Use a USB keyboard for text entry

Use a standard USB keyboard to enter some text and have it spoken live on VAX (soon on Kaiwa and Emy).

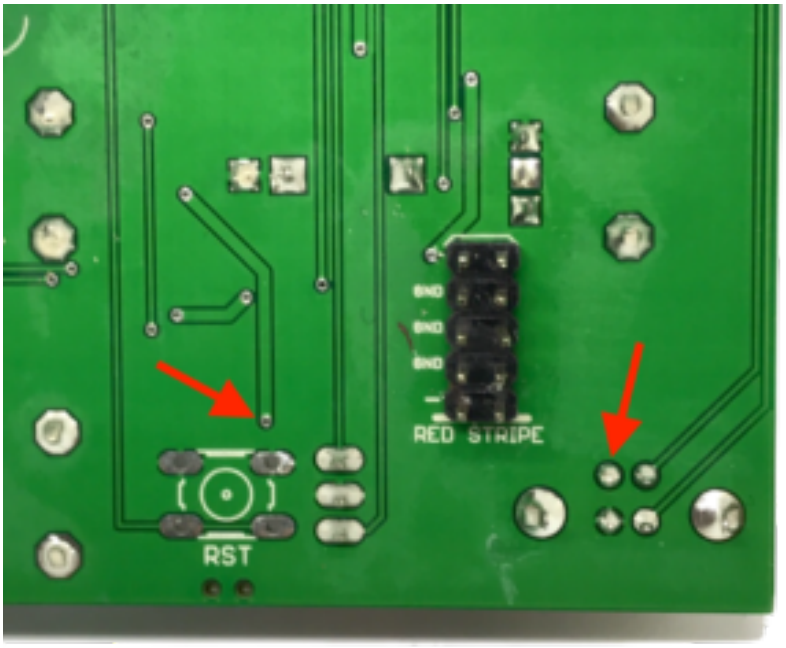
The keyboard is connected to a USB gender changer (provided with the last edition of Emy) which also powers the keyboard.

The speech can be triggered by the Gate signal, pressing the rotary or by pressing **Enter** on the keyboard.
Basic editing is possible with the **backspace** character while the **Escape** key is erasing the current text.

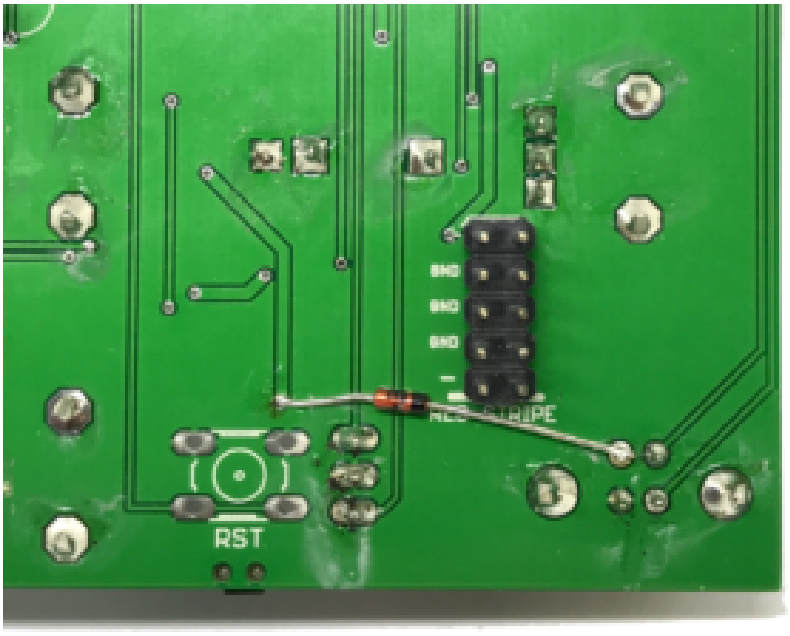
There is also a gate out signal on the Busy/Aux jack that stays up while the keyboard is pressed (it is not needed for the speech but I thought it could be fun to use it)

The newer version of Emy’s PCB 1.0c allow powering the keyboard but If your PCB is 1.0b you will need to apply the following hack.

- Add a Schottky diode (like the BAT85) between the +5v via end the USB connector like shown here :



- Be careful not to overheat the via while soldering it.



- Do not use this port to power something else than a normal keyboard. The maximum current that can be provided is 300 ma.

Enjoy your new talking keyboard!

This works great with wireless keyboards too and allows triggering the speech from quite a distance.

How to add new vocabulary to Talko, the easy way

Previously, I had described here: how to do compress sound using a venerable Windows 3.1 tool: Qbox Pro. The process what quite long to set up and does not always produce good results.

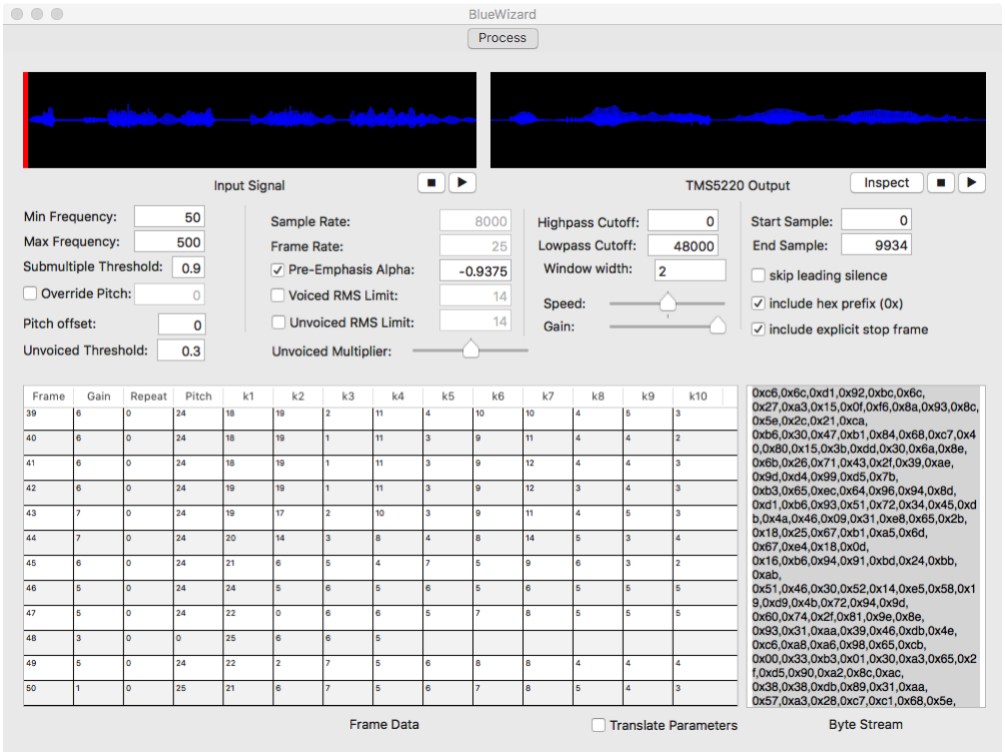
Today I discovered BlueWizard from this post.

BlueWizard runs on Mac and allows to tweak the process in real time to optimize the output! The author has been kind enough to make some small tweaking just for the Arduino and Talko!

Simply open your file (which has to be recorded at 8 kHz with 16-bit depth), click on the 2 tick boxes:
– “include hex prefix (0x)” to allow direct pasting into the Arduino IDE
(if you are using it for Emy’s SD card do not tick this box)

- and “include explicit stop frame” to avoid the library producing gibberish noise a the end of the sound

then copy the resulting the data from the “Byte Stream” windows.



Open the Arduino IDE and paste the data stream into your code before uploading it to Talko.

