

retroSpeak Construction and User Guide

SP0256-AL speech synthesiser for Raspberry Pi

retroSpeak is a simple hardware speech synthesizer add-on using the General Instruments SP0256-AL2 IC. It produces a very robotic voice using fragments of language called allophones.

The General Instruments SP0256-AL2 is a chip that was widely available in 1980s designed to generate speech. It was sold by Maplin and Tandy in the UK, and Radio Shack in the US. It was also used in a number of DIY projects in computer and electronics magazines and also available in add-on speech synthesisers for home computers of the time, like the BBC Micro, ZX Spectrum or Vic 20. It was soon superseded by software speech synthesisers that were much more flexible and were capable of producing better quality speech than the SP0256 chip. The SP0256 has been out of production for many years.

This project is for a small HAT-like PCB that fits on Raspberry Pi computers to interface the SP0256-AL2 chip using SPI. Also there is simple Python code to generate speech using allophones, and a rudimentary text-to-speech system.

Why use the SP0256?

Because it's there. It is a very retro lo-fi robotic speech synthesizer. There's also something funny about interfacing a chip that's over 30 years old it with a computer that is more than capable of synthesising speech and audio of a much higher quality. It's a bit of a nostalgia-fest as I remember playing with one of these on a BBC Micro at school, and remember projects using the chip in the Maplin Magazine.. It's also an exercise in creating a HAT style add-on board using free tools.

If you require better quality, more flexible speech synthesis, then look to established and higher quality systems like eSpeak, Festival or Google. A good overview of speech synthesis is on the Raspberry Pi is at eLinux:

[http://elinux.org/RPi_Text_to_Speech_\(Speech_Synthesis\)](http://elinux.org/RPi_Text_to_Speech_(Speech_Synthesis)) They also have the advantage of not requiring any extra hardware other than a loudspeaker.

Getting a SP0256-AL2

Having not been manufactured for decades, the SP0256-AL2 is quite rare – but it is still possible to buy it from a few places.

In the UK I got a couple from <http://sciencestore.co.uk/acatalog/Electronics.html> (look at the bottom of the page) for £9.50 each plus postage. You may also find

them listed on eBay for around £8 (and some risk of getting a duff one), or indeed it may be lurking in your bits box.

The retroSpeak circuit

The circuit generally follows the one in the Archer/RadioShack datasheet – especially on the output side. There is a simple filter feeding to an old-school LM386 audio amplifier to buffer and drive a speaker. A stereo 3.5mm socket is used for headphone or speaker, but there is a header for a small external speaker. The chip reset section is simplified, under control of the Raspberry Pi.

retroSpeak uses a programmable oscillator that allows the SP0256 to run at different speeds under control of the Raspberry Pi. This is an LTC6903 oscillator IC which is controlled by SPI. The SP0256 seems to have a fairly wide tolerance of clock speeds – from less than 1Mhz up to around 5Mhz. This allows a wide variety of variation in the pitch of the voice. Unfortunately the LTC6903 is only available as a tiny SMD chip – but it is just about possible to carefully solder it by hand.

An MCP23S17 16 way port expander controls the speech chip which like the LTC6903 uses the SPI port – it is easily programmable with the WiringPi library in C and Python. This means that all the other GPIO pins on the Raspberry Pi are available for your own projects.

The speech chip only needs 10 of the available 16 pins on the MCP23S17 – so the remaining 6 pins are brought out to an optional pin header.

Another pin header is made available for a commonly available mini real-time clock (RTC) module (e.g: <http://thepihut.com/products/mini-rtc-module-for-raspberry-pi>) – which will allow a Pi to be made into a talking clock without being connected to the internet. Other I2C devices could be plugged in here, too.

The retroSpeak PCB is designed for Raspberry Pi A+ or B+, it should work perfectly with an older style Pi by soldering a 26 way header in place of the 40way. None of the extra pins on the larger expansion connector are used. A cable or extended header may be needed, though.

Up to four can be stacked – provision is made for giving each a unique address with jumpers on the PCB. A simple circuit is used to allow the LTC6903 on each board to be programmed individually.

The SP0256-AL2 is programmed simply by writing a 6 bit number (a binary version of 0 to 63 decimal) to pins A1-A6 representing the allophone (or pause) to speak. The ALD (Address Load) pin is pulsed from high to low to start

speaking the allophone. The chip outputs the audio on pin 24, which goes through a low-pass filter and into the LM386 amplifier.

