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,0
x5c,0x7c,0xb2,0xa2,0x44,0xba,0x6a,0xf2,0xc9,0xaa,0x32,0xce,0xa
c,0x39,0x27,0x2b,0xc6,0x24,0x63,0xaa,0x9c,0xbc,0xc9,0x60,0x97,
0x9c,0x7d,0x8a,0xca,0x52,0x5c,0x62,0x76,0xab,0x0a,0x4b,0x51,0x
f3,0xc9,0xa5,0xae,0x2c,0x45,0x34,0x1a,0xa7,0xaa,0x53,0x57,0x25
,0x8f,0x5b,0xca,0x6e,0x42,0x44,0xd4,0x4e,0xc9,0x3b,0x77,0x31,0
xf1,0xd4,0xa5,0xae,0x32,0xc9,0xd4,0x6b,0x9f,0x21,0xab,0xc0,0xc
c,0x28,0x7d,0x8a,0xa2,0x1d,0x33,0x62,0xf4,0x29,0x9b,0x0c,0xcc,
0xf0,0xd1,0xa7,0xec,0x2a,0xd1,0xdd,0x46,0x9f,0xb2,0xeb,0xc0,0x
30,0x1b,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};

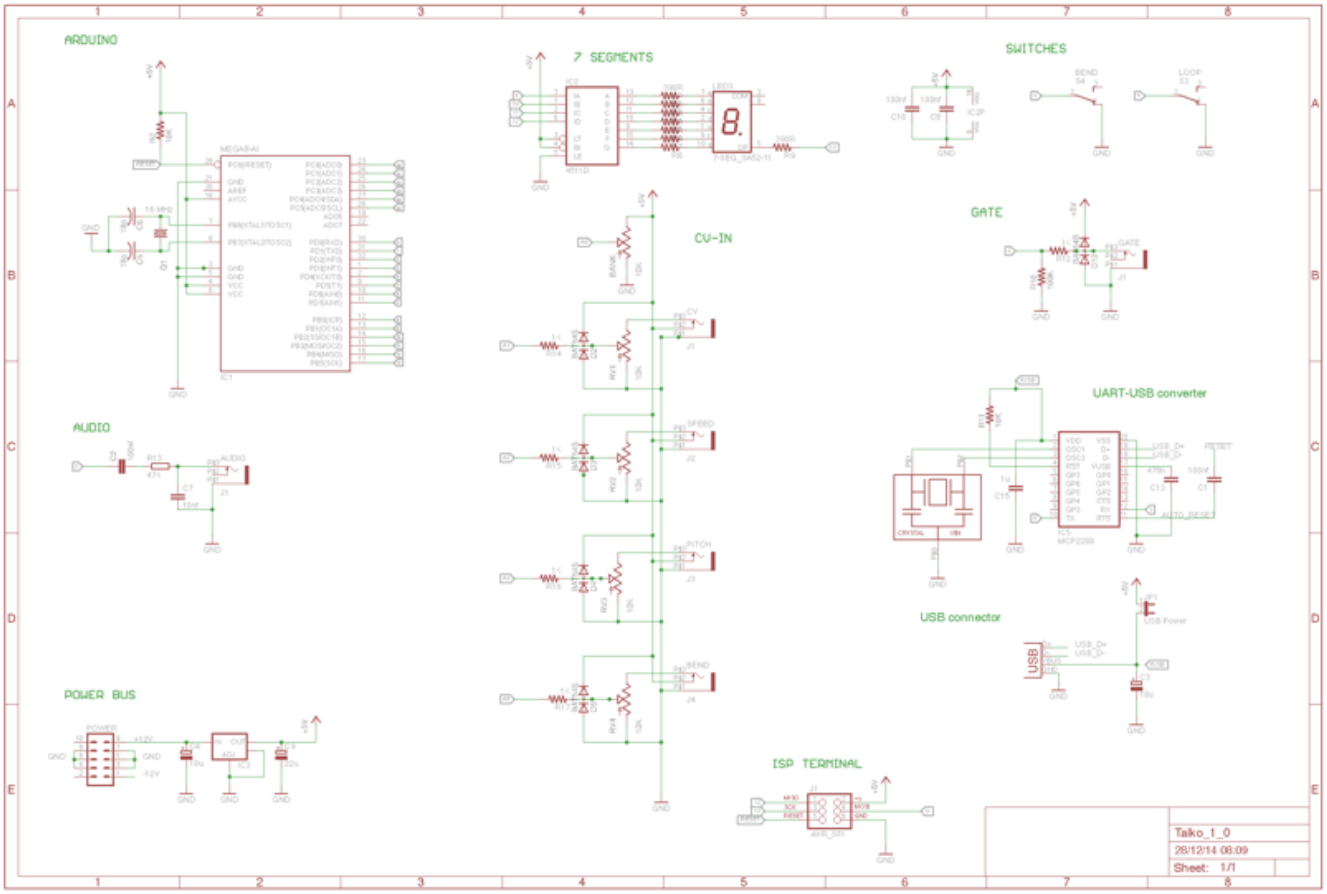
// copy this code in the main loop
voice.say(spROBOTER);
```

and here is how it sounds:

Talkie Eurorack Module – Part 2 : Schematic & Pcb

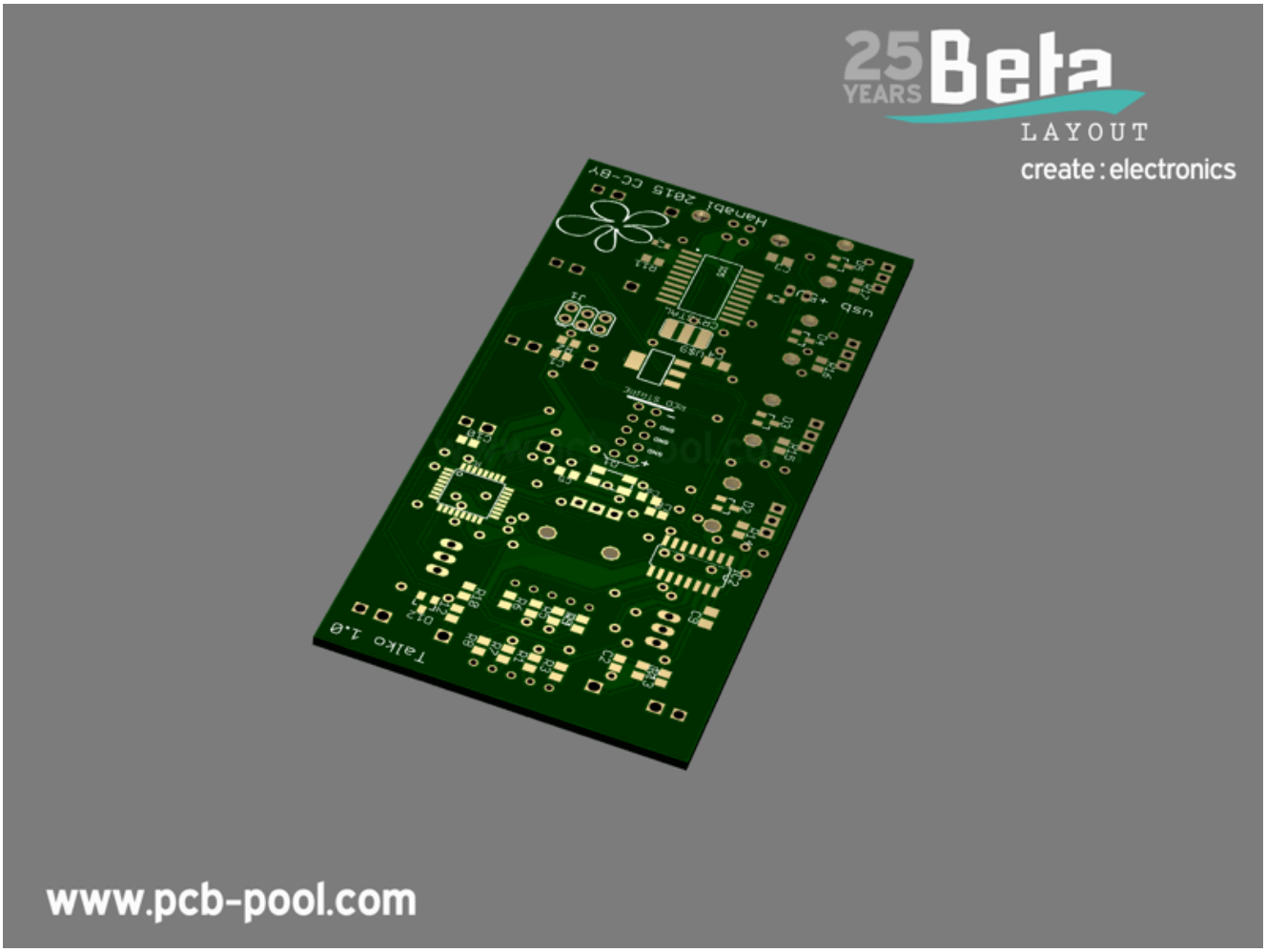
Schematic

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

SSI 263 text to speech in Python via Nanpy on the Arduino

The almighty SSI-263 speech synthesizer chip

As you can read on the Wikipedia page, the SSI-263 is the other name for the Votrax SC-02.
It is becoming quite difficult to find, much harder then it's little brother the Votrax SC-01 but I finally got one by buying a Mockingboard B on Ebay.
The SSI-263 is TTL compatible and need only very few component to be added. It is very easy to connect it to an Arduino but is much more complex than the Votrax-SC01 to program.