Let’s make a sound and process it:

```
say -v"alex" "We are charging our battery. And now we are full of energy. We are the robots." -r 100 -o roboter.wave
```

converting to 8 kHz with 16-bit depth using SoX

```
sox roboter.wave -r 8k -b16 roboter.wav
```

and the compressed version made with Talko :

the Arduino code:

```
// Talkie Adafruit library
// Copyright 2011 Peter Knight
// This code is released under GPLv2 license.

#include "talkie.h"

Talkie voice;
```

```
const uint8_t spROBOTS[]
={0xa6,0xd6,0x4a,0x67,0xd8,0xed,0xa8,0x9a,0x35,0x37,0xd5,0x8a,
0x1c,0x72,0xbf,0x84,0xcd,0x2a,0x4a,0xca,0x42,0x50,0x77,0xed,0x
6a,0x25,0xcb,0x1a,0xb3,0xa3,0xa3,0xa4,0x34,0x4b,0xb2,0xe9,0x0c
,0x9b,0xd2,0xc4,0xd9,0xa7,0xca,0x6c,0x49,0x8a,0x26,0x9d,0xaa,0
x28,0x2d,0xc9,0x96,0xad,0xbc,0xaa,0xb6,0x24,0x47,0xb5,0xd0,0x1
c,0xd3,0x92,0x9c,0xcd,0x43,0x7c,0x68,0x4b,0x72,0xd6,0x2c,0xf6,
0xa2,0x25,0xcd,0x9d,0xb3,0x4d,0xc6,0xa4,0xb4,0x4c,0x8a,0x32,0x
19,0x9d,0x92,0xb2,0xc8,0xd2,0x79,0x92,0x89,0xeb,0x14,0x36,0xd3
,0x26,0x04,0xd8,0x79,0x2c,0x01,0xbb,0x5e,0xb6,0x68,0x8e,0xed,0
x2e,0x8b,0xdd,0xea,0x51,0xd3,0xd3,0x35,0x4e,0xcb,0x73,0x74,0xf
3,0xd4,0x38,0xa3,0x4c,0xd9,0xdd,0xc3,0xed,0x96,0x2a,0x76,0xb5,
0x36,0x5d,0x9c,0xaa,0x38,0x4c,0xc7,0x75,0x52,0xaa,0xc3,0x14,0x
5b,0xb3,0x51,0xa1,0x0e,0xdd,0x64,0x92,0x4f,0x86,0x3a,0x54,0x89
,0x31,0x99,0x98,0xea,0x58,0x38,0x27,0x64,0x42,0xa8,0x73,0x81,0
x49,0x8b,0x91,0xa6,0x4e,0x19,0x36,0x22,0x4a,0x88,0x3a,0x25,0x9
8,0x0c,0x2b,0x69,0x9a,0xa4,0x49,0xbd,0xb2,0x56,0x69,0x46,0x55,
0x66,0xd5,0xb6,0xa5,0x19,0x95,0xd8,0x2c,0xda,0xb4,0xa6,0x5a,0x
52,0x8f,0xac,0x53,0x9a,0x62,0xd1,0xab,0x2a,0x72,0x68,0xb3,0xc1
,0xd8,0xcc,0xd0,0xae,0x8b,0x8a,0xe2,0xd2,0x4c,0xb9,0x4e,0x1b,0
xed,0x56,0xb7,0x62,0x3a,0x63,0xa4,0xdd,0xd3,0x4a,0xe8,0x8a,0xc
1,0x95,0x52,0x45,0xa9,0x2d,0x16,0x47,0xcb,0x94,0xb4,0xa6,0x65,
0x2a,0x8b,0x92,0xdb,0x9a,0x5a,0x25,0xd2,0x2a,0x6c,0xab,0x6b,0x
93,0x32,0x6f,0xab,0xa3,0xce,0xd5,0x32,0x2d,0xeb,0xb4,0x3a,0x55
,0xcb,0x32,0x7f,0x54,0xea,0xd8,0xac,0x5a,0xed,0x76,0xaa,0x63,0
xb7,0x18,0x93,0xcb,0xa1,0x8e,0x43,0x7d,0x54,0xae,0x84,0x3a,0x0
e,0x8e,0x15,0x99,0x93,0xea,0x58,0x38,0x43,0xbd,0x32,0xab,0x6d,
0xe2,0x0c,0xd7,0x21,0xa4,0x8a,0x5a,0xc2,0xc3,0x62,0x95,0xac,0x
24,0x71,0xf7,0xa8,0x55,0x8a,0xe4,0xb9,0xdd,0xcb,0x48,0x2b,0x72
,0xb0,0x68,0xcb,0x30,0x23,0xcf,0x49,0xa3,0x3c,0x43,0xb7,0x34,0
x17,0xad,0xb4,0x32,0xd3,0xd2,0xd2,0xa4,0xdd,0xdb,0x6c,0x4b,0x7
3,0x93,0xb6,0x28,0xa9,0x25,0xcd,0x4d,0xda,0x23,0xa5,0x96,0x3c,
0x16,0x1b,0x8b,0xb0,0x9a,0xca,0x50,0xbc,0x2c,0xdd,0x6a,0xa8,0x
93,0xd7,0x96,0xb2,0xb0,0xae,0x2d,0x51,0x92,0xd3,0x2b,0x87,0x3e
,0x17,0x0e,0x6d,0x19,0xeb,0x86,0x34,0xb1,0x35,0xb4,0xaa,0x19,0
xe2,0x80,0xd1,0xf4,0xaa,0x66,0x88,0x1d,0x46,0x23,0xa3,0xba,0x2
1,0x35,0x1c,0x89,0x8e,0x1a,0x86,0x92,0xa1,0x3c,0xb2,0x6c,0x18,
0xb3,0x61,0xad,0xb1,0xd8,0x61,0x2c,0x86,0xa4,0x26,0xa2,0x84,0x
a9,0x28,0xd2,0x9a,0x8c,0x1d,0xa6,0x66,0x88,0x63,0x2a,0x4e,0x98
,0xba,0x61,0xf2,0x99,0xd0,0xee,0x1f,0x0a,0xc5,0x2b,0xad,0x88,0
x7f,0x6a,0x50,0x6f,0xb5,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x0
```