When speaking starts the SBY (standby) signal pin goes low, and returns to high when the chip finishes outputting the allophone. By polling the state of the SBY pin, the Pi can establish when to repeat the process for the next allophone. Words and sentences are simply made up of lots of allophones strung together.

Building retroSpeak

It's pretty simple – all through hole components aside from the LTC6903. This is the only tricky bit.

Parts List

Resistors	Value
R12, R14	10ohm
R2	1K
R1, R5, R6, R13	10K
R10, R11	33K
R3	220K
VR1	10K 9mm RA potentiometer or multi-turn trimpot

Capacitors	
C9	47pF ceramic 2.5mm lead pitch
C6, C7	22nF metal film 5mm lead pitch
C1, C3, C10, C11, C13, C14	100nf (0.1uF) ceramic bypass caps 2.5mm lead pitch
C8	1uF Electrolytic capacitor 2.5mm lead pitch
C2, C12	10uF Electrolytic capacitor 2.5mm lead pitch
C15	100uF Electrolytic capacitor 2.5mm lead pitch

Diode	
D1	BAT42 or similar Schottky diode
D2, D3	1N4148 or similar. Schottky like D1 will also work

Transistor	
Q1	2N7000 MOSFET 3pin TO-92

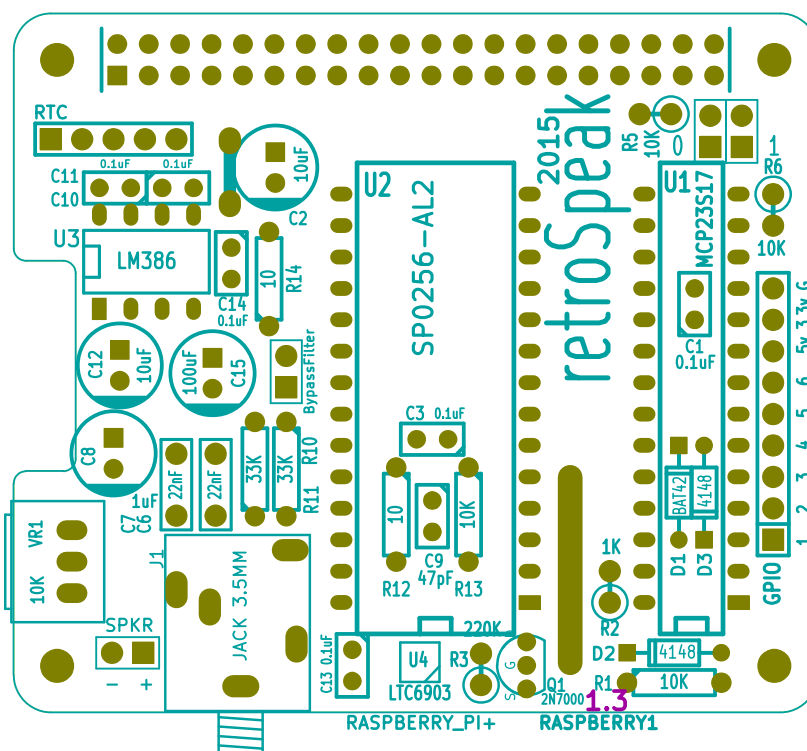
Integrated Circuits	
U1	MCP23S17
U1-Socket	28pin 0.3" socket
U2	SP0256-AL2
U2-socket	28pin 0.6" socket
U3	LM386

U3-socket	8pin 0.3" socket
U4	LTC6903 programmable oscillator MSOP8 SMD

Other components

Female header	40pin 20x2 female header socket
P1	9 way pin header for spare GPIO
P2	5 way pin header for optional RTC
JP1	2 way pin header for Filter Bypass jumper
JP2,JP3	2x2 pin header for MCP23S17 device ID
LS1	2 way pin header for 8ohm speaker
8ohm Loudspeaker	Speaker – wires to 2 pin header connector – optional
J1	3.5mm stereo jack socket
X1 x	3.2768Mhz Crystal

PCB Layout



PCB Layout

Programmable Oscillator LTC6903

Solder the LTC6903 before any other components.

The LTC6903 is a tiny MSOP8 8pin surface mount chip. It is possible to hand solder it – but be careful as it is a bit fiddly.