Nanpy

As I planned to make many software iteration and process the text to speech routine from Python, I decided to use Nanpy on the Arduino to « relay » the instructions to the chip. I guess I could also use Firmata to do the job as I am using really basics stuff like writing bytes on various pins.

Schematics

The SSI-263 datasheet is quite easy to find and so the programming guide via some Google search but the best one I found is the Votrax SC-02 version that has an extra page at the end with a nice and very simple schematics to get started with .

As I plan to build a shield for the Arduino, I started to draw this in Eagle too :

You can download the Eagle version here
The chip needs a 1-2 MHz clock signal. I am using a Attiny 45 to do the job because all the timer pins are already in use on the Arduino. I have described how to use the Attiny45 to do this in a previous post

Books

The best reference to get started is the article from Steve Ciarca : "Build a Third-Generation Phonetic Speech Synthesizer ».
A complete version is available on Google books. *Ciarcia, Steve, "Build a Third-Generation Phonetic Speech Synthesizer," Byte, March, 1984, p 28. (SSI-263)*
There is also a reference to the SSI-263 and to many other speaking chip in "*Chip Talk: Projects in Speech Synthesis*" but the chapter mainly refers to Steve's article.
The book is still available in some second hand book shop (I got mine via Amazon). The information provided is minimal but it's fun to ready about all these veterans speaking chips.
Prochnow, Dave, Chip Talk: Projects in Speech Synthesis, Tab Books, Blue Ridge Summit, PA: 1987. ISBN is 0-8306-1912-7 (hard cover) and 0-8306-2812-6 (paperback).
Have also a look here : http://www.redcedar.com/sc01.htm. The page is regularly updated and makes an excellent starting

point.

Code

Here is the Main code

The text-to speech is performed via a lookup in the CMU Dictionary (The dictionary can be install with these instructions
: http://stackoverflow.com/questions/11911028/python-arpabet-phonetic-transcription)

It return an arpabet version of the text which is then translated into the Votrax allophone table.
(thank to this modified Arpabet to unicode script)
The allophone chain is then sent to the chip
The code is very basic and doesn't use the SSI263 registers to their full potential : the voice generated is very robotic … but I like it a lot

Souncloud

Listen to the chip saying the Issac Assimov's 3 laws of robotics with 2 different voice settings :

Todo

Next thing I will try is to add prosody or even make it sing
On the hardware side I plan to add a RC filter with an amplifier and add a midi in to the circuit .. so stay tuned.