0,0x00,0x00,0x00,0x00,0xee,0xad,0x49,0x32,0x2b,0xa3,0xb6,0xa2,
0x46,0x75,0xcf,0x2c,0xdd,0x8a,0xe2,0x34,0x33,0xd2,0x4c,0x2b,0x
8a,0x93,0xb2,0x0e,0x29,0xad,0x2c,0x5e,0xd2,0xab,0x24,0x97,0x32
,0x3b,0x49,0xef,0xa4,0x6c,0x4a,0x69,0x74,0xad,0x8c,0x89,0xa9,0
x84,0xd1,0xd3,0x66,0x47,0xa6,0x92,0x5a,0xcf,0x5a,0x1c,0x9b,0x8
a,0x5b,0x5b,0x0b,0x75,0xa2,0x2a,0x18,0xe3,0x3c,0x4c,0x89,0xa8,
0x40,0x8e,0x75,0x37,0x39,0xa9,0xf4,0x5a,0x2b,0xad,0xe8,0xb4,0x
3c,0x59,0x2b,0xcf,0xa0,0xd2,0xb2,0x9c,0x24,0x3d,0x53,0xca,0x48
,0x4a,0xd5,0xb4,0x48,0x39,0x2d,0x2e,0x55,0x4b,0x33,0xe4,0xb4,0
x38,0x37,0x2d,0x29,0x97,0xdd,0x92,0xdc,0x2d,0xd5,0x5b,0xca,0x4
8,0x52,0x8d,0x0c,0xab,0xc8,0x2d,0x4d,0xa5,0x22,0x34,0xcb,0x94,
0x2c,0xb6,0x76,0xb7,0x1c,0x53,0xf2,0xd0,0x2a,0x4d,0xb3,0x4a,0x
2a,0x43,0xef,0x10,0xcb,0xc9,0xae,0xb4,0x7d,0x9c,0xd5,0xeb,0xa8
,0x5a,0x95,0x4d,0x62,0xab,0x6d,0x1a,0x95,0x37,0xc8,0xad,0x74,0
x68,0x54,0x4a,0x37,0xcf,0x2a,0xa1,0x75,0xd6,0x2a,0xa3,0xc2,0xa
4,0x2e,0x69,0xf6,0xae,0x2a,0x5d,0xba,0x62,0x48,0x33,0x6b,0x72,
0x69,0x5b,0x24,0xb1,0xaa,0x29,0xad,0xed,0x99,0xd9,0xc2,0xa7,0x
b4,0xa6,0x7a,0x55,0x77,0x9d,0xd3,0x9a,0xe2,0xdd,0xc3,0x65,0x76
,0xab,0x73,0xc8,0x28,0xa7,0x45,0xad,0x4e,0x39,0xb3,0x9c,0x96,0
x94,0x2a,0xf5,0x88,0x50,0x9d,0xe2,0xaa,0xf8,0xb0,0x8a,0xbc,0x9
1,0xa9,0xe2,0xa1,0x68,0xd2,0xc6,0xae,0xce,0x2e,0xc3,0xcd,0x2c,
0xbb,0x75,0xf8,0x8e,0x0c,0xb3,0xad,0x80,0xe5,0xba,0x14,0x30,0x
43,0x87,0x02,0x56,0xac,0x48,0xe7,0x70,0x61,0x61,0x96,0x34,0xc5
,0xc9,0x54,0x44,0xa8,0xe3,0x54,0xf8,0x56,0xaa,0xee,0x4d,0x52,0
xe9,0xcb,0x18,0x5b,0x2c,0x4a,0x65,0x08,0x93,0x2c,0xf6,0x28,0x9
4,0xbe,0x8e,0x52,0xe4,0xa4,0x50,0xfa,0xb0,0x41,0x16,0x97,0x5c,
0x69,0xd2,0x39,0x5b,0x4e,0x09,0x65,0xf0,0x93,0x42,0x35,0x25,0x
95,0x21,0xb4,0x99,0x76,0xb9,0x52,0x46,0x57,0x16,0xb2,0x61,0x5b
,0x99,0x5c,0x9a,0x59,0x47,0x29,0x65,0x92,0x63,0x62,0x55,0xb6,0
x54,0x99,0x8d,0xa9,0xc7,0x98,0x54,0x47,0x55,0xe6,0x16,0x93,0x5
d,0x9b,0x65,0xa8,0xa9,0x57,0x71,0x6d,0xd2,0xec,0x69,0x56,0xba,
0x35,0xc5,0xa9,0x85,0xe4,0xe8,0xd1,0x14,0x6f,0x9e,0x56,0xa3,0x
46,0x53,0xbd,0x79,0x6a,0x97,0x2e,0x75,0x4e,0x12,0x19,0x2d,0x26
,0xd4,0x29,0x70,0xf2,0x98,0xe5,0x50,0x25,0xcb,0xc5,0x6b,0xa1,0
x53,0x9d,0x03,0xb5,0x96,0x06,0x0d,0x75,0x4e,0xd8,0x56,0x56,0xd
4,0x35,0xb9,0x60,0x45,0xe8,0x50,0xd7,0xe6,0x0a,0x99,0xa5,0x47,
0x5c,0x57,0x2a,0x78,0x94,0xce,0x12,0x43,0x0a,0x50,0x5e,0x51,0x
29,0x8c,0xa3,0xa6,0x10,0x6a,0xb2,0x04,0xcc,0xd1,0x5e,0x86,0x99
,0x94,0x59,0xbd,0x53,0x1a,0xab,0x61,0xf1,0xea,0x44,0x69,0xcc,0
x86,0x2c,0x36,0x23,0xa7,0x31,0x5b,0xd6,0xb8,0x88,0x55,0xc6,0xe
6,0x58,0x64,0x72,0x4e,0x19,0xbb,0x67,0x91,0xca,0xc5,0x65,0x1c,

```
0x81,0x45,0x22,0xe6,0xa4,0xb1,0x7b,0x66,0xa9,0x9c,0x92,0xa6,0x
ae,0x59,0xac,0xb3,0x76,0x9a,0xba,0x66,0xf6,0x89,0xc8,0x61,0xee
,0x96,0xd8,0x27,0x42,0x9b,0xb9,0x19,0xe4,0x18,0xb7,0xa2,0xe6,0
xa6,0x90,0x7d,0x23,0x8a,0x58,0xaa,0x25,0x96,0xd2,0xa8,0x00,0x0
0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xea,0x65,
0x21,0x3a,0xc9,0xc3,0xa8,0x9e,0xf9,0xe8,0x06,0x8f,0x6c,0x3a,0x
51,0xcc,0x55,0x7a,0x74,0x68,0xad,0xd7,0x4c,0xaf,0x22,0xa9,0x2b
,0x06,0xb5,0x3b,0xca,0x86,0xae,0x1a,0xd4,0xea,0x2a,0x1c,0xba,0
xac,0x51,0xab,0xa3,0x4c,0xe9,0xb2,0x12,0xe9,0x72,0xdb,0xad,0xa
b,0x8e,0xa5,0x23,0xa2,0xb4,0xb6,0x04,0x89,0xd4,0x0a,0x9b,0xda,
0x54,0xa5,0x3c,0xd2,0x58,0x6a,0xe2,0xd0,0x72,0xf5,0xb2,0xa9,0x
49,0x5d,0xab,0x45,0xc7,0x86,0x26,0x0f,0xee,0x61,0x1b,0x1a,0x9a
,0x34,0xa4,0x8b,0x6d,0x68,0x68,0xd2,0xe0,0x4e,0xf6,0xa1,0xa9,0
xc9,0x8d,0xda,0xd5,0xcb,0xa6,0x36,0x25,0x8e,0xf0,0x8c,0x5c,0xd
a,0xec,0x55,0xd2,0x33,0xb6,0x69,0x83,0x51,0xa9,0x28,0xcb,0xa9,
0x09,0x26,0xad,0xbc,0x8b,0xa4,0x36,0x3a,0xf7,0xb2,0x2a,0x13,0x
ba,0x10,0xb9,0x3c,0xa3,0x96,0xe9,0x7c,0xa6,0x8a,0xb4,0x46,0xa6
,0x33,0x49,0x2d,0x4a,0x5c,0xbb,0xce,0x14,0x33,0x2d,0x63,0x1b,0
x1a,0xdf,0xc4,0x22,0xcc,0x6d,0xa8,0xfc,0xa4,0x8c,0xb0,0xd4,0xa
5,0x0c,0x39,0x5d,0xc2,0x6a,0xb5,0x3c,0xa4,0x76,0x35,0x9f,0x53,
0xb2,0x10,0x26,0x54,0xfd,0x61,0xc9,0x82,0xdb,0x54,0xd5,0x87,0x
29,0xf7,0x71,0x43,0xc4,0x6f,0x86,0xc2,0x87,0x75,0x72,0x9f,0xa8
,0x1a,0x95,0xdb,0xc4,0x7c,0x32,0x19,0x69,0x2a,0x13,0xf5,0x38,0
xa9,0x0b,0xa1,0xd2,0x45,0xa7,0xa4,0x3e,0x94,0x56,0x77,0x1f,0x1
3,0xc6,0xd0,0x4b,0xc3,0xb3,0x4c,0x1a,0x63,0x1d,0x09,0x8f,0x32,
0x65,0x4a,0xa9,0x23,0x38,0xcb,0xa6,0x39,0xa5,0xf6,0x10,0xaf,0x
92,0xe6,0xd4,0x32,0x4c,0xdb,0x4a,0x59,0x52,0xce,0x48,0xe9,0xb0
,0x69,0xc9,0xd9,0xcb,0x74,0xa4,0xb9,0x25,0x45,0x2d,0xf3,0x14,0
xa3,0xd6,0x62,0xc3,0x5d,0xdd,0x34,0x59,0xba,0x74,0xcf,0x08,0x4
b,0x08,0x58,0xa9,0xc3,0x30,0xc3,0x65,0x84,0x7b,0x1c,0x0,0x0,0x
0,0xf0,0xff,0xff};

void setup() {
}
void loop() {

     voice.say(spROBOTS);
}
```

