from the [sid emulator lib](#)

- It can replay some sid files, from the arduino memory, which is limited (so tunes will be cut). We'd like to find a way to stream the tune from the sd card, it's probably possible.
- It can be used as a Sid synth, driven by midi.

## 1-bit music

From various 1-bit engines (on arduino)

- It can play music from the arduino memory. 1-bit music don't need much memory so a whole song, even a complex one, can fit in the memory. You can create those songs with Beepola or MilkyTracker.
- It might be possible to stream data from SD if the 1-bit music engine chosen is the same for all the songs.
- It might be possible to create a 1-bit synth in the future. Probably not an easy task.

## Mozzi synth

- [Mozzy synth](#), can output fun FM sounds and many other effects.

## Hardware

The board has 2 chips :

- an Arduino Nano for the compute (it is a complete board for development)
- an optional Atmega8 for AY sound output.

A few components :

- a 24 Mhz crystal that is driving the Atmega8
- some capacitors and resistors to filter the output
- a SD card reader to read AY music files (optional)
- a DIN5 midi input to use the board as a midi synth (optional)



There are a few jumpers on the board. They are for switching between some modes.

When used in AY mode, the mono jumper should be off, when used in beeper, sid or mozzi mode the mono jumper should be on, the J4 (ay chip ground, v1.09 board only) should be off. Also the AY>L and AY>R can be off to reduce noise (but it will work anyway). And TX could be off as well when flashing the arduino chip (you can leave it on, but if the transfer fails, try to remove it)

- mono: used in mozzi, 1-bit and sid mode. You can remove the jumper in AY mode because it's in stereo (but you might also prefer using the mono mode)
- TX: used for transmitting data to the Atmega8 chip in AY mode. You can remove the jumper when not in AY mode. You can also keep it, it doesn't matter, unless you have some problem for flashing the arduino board. On some boards, the jumper is below the board for easier access to the push button.

- AY>L and AY>R: used in AY mode. You can cut a voice if you remove the jumper. If you remove the jumper when not in AY mode, you'll be sure there won't be extra noise from the Atmega8 chip.
- J1: extra midi input GND (below) and midi wire nb 5 if you don't want to use the DIN5 plug.
- J2/J3: for connecting regular (big) SD reader. Most board will use the smaller SD reader
- J4 / ay chip GND / arduino chip (v1.09 board only): ay chip used in AY mode. It's better to put the jumper in the "arduino chip" mode (bottom), when not using AY otherwise you'll get extra and residual noise from the atmega8 chip.
- J5 (v1.09 board only): not used. Linked to midi 4 wire "just in case".
- J6 (v1.09 board only): not used. Could be used to switch off the atmega8 but since there are still some residual noise without it, it can be kept connected. It's been soldered on most boards.
- J7 (v1.09) / J11 (v2.10): can be used for programming the Arduino for Analog input (potentiometer, light sensor for example)
- J8: not used. Can be connected to arduino +5V, RST, GND and VIN but shouldn't be necessary.
- J9: can be used for programming the Arduino for digital output (more leds for example)
- J10: not used. Extra digital output.
- Led: for monitoring some infos

You can of course connect audio out to the mini-jack, but it's also possible to connect a small buzzer (HP), for example 8 ohms / 1 W to the AY>L (under the L) and to the GND (for example in the middle 3-pin J4 or on the GND of J1, if it's installed) then you'll get very cheap sound. It can be useful for monitoring.



Midi: You can plug a DIN5 midi cable into the dedicated port. When removing it, please do it slowly, to avoid tearing everything apart. And when plugging it in and out, hold the midi port with your hand to avoid extra pressure on the connexions.

You can also use [hairless midi serial bridge](#) (windows, mac os x, linux) or [ttymidi](#) (linux) to create a virtual connection going through serial port. You'll need to adjust the arduino sketch to enable this (search for hairless midi or ttymidi in the sketch, or for `#define USBMIDI` at the begining of the sketch).

## Flashing the Garvuino board

This board is supposed to work extensively with the Arduino IDE (<http://arduino.cc/>). You must download it and use it for developping, changing sounds and engines. The only exception is if you only intend to use the board as a AY player, then you can just change the tunes on the SD card.

Programs on Arduino are called sketches. Most sketches are tested with Arduino IDE 1.8.# but they should work with IDE 1.6.#

Get the sketches from there:

<https://github.com/farvardin/garvuino/archive/master.zip>

For flashing, just launch the Arduino IDE, connect the arduino to your computer using an USB mini cable, select in the tools>arduino nano, atmega328, select the right port.

Load a sketch from the `arduino_sketches/` folder.