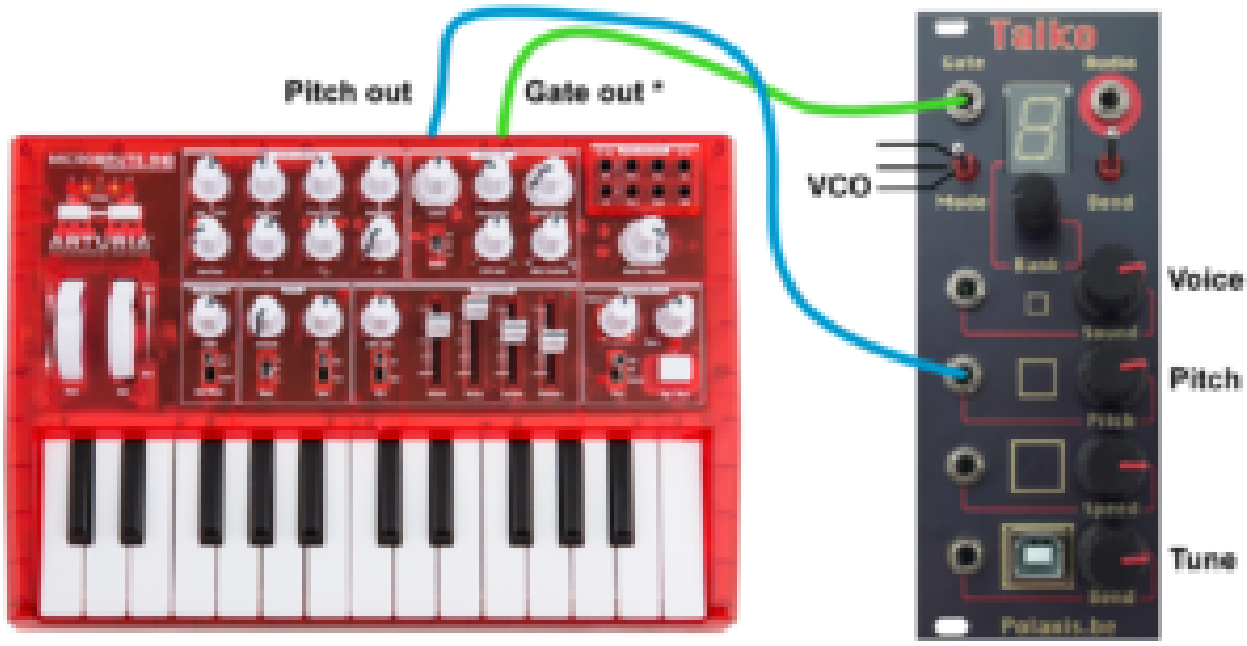
New Talko firmware : VCO mode can play semi-tones

The rev 2 of Talko's firmware can now play notes over 2 octaves.

It may seem obvious to have a 1V/octave CV to note in a VCO mode but this was quite challenging to achieve. (I may explain the process in another post as it could be also applied for another of my speech synths)

Connections

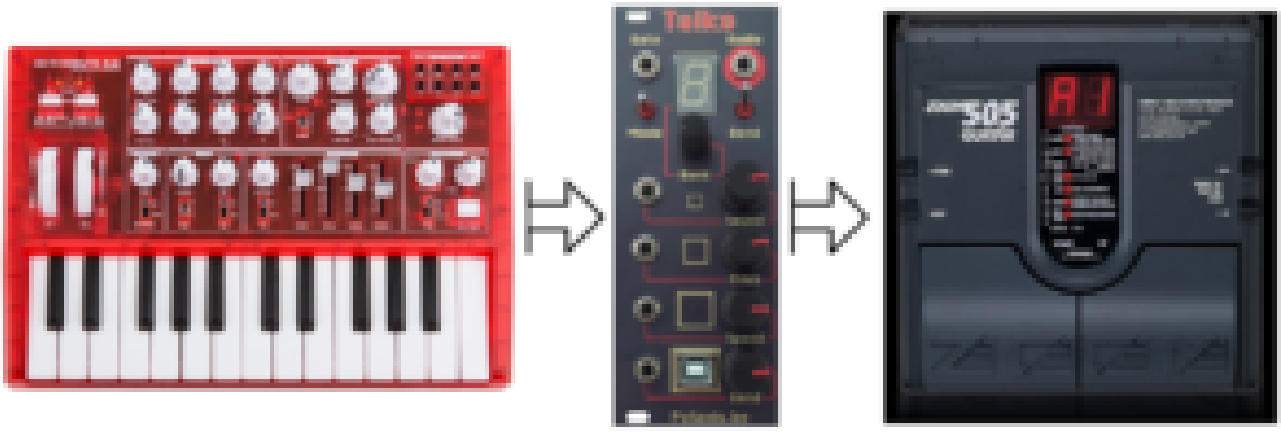
Quite easy: just feed a pitch signal into the Pitch entry. Tune the synth up and down with the Bend pot and trigger the note with the Gate.
Set it to Bank 16 and play with the Sound pot to choose among the 26 vowels sounds.



(*) Please note that the Microbrute’s gate out can’t trigger Talko directly (it’s a Microbrute know impedance issue): use a gate buffer in between.

Sounds

The VCO sound very much like an organ and it’s quite fun to pass the audio through a guitar pedal like the Zoom 505:



#Talko new firmware : the VCO mode can now play semi-tones ! #arturia keyboard driving #talko pitch cv in . I apologize for my poor keyboard skill

(^_^)

Une publication partagée par Jean-Luc Deladrière (@polaxis) le 24 Févr. 2017 à 14h04 PST

The Microbrute can also send notes via its sequencer: this allows instant fun for the poor keyboard player like me!

#talko in VCO mode with the #arturia #microbrute as sequencer

Une publication partagée par Jean-Luc Deladrière (@polaxis) le 24 Févr. 2017 à 13h26 PST

Firmware

The code is available here: Talko 1.2 rev2
Right click to save it as a .hex file and use Easy uploader to install it into Talko.

User Manual

The manual has been updated to reflect this revision : http://www.polaxis.be/wp-content/uploads/2016/05/Talko-Manua

l-1_2.pdf

---

3 new sounds banks for Talko

Talko’s code has been updated with 3 new sounds banks :

Bank 15 : French Vowels Male (5)

View this post on Instagram

Bank 16 : English voiced allophones (72)

Bank 17 : VCO friendly voiced allophones (25)

This bank is a selection of allophones from bank 16 that produces nice looping sounds in VCO mode (great for mouth drumming for example)

The new code is available here
and can be uploaded with Easy uploader as described on the downloads page.

The updated manual is to be found here.

Have fun !

Talko as Ginkosynthese Grains

Both Talko 1.1 & 1.2 share some hardware with the Ginkosynthese Grains.

The Grains is also an Arduino module (based on the famous Audino code from Peter Knight) with the 3 first analogues port being used to manipulate the sound

Grains Talko 1.1 Talko 1.2

Talko can also do this and the Grains code need just to be tweaked a bit to get the output on pin 3 (very easy to do)

I tested a few examples found here: http://www.ginkosynthese.com/product/grains/

I uploaded some example on the Github and also posted their compiled firmware here so can be uploaded directly using EasyUploader :

---

fresh.hex

TALKO 1.1 control
Sound : sample offset
Pitch : loop length
Speed : pitch

TALKO 1.2 control

Pitch : sample offset
Speed : loop length
Bend : pitch

View this post on Instagram

A post shared by Jean-Luc Deladrière (@polaxis)

[hrtl-cereals-V2]

TALKO 1.1 control
Sound : sample start
Pitch : grain size
Speed : pitch

TALKO 1.2 control
Pitch : sample start
Speed : grain size
Bend : pitch

[jgb-patternrain-v2]

TALKO 1.1 control
Gate : clock in
Sound : select pattern
Pitch : select bank for patterns
Speed : stop/reset and then pattern rotate (to be able to make it fit better to other parts of your music)

TALKO 1.2 control
Gate : clock in
Pitch : select pattern
Speed : select bank for patterns
Bend : stop/reset and then pattern rotate (to be able to make it fit better to other parts of your music)

View this post on Instagram

A post shared by Jean-Luc Deladrière (@polaxis)

[jgb-RZ1-drums]

TALKO 1.1 control
Sound : Pitch CV 0-5 V
Pitch : Play on / off. Set it to max for normal function
Speed : SAMPLE_SELECT between two wavetables 0-5 V

TALKO 1.2 control

Pitch : Pitch CV 0-5 V
Speed : Play on / off. Set it to max for normal function
Bend : SAMPLE_SELECT between two wavetables 0-5 V

If you would like to adapt other Grains code, just let me know so I can post there here too.

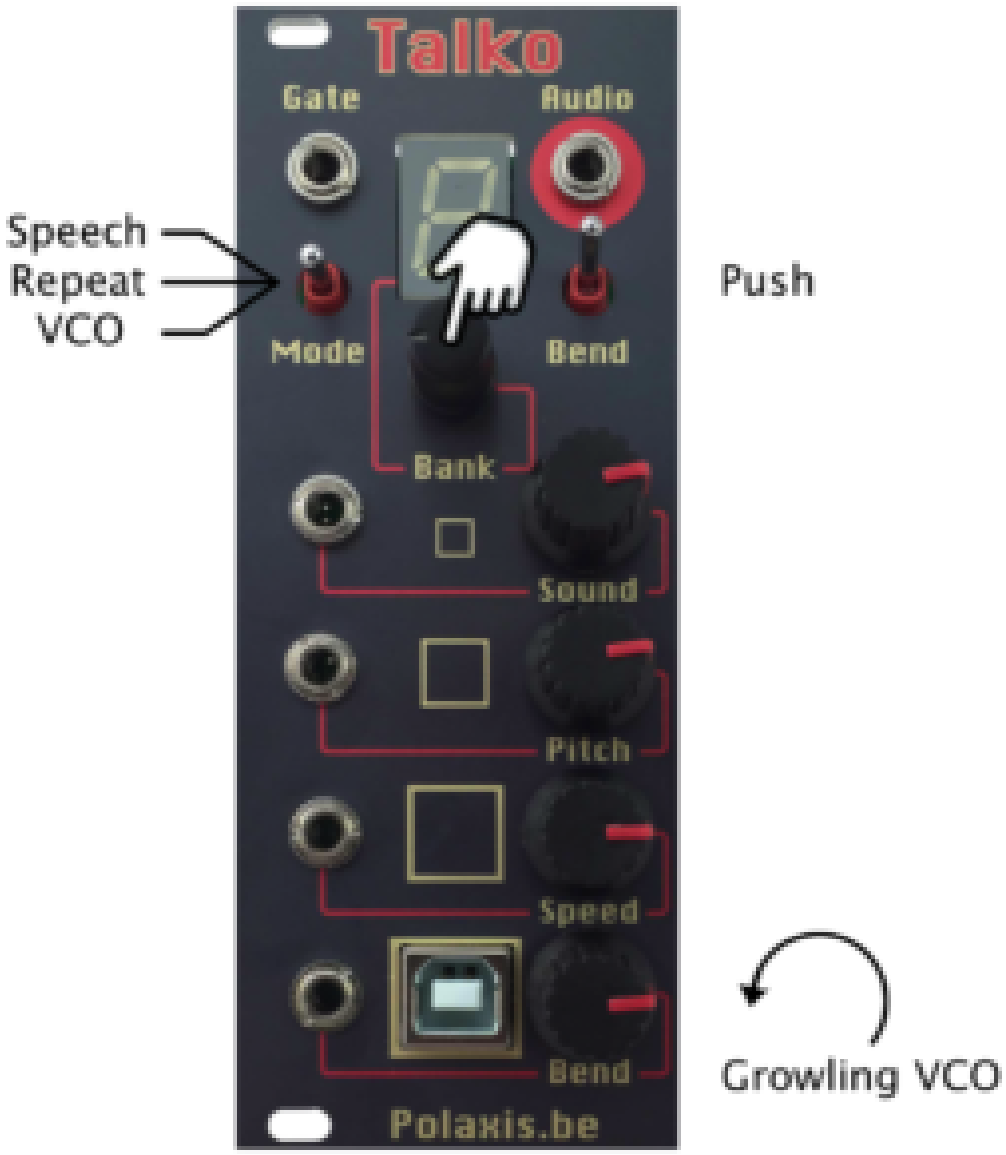
Talko 1.2 now available

Talko 1.2 are now in stock, both as kits and as assembled modules.

What's new in version 1.2

- Rotary encoder for smoother Bank change
- Encoder's button can be pressed to simulate the gate signal going HIGH and make the module speak.
- Mode selection via a 3 positions switch : Speech – Repeat – VCO

- Growling mode in VCO (turn the Bend pot fully CCW)



My first patches

Beyond having it to speak and bending it in (the obvious) Speech mode, I would recommend testing the Repeat mode by feeding some rhythmic pattern into the Gate entry and particularly playing with the Gate length.

Another fun one is to set it to Bank 0, VCO mode, turn the Bend pot fully CCW for growling mode and then play with the Sound, Pitch and Speed pots.

Try also to press the rotary button while the Sound entry is being sequenced in VCO mode to hold notes on and manually

alter the sequence.
Do you have a nice patch to recommend ?
Thanks in advance for sharing

Talko 1.2 is coming soon

Talko is an open source Arduino based LPC speech synthesizer. It’s firmware can be updated via the onboard USB port, using the standard Arduino IDE.

In **Speech** mode, the speech starts with a gate signal and complete before waiting for a new gate signal. The speech has the priority.

In **Repeat** mode, the speech starts and stops with the gate signal going high or low. The gate has the priority and the speech repeats while the gate is high. This mode is very useful to create crazy rhythms.

In **VCO** mode, the LPC engine loops while the gate is high, producing steady notes.

The VCO mode can also produce sounds using white noise instead of tones, making strange throat like sounds.

The sound synthesis can be is driven via CV signals or knobs to choose sounds and alter pitch, speed & bending.

flyer1.2.pdf

LPC encoding for the Arduino’s Talkie library

Adding new sounds or vocabulary for the Talkie library is not straightforward and I needed a checklist to smoothen the process.
Here are the main steps :

Recording audio with Audacity

- The recording has to be made at 8 kHz with 16-bit depth
- Export to Wav signed 16-bit PCM(note that you can also use Audacity to re-sample the audio to 8000 kHz via the [track/re-sample] menu)

Converting sounds using SoX

Alternatively, you can also convert various audio format to 16 bits 8 kHz with SoX, using the following command:

```
sox audiodump.wav -r8000 -b16 audio-8k.wav
```

Coding with QBOX pro

QboxPro was made to code sounds for the venerable TMS5220 chip that Talkie library is emulating.

It runs only on an ancient system like Window XP or older

[fac_icon icon="exclamation-triangle" color="#dd3333"]Note: it seems that QBOX doesn't like when the audio starts immediately. In that case the compressed audio is totally inaudible, so adding a little pause before the sound starts helps a lot

Installation

Get the software here : ftp://ftp.whtech.com/pc%20utilities/qboxpro.zip

Don't forget to install QBOX at the root of the disk : c:\QBOX and to move the QBOXPRO.ini file to c:\WINDOWS

Coding

The process of coding has already been described in detail

here : http://furrtek.free.fr/index.php?a=speakandspell&ss=9&i=2

The process goes like this:

- Create a new project using the following project parameters : Byte / 8 Khz / 5220 coding table
- Goto Project and add the audio file
- Choose process using : medium bit rate and pressing OK
- Edit concatenation : insert concatenation after by adding a name; then insert phrase and press ok
- Format it by choosing the first line in the format menu : LPC 10V, 4UV

Arduino code

Recuperate the .bin file that Qboxpro has generated This file contains the LPC stream and need to be translated into C++

I use this small Python script to convert the .bin

```
import binascii
fname="SOUND.BIN"
f = open(fname, "rb")
#print "{",
code ="const uint8_t sp"+ fname[:-4]+"[] PROGMEM ={"
try:
    byte = f.read(1)
    while byte != "":
        # Do stuff with byte.
        byte = f.read(1)
        code = code + "0x"+(binascii.hexlify(byte)) +","
        #print "0x"+(binascii.hexlify(byte))+" ,",

finally:
    f.close()
    code = code[:-4]
    code = code +"};"
    print code
    print
```

```
    print("voice.say(sp"+fname[:-4]+");")
```

Simply paste the script’s outputs at their respective places into the Arduino code and upload

Here is an example I generated with the Mac’s say command (note the 0.3-second silent before the speech starts)[edit : it’s fine with 0.1 too]

```
say -v"Yannick" "[[slnc 300]] Wir sind die Roboter" -r 100 -o roboter.wave
```

then I converted to the appropriate format using SoX

```
sox roboter.wave -r 8k -b16 roboter.wav
```