For flashing, you need to unplug the midi cable from your synth ir garvuino if it's connected, or powerdown the synth. If you want to develop a sketch on garvuino, it can be tiresome to always unplug the midi. You can use a simpler wire <=> midi connected to the J1, with GND to midi GND (pin 2) and the other slot (which is connected to the RXD port) to the midi pin 5, this way it's faster to unplug. Or use ttymidi or hairless midi serial to usb port (see above)

In the case you get a:

```
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x32
```

it can be from different causes. If the midi plug is connected, see advice above. But it can also appear at random occasions. I suspect it's because of some electricity in the atmega8 chip. In this case, you can remove the

TX jumper (above the arduino nano). You only need it connected when playing AY chiptunes (see below). This kind of message can also be seen if you've chosen the wrong port to connect the arduino.

## Flashing for AY music

Just load `avray/avray_sd_sketch/avray_sd_sketch.ino`, put some tunes from `tunes/aym_tunes/1_75MHz/rsf/` into your SD card and power the device!

If you create your own tunes (using Vortex Tracker II for example), you'll need to convert them to RSF using the AVR-AY player: <http://www.avray.ru/avr-ay-player/>

## Sid emulator

load `/sid/sid_midi/sid_midi.ino` for the Sid emulator synth. Change sound mode with the switch. You can change some sound with knobs on your keyboard. The sketch is programmed for my Behringer UMA25S keyboard, knob E09 to E13 (midi control 71 to 75).

There is also a sid player (`/sid/sid_player/`) which can load some SID data and play it from the arduino (at the moment it can only load from memory, which isn't big enough for a complete song).

You can also connect a gyroscope module to the Garvuino and have fun with sounds. See `sid/gyro_header_sid/` for this.

## Flashing for 1-bit music

Only one 1-bit music can be installed into the Arduino at a time. Load a sketch, for example `beeper/arduino_qchan/arduino_qchan.ino` and flash the Arduino. Music should go out of the mini-jack now!

You can compose music using the [beepola tracker](#).

Beepola can use the Tritone, Qchan and Phaser1 engines.

There are python converters in the folder to convert your own music to arduino code. Just make it into beepola and "compile" it to .bin format in

the tools menu (without song engine, "song data only"). Convert the bin to data readable by the arduino with the python script.

Then include your code into the arduino sketch (use `#include "your_song.h"` instead of `#include "music_data.h"` for example.) Use the line that is not commented (a comment on arduino starts with `//`)

For the octode engine, you'll have to edit a .xm file with a tracker like milkytracker. See <http://battleofthebits.org/lyceum/View/Octode> for more informations. There is a xm to arduino converter in the `arduino_octode` folder as well.

Read more on the [1-bit forum thread](#) (search for arduino)

If you have some problem for creating your own music and exporting it to arduino, you can join the 1-bit forum to request for help.

## Flashing for Mozzi synth

There are several examples in the `mozzi/` folder, just try them out!

You can also use most mozzi examples from the original library (see `Examples/Mozzi` in the Arduino IDE).

## Flashing other sketches

The Garvuino should be able to play many other sketches from various projects on internet.

This one can play some MIDI files after converting them:

<https://bitbucket.org/farvardin/playtune-arduino>

Just connect PIN 5 to AY>L and PIN 6 to AY>R (PIN 9 is already connected to audio output) and you'll get a midi player able to play up to 3 voices. You can connect even more voices to the free Digital Output on the Garvuino (for example by adding a pin on PIN 10).

Use the `examples/test_nano/test_nano.in` sketch.

The Auduino is a project for using granular synthesis with 5 potentiometers, you'll have to connect PIN3 from the arduino to the AY>L audio output (see the sketch for more informations).

## Assembly the Garvurino board

If you got the board through a kit, soldering it should be pretty simple as there are only basic components.

If you have no clue about soldering, this guide might be of some use for you: <https://www.makerspaces.com/how-to-solder/>

Solder the resistors and capacitors first for example, then the jumpers, socket, led, switch, crystal and audio jack. In any case, the regular sd card reader should be soldered after all of this.

If you use a micro-sd reader, it could be better to solder first the left socket for the atmega8 chip (by side of the arduino chip), then the other elements (led, resistor, pins), then the micro-sd reader, then the right part of the socket for the atmega8 chip, otherwise it's more difficult to solder the micro-sd reader, yet still possible.

There is a little error on the version 1.09 / 2017-08 of the PCB board, which I've fixed with a wire. On this version, the D6 output on arduino is shared with the LED so it might not always be available on some sketches.