After the QBOXpro coding and the Python converting, I copied these lines into the Arduino IDE

```
// copy this part before the setup() section

const uint8_t spROBOTER [] PROGMEM ={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x80,0x4a,0x6a,0xca,0xac,0x2a,0x26,0x29,0x79,0x50,0xf1,0xae,0x88,0xae,0xb4,0x2a,0x54,0x33,0x5a,0x87,0x36,0x4b,0x65,0x35,0x8b,0xd3,0xba,0xe6,0x43,0x45,0x44,0x49,0xe9,0x86,0x32,0x61,0x92,0x24,0xad,0x1c,0x36,0x04,0x45,0xe3,0xb6,0x62,0x58,0x25,0xb1,0x58,0x72,0x8a,0x69,0x14,0x25,0x72,0xf1,0x29,0x36,0x17,0x92,0xaa,0x59,0xab,0x58,0x9c,0x28,0xab,0x26,0x9d,0x62,0x71,0x26,0xaf,0x9e,0x74,0xf2,0xc5,0x85,0xac,0x6a,0xf6,0xc9,0x97,0x56,0x94,0xaa,0xc5,0x27,0x9b,0xde,0x50,0x33,0x67,0x9f,0x6c,0x18,0x43,0xcb,0x5c,0x7c,0xb2,0xa2,0x44,0xba,0x6a,0xf2,0xc9,0xaa,0x32,0xce,0xac,0x39,0x27,0x2b,0xc6,0x24,0x63,0xaa,0x9c,0xbc,0xc9,0x60,0x97,0x9c,0x7d,0x8a,0xca,0x52,0x5c,0x62,0x76,0xab,0x0a,0x4b,0x51,0xf3,0xc9,0xa5,0xae,0x2c,0x45,0x34,0x1a,0xa7,0xaa,0x53,0x57,0x25,0x8f,0x5b,0xca,0x6e,0x42,0x44,0xd4,0x4e,0xc9,0x3b,0x77,0x31,0xf1,0xd4,0xa5,0xae,0x32,0xc9,0xd4,0x6b,0x9f,0x21,0xab,0xc0,0xcc,0x28,0x7d,0x8a,0xa2,0x1d,0x33,0x62,0xf4,0x29,0x9b,0x0c,0xcc,0xf0,0xd1,0xa7,0xec,0x2a,0xd1,0xdd,0x46,0x9f,0xb2,0xeb,0xc0,0x30,0x1b,0x73,0xca,0xae,0x83,0x42,0x6c,0xcc,0xa8,0xab,0x12,0x09,0xec,0xaa,0xad,0xac,0x92,0x64,0xdc,0x67,0x97,0xaa,0x28,0xe8,0
```

```
xf0,0x72,0x93,0xaa,0x2c,0x61,0x22,0x53,0x75,0x69,0xa2,0xc4,0x6
9,0x0b,0xd7,0xa1,0xb1,0x92,0x37,0xad,0x1c,0xb9,0xc6,0x73,0xcd,
0x2c,0x4f,0xd2,0x9a,0x11,0x5c,0x29,0x42,0x6c,0xa9,0x9b,0x22,0x
76,0xaf,0xca,0xa3,0x1a,0x9c,0x28,0xaa,0x27,0xaf,0x6a,0x49,0x26
,0xcd,0x9e,0x73,0xea,0x65,0x04,0x39,0x6b,0xf6,0xa9,0xa7,0x57,0
xe4,0xa8,0xd9,0xa7,0x1a,0x41,0x48,0x23,0x97,0x9c,0xba,0x1b,0x6
6,0x8b,0x5a,0x7c,0x9a,0x2a,0xdc,0xc4,0x62,0x4e,0x00,0xb4,0x4e,
0x71,0x40,0xf0,0x49,0x0a,0x48,0x6a,0x4d,0x00,0x96,0x8d,0x0b,0x
40,0xf8,0x15,0x05,0x88,0x69,0xb4,0xaa,0x90,0x36,0x45,0x6d,0xf2
,0x29,0x43,0xee,0x12,0xf1,0xc9,0xa7,0x0c,0x75,0x8c,0x3d,0x6b,0
x9d,0x32,0xe4,0x75,0xcc,0x8e,0x7c,0x8a,0x18,0x36,0x31,0xb3,0xf
2,0xca,0x83,0x9e,0x26,0x8d,0x99,0xa9,0xf0,0x66,0x0b,0xd3,0x43,
0xab,0x82,0x85,0x4a,0x61,0x8b,0x45,0x32,0x66,0xd3,0x5d,0x35,0x
4e,0xaa,0x07,0x77,0xd7,0x0a,0x2b,0xad,0x9f,0x54,0xc3,0x33,0xe2
,0x84,0x3a,0xf1,0x2a,0x55,0x89,0xb5,0xd2,0xe0,0xd6,0xd9,0x62,0
xe2,0xc9,0x43,0x59,0x63,0xcb,0x51,0x27,0x8f,0x79,0x52,0xc4,0x6
f,0x9d,0x3c,0xa6,0x4a,0xd5,0x98,0x72,0xb2,0x14,0xca,0x4c,0x7d,
0xd1,0x49,0x53,0x28,0x75,0xc9,0xda,0x2d,0x0d,0x31,0xd4,0x25,0x
ab,0x10,0x80,0x26,0x4b,0x02,0x48,0x6f,0xc1,0xf2,0x6a,0xcd,0x45
,0x54,0x6a,0x4b,0xba,0xf5,0xaa,0xd2,0xc4,0xa3,0x99,0x32,0x35,0
x3c,0xea,0x8c,0xb1,0xab,0x64,0xb7,0xa8,0x7c,0xb6,0xa2,0x4d,0xc
a,0xab,0xf4,0xe9,0x8a,0x56,0x2b,0xcd,0xc1,0xa7,0x2f,0xc1,0xd4,
0xad,0xcb,0x9c,0xbe,0x04,0x73,0xd7,0xac,0x7c,0xc6,0x92,0xdd,0x
5d,0x72,0xce,0x19,0x4b,0x4c,0x0b,0xf5,0x39,0xa7,0x2f,0x21,0x3c
,0x34,0xe6,0x9c,0xa5,0xf8,0x88,0x94,0x5c,0x72,0x8e,0xea,0xc2,0
x4b,0x6d,0xc9,0x59,0xaa,0x4b,0x2f,0xd5,0x26,0x67,0xad,0x2e,0xb
d,0x54,0xd7,0x9e,0xb3,0xba,0xf2,0x52,0x5d,0x73,0xba,0x6a,0xca,
0x4a,0x75,0xcd,0x59,0xaa,0x6b,0x2b,0x91,0x35,0xa9,0xca,0x69,0x
28,0xdc,0x67,0x91,0x2a,0xe4,0xd4,0xd2,0xae,0x88,0xaa,0xe0,0xc7
,0xdc,0xb2,0x01,0xaa,0x82,0xc8,0x2c,0x33,0xdb,0x00,0x00,0x00,0
x00,0x00,0x00,0x00,0x00,0x00,0x0f};

// copy this code in the main loop
voice.say(spROBOTER);
```

and here is how it sounds:

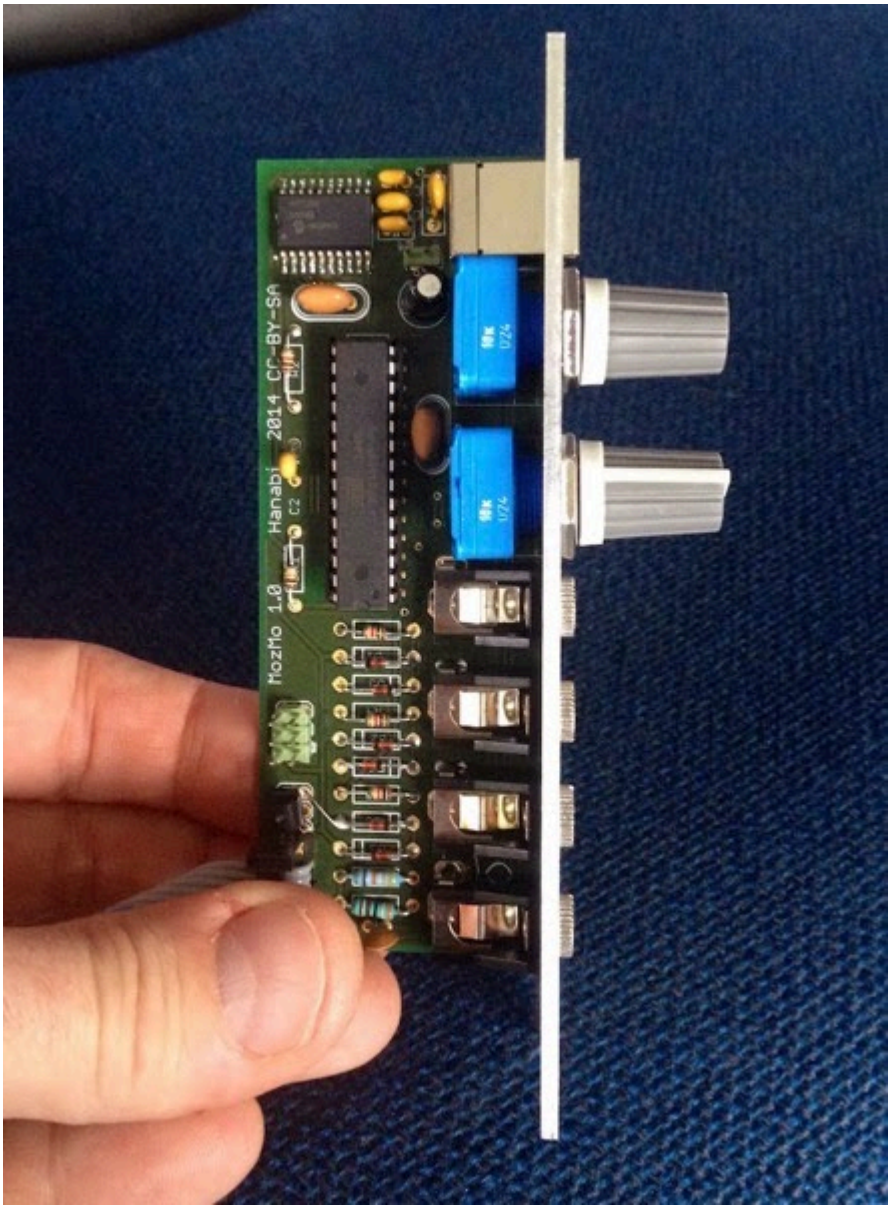
MozMo : the brilliant Arduino Mozzi synth in an Eurorack hardware.

The idea

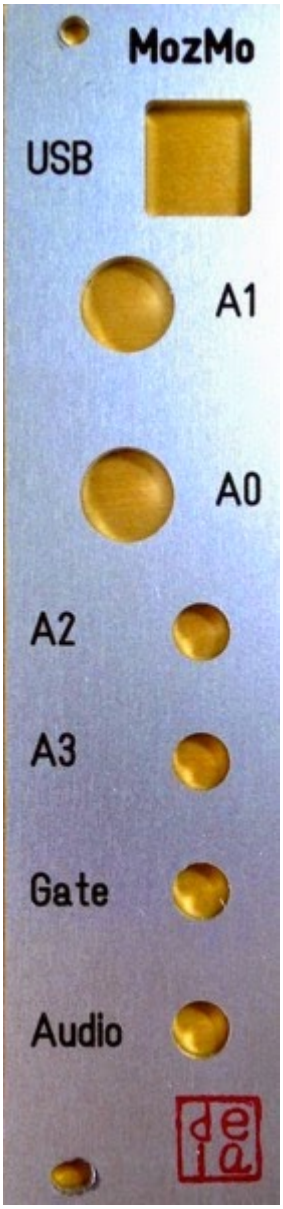
Mozzi by Tim Barras is an outstanding library that allow the Arduino to produce complex and exiting sounds with almost no additional hardware.
(In fact MozMo uses the hifi mode that requires … 2 resistors and a cap !) The idea was to build a dirty cheap modular synth exploiting the vast potential of this library.

Features

- Arduino compatible with Usb connection
- Powered via Doepfer 5v bus or via Usb (via jumper)
- Uses Mozzi Hifi mode
- 2 potentiometers
- 2 CV entries
- 1 gate entry
- 1 audio out
- Depth : 40 mm
- Size : 6 hp



Panel

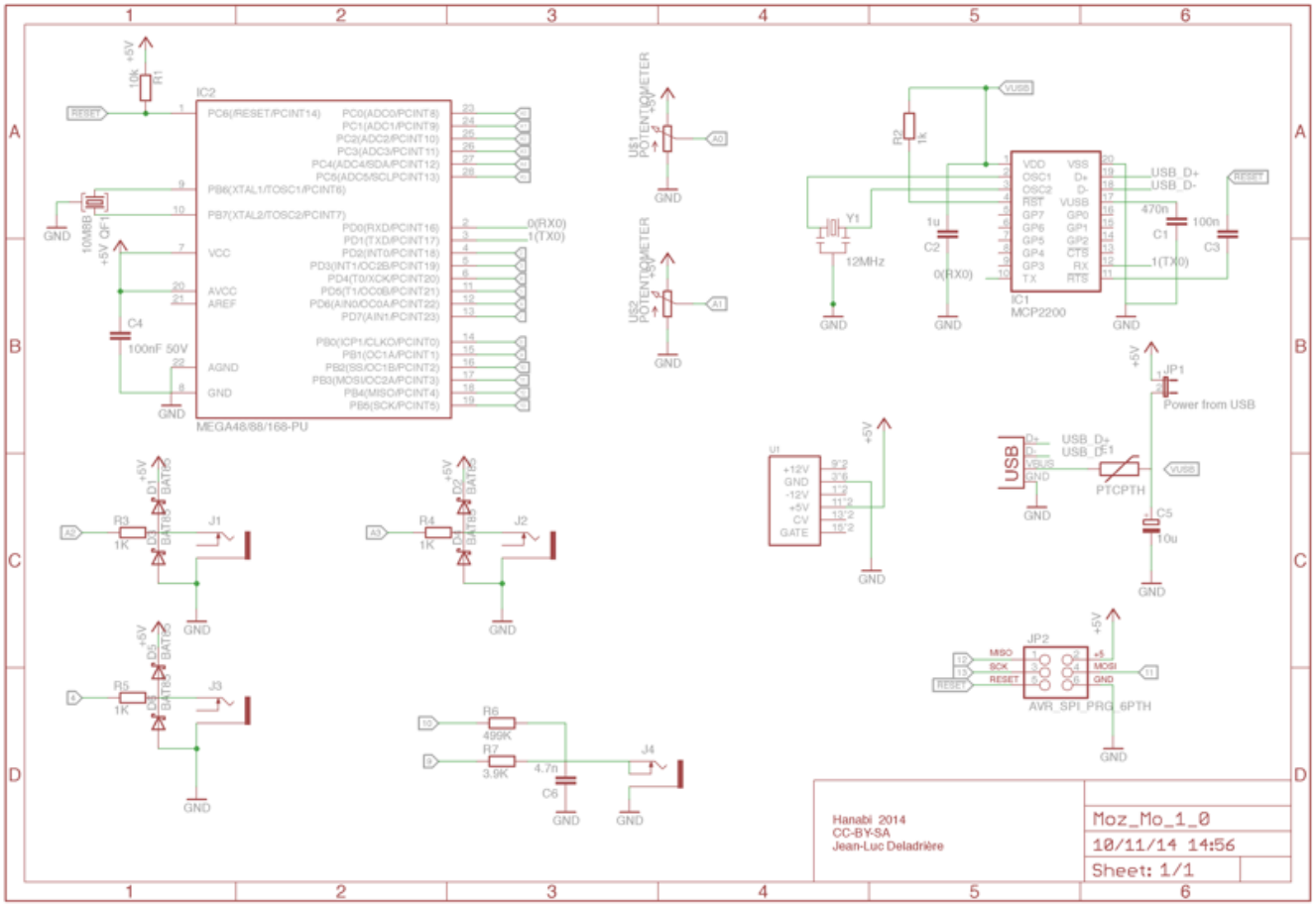


Soundcloud

Here is one example

Check my [Soundcloud]() for more demos

Schematic



Examples sketches

I am building a collection of [Mozzi sketches] adapted for this module.
(wait for the page to load as there a few Soundclouds files embedded)
You can help me to build this collection by sending me your best sketches.