## Bill of Materials

| Name                      | nb | (optional part) |
|---------------------------|----|-----------------|
| Garvuino PCB              | 1  |                 |
| crystal 24 mhz            | 1  |                 |
| arduino nano              | 1  |                 |
| atmega8                   | 1  |                 |
| ceramic capacitor 820 pF  | 2  |                 |
| electrolytic capac. 10 uF | 2  |                 |
| ceramic capacitor 100 nF  | 1  | x               |

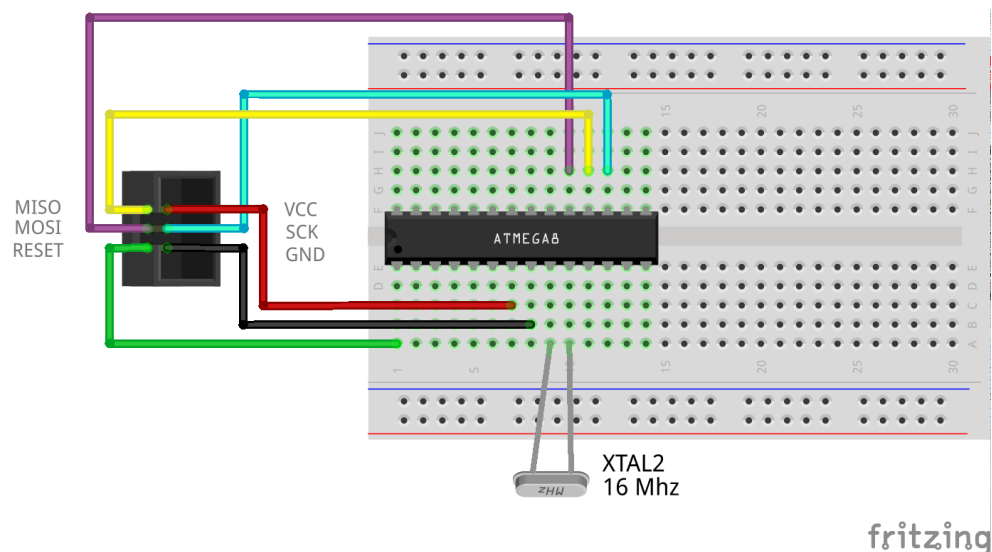
|                                 |   |   |
|---------------------------------|---|---|
| led                             | 1 | x |
| DIN5 connector (midi)           | 1 | x |
| 1 k Resistor                    | 1 | x |
| 10 k Resistor                   | 2 |   |
| micro sd module                 | 1 |   |
| momentary switch (6x6mm type)   | 1 |   |
| audio jack 3.5 mm (TRS)         | 1 |   |
| jumper                          | 6 |   |
| female header socket PCB 2.54mm | 1 |   |
| male pin strips 2.54            | 1 |   |
| female pin strips 2.54          | 1 |   |

## Atmega8 setup and flashing

If you've bought the Garvuino assembled or in a kit, the atmega8 is already flashed for AY emulation. Yet, you might want to install a new revision of the AVRAY program, and use different speed and settings. Here are the instructions:

Get a "USBASP USB ISP Programmer & 10 Pin ISP interface Cable - AVR ATMEL ATmega".

Connect it according to this schematic for the atmega8:



Basically it's:

| SD reader | Atmega       |
|-----------|--------------|
| MISO      | pin 18: PB4  |
| MOSI      | pin 17: PB3  |
| RESET     | pin 01: PC6  |
| VCC       | pin 20: AVCC |
| SCK       | pin 19: PB5  |
| GND       | pin 22: GND  |

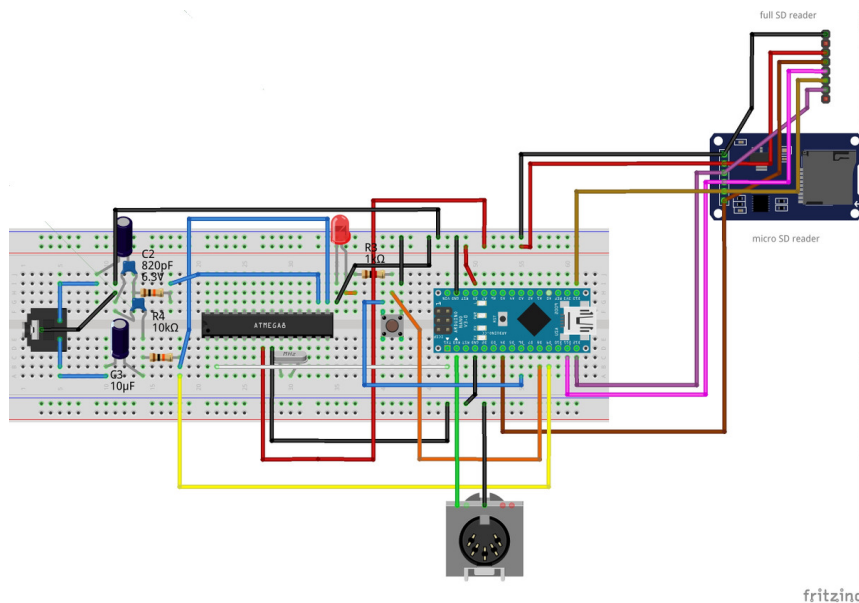
Use a 16 Mhz crystal for example, on pin 09 (PB6) and pin 10 (PB7).