Github

All the hardware files and the Arduino sketches are kept under my [Github repository]

Assembly

If you plan to build one, have a look a this [https://www.evernote.com/l/AAUrhQ524SpHc4VzwlT0_xva_Gbh5O]

W4n_8

Shipping

I keep a few pieces of each components and I can ship :

- pcb
- panel
- programmed Atmega328p
- full kit
- assembled module

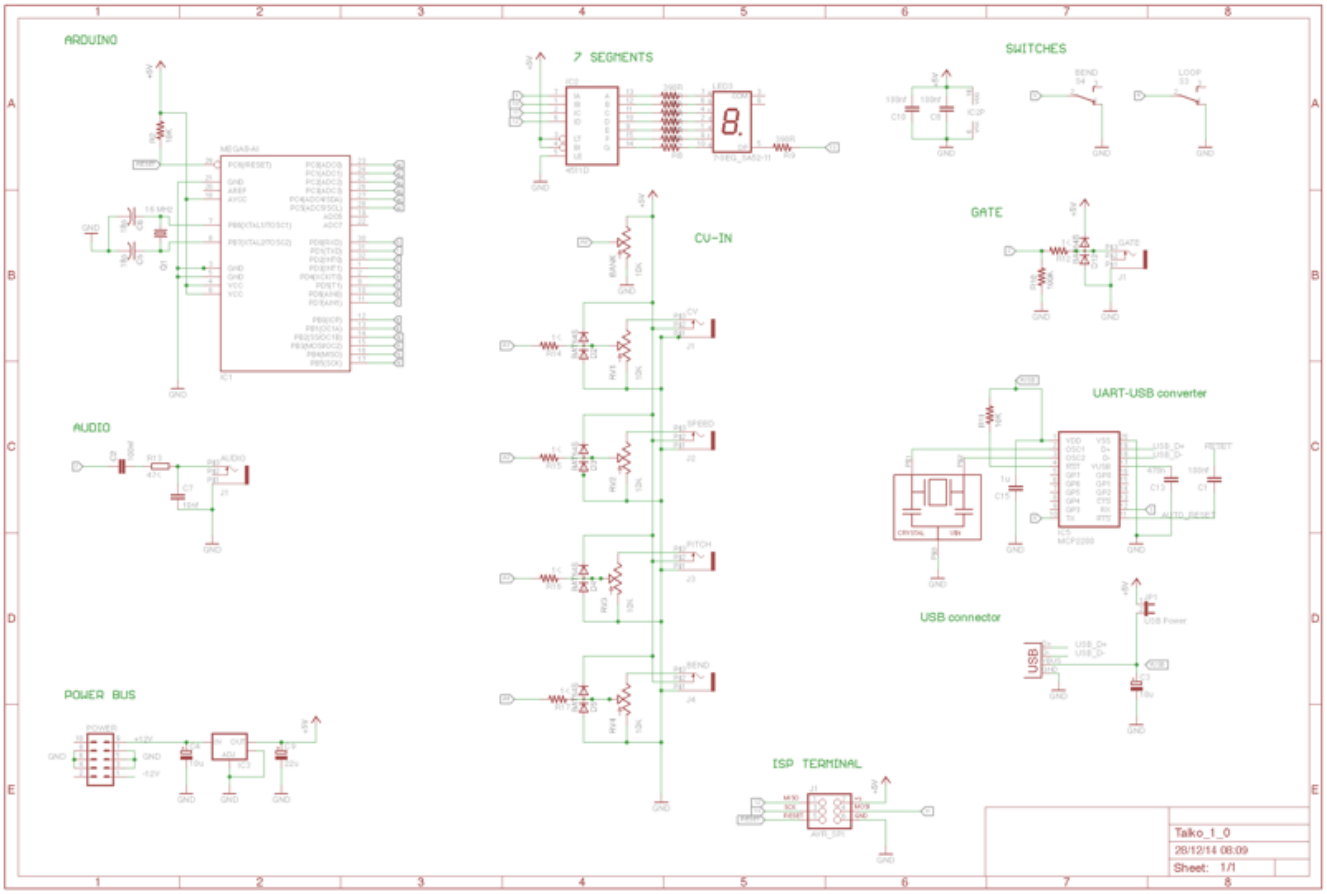
Next

- re label panel entries A2 A3 and pots A0 A1
- install onboard 5V regulator

Talkie Eurorack Module – Part 2 : Schematic & Pcb

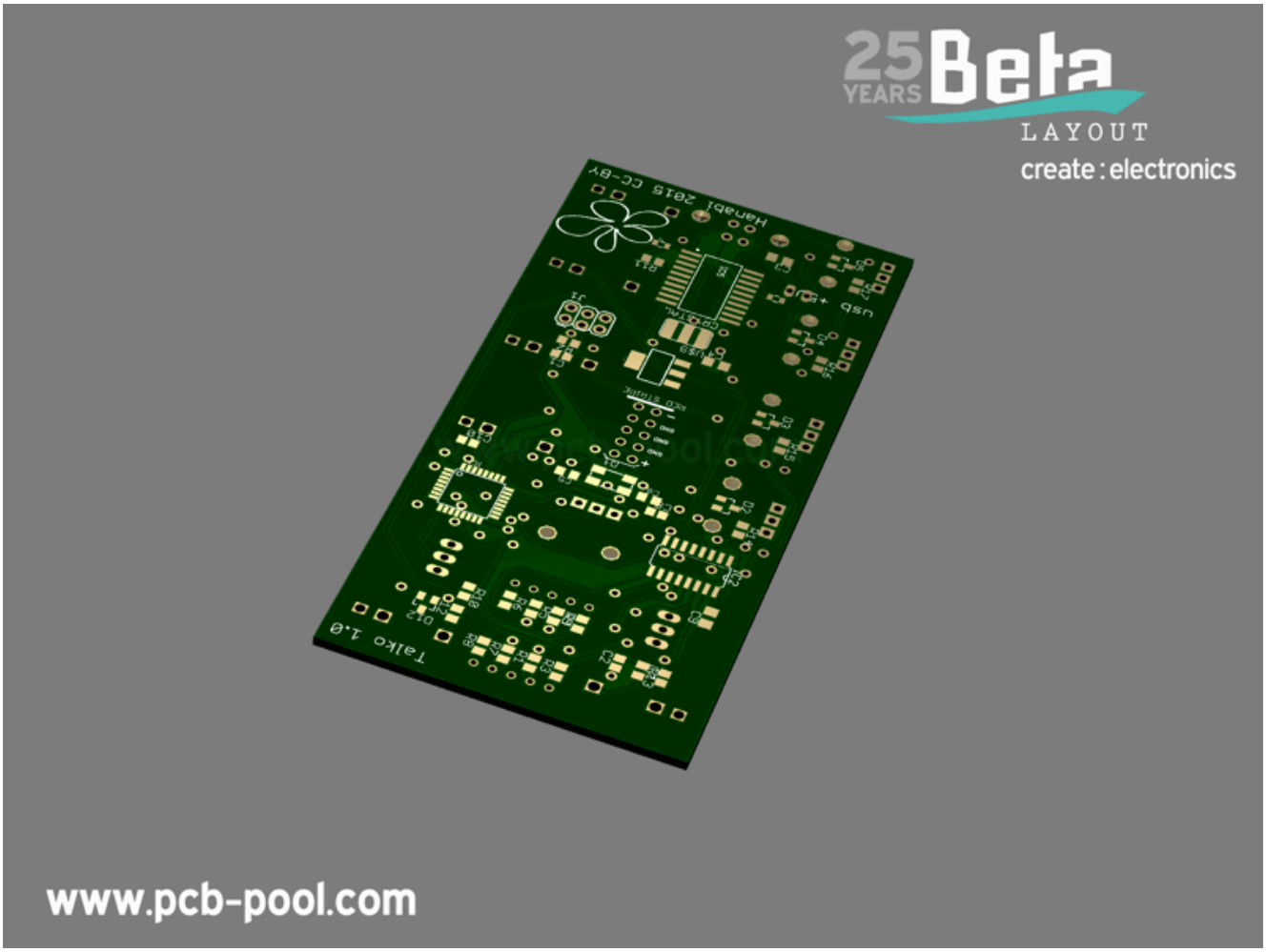
Schematic

Here is the schematic. Nothing really special : A simple RC filter I have used before with the Talkie library and a few pots to fiddle with the various functions. All the entries (cv and gate) are now protected with diodes to allow connections with modulars synths modules using higher voltages. A simple 7 segment to show the current playing mode. I plan to use the dot as the clock led



PCB

I am ordering 8 Pcb from Beta Layout.
Here is the preview I got by uploading the file to their web site



Software

Added female voice used in the [talking clock]

New sound demo

Here is the setup : a clock with variable pulse width is triggering the sound and stepping a sequencer feeding cv to the module. A bit of reverb is sometimes added just for fun

More demo on my [Soundcloud]

Github

You can find all the files (hardware & software) on my [Github]

Next

- To share the Mouser cart
- Module assembly
- Eurorack Panel design