Once it's connected, you can program the atmega8 chip with this command-line:

```
avrdude -p atmega8 -c USBasp -U flash:w:AY_Emul_244_2ch.hex -U
eeprom:w:Conf_standard_24MHz_1_75Mhz.hex -U lfuse:w:0xCE:m -U
hfuse:w:0xCF:m
```

## Breadboard

It can also work without the PCB. Just use a breadboard and a few wires!

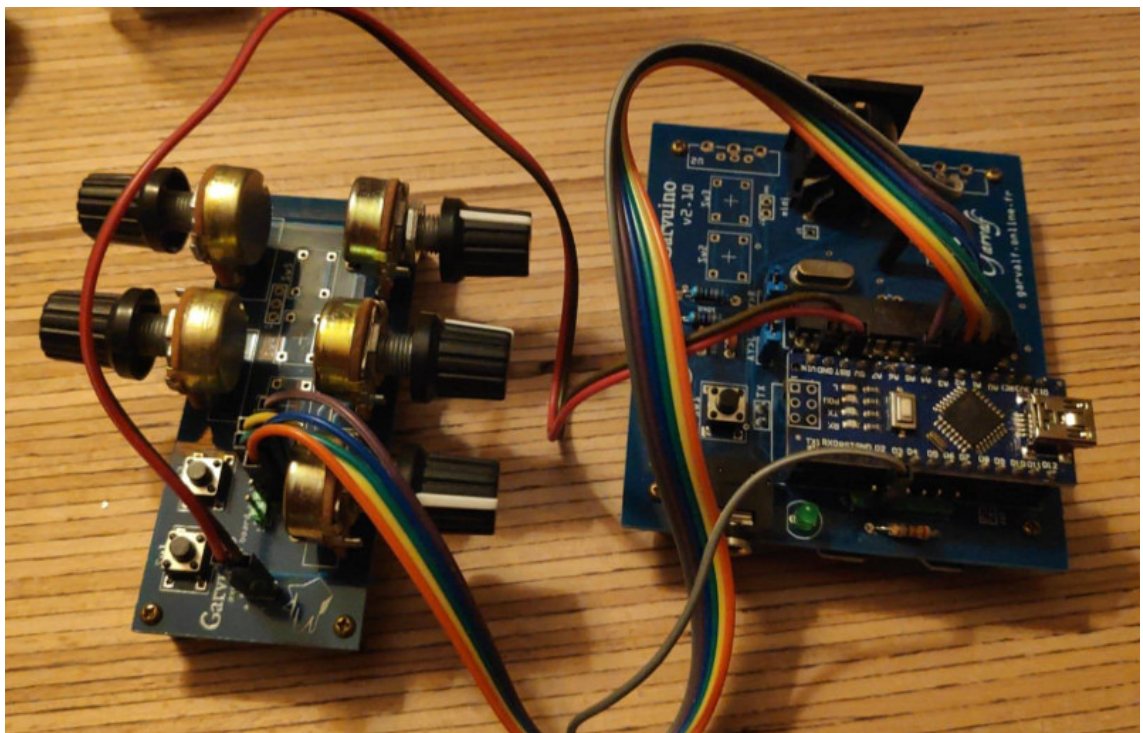


Please refer to the garvino/breadboard folder for the fritzing sources and pdf version.



## Expansion board

The expansion board is a new board for using with the Garvuino or other projects. Solder 5K or 10K potentiometers onto the board. You don't need to solder the switches, they are not used at the moment.



Connect the digital pin D3 to the audio out, under the L or R of the AY>L or AY>R pin.

And also connect the 2 right top pins and the 3 bottom pins to the A0 to A4 pins on the arduino, connect the + to 5V and the - to ground, and you'll be able to control the Auduino and Auduino midi sketches found there: [https://github.com/farvardin/garvuino/tree/master/arduino\\_sketches/others](https://github.com/farvardin/garvuino/tree/master/arduino_sketches/others)