You will need a very fine pointed soldering iron, fine solder, and probably a magnifying glass and pointed tweezers. A flux syringe is helpful, and solder wick. Make sure the chip is oriented so end with the tiny recessed dot is facing the lower edge of the board.

I found it useful to put some flux on the pads first, and a tiny bit of solder on pads 7 & 8 on the board. These are joined – so offer a larger target area. I placed the chip on the board with the tweezers, and held in position with a little blob of blue-tac putty. Placing the iron over pins 7&8 of the chip fixes it in place.

Then melt a tiny amount of solder over the other pins. The flux should help the solder flow onto the pads and pins. Inspect your work under good light and make sure the pads are not shorted with solder. It is very easy to put too much solder on. More flux and solder wick can be used to remove excess solder.

Check continuity with your multimeter – make sure there are no shorts between the pins of the chip, other than 7&8. Then make sure there's continuity between the chip pins and the Raspberry Pi header on the opposite edge of the PCB.

Raspberry Pi header

Now is probably a good idea to solder the Raspberry Pi header. Use a full 40 way female header for a Raspberry Pi A+ or B+ - or use a 23 way female header if you will be using an older Pi. If the board is to be stacked, then use a socket with long leads.

Resistors

Place and solder all the resistors. Once soldered, clip their leads. Some resistors are stood on end for space reasons.

Diodes

Place and solder the Schottky diode D1 – this is a BAT41, BAT42 or similar small signal Schottky diode. Observe the polarity – the line on the diode must match the marking on the silkscreen. D2 and D3 can be standard 1N4148 or Schottky diodes. These are placed side-by-side under the MCP23S17 socket. If a turned-pin socket is used, there is just enough room. If the socket is a standard type, then solder these diodes to the underside of the PCB. Make sure the line on the diode is closest to the hole with the square solder pad.

Small Ceramic Capacitors

Place and solder the 6 0.1uF (100nF) capacitors, and the 47pF capacitor.

IC Sockets

Place and solder the three IC sockets for U1, U2 and U3. Narrow 28pin, wide 28pin and narrow 8pin sockets. Note the markings and place the notch on the socket appropriately. With the 40way header at the top, the wide and narrow 28pin sockets point downwards, and the 8pin socket to the left. A turned-pin socket should be used for the MCP23S17 if possible.

Film Capacitors

Place and solder the two 22nF box-style film capacitors. These form part of the filter for the voice output.

3.5mm Stereo Jack Socket

Place and solder the 5 pin jack socket. This is a stereo socket wired so the mono output of the speech synthesiser goes to both channels of headphones or external amplifier.

Header Pins

From a pin strip, break off four strips – two with two pins, one with five pins and one with nine pins. These are for the speaker header, filter bypass jumper, RTC header and GPIO pin header. Place and solder these. To ensure they are straight, solder one pin first, then check for straightness. If necessary, reheat the solder of the pin and reposition the header before soldering the remaining pins. A 2x2 header is used for the address jumpers. This can be made from a 2row header or from 4 pins broken from a single row strip.

Volume control

Solder the volume control potentiometer. It is also possible to use a 3pin header here if you want to use a potentiometer away from the board. Alternatively a 10K multi-turn trimmer could be used instead.

Electrolytic Capacitors

Finally solder the four electrolytic capacitors – ensure the negative side of each capacitor lines up with the stripe on the silkscreen. Also, to double check, the long leg (positive) of the capacitor goes into hole in the square pad on the PCB.

11mm Standoffs

Standoffs can be used to securely hold the PCB onto a model A+ or B+ Raspberry Pi. These use M2.5 screws and are 11mm tall. Attach them to the holes on the edge opposite the 40 way header.

If you are using the board with a case like Pimoroni's Coupé then make sure the retroSpeak PCB can't short against the HDMI socket or audio socket. Use short spacers or electrical tape.

First Test

Inspect your soldering on the board. It's probably a good idea to clean any remaining solder flux residue of the board. I use distilled water and an old toothbrush. If you are not using water-soluble flux, you may need to use a solvent of some sort. Solder joints should be shiny and neat. Before proceeding wait until the board is completely dry.

Use the continuity tester of your multimeter to check that there are no shorts between +5V and GND. Check this by putting the probes on pins 2 and 6. If there's continuity check your soldering again.

Before inserting the chips, plug the retroSpeak PCB onto your Raspberry Pi, and switch it on. If the Pi doesn't boot, switch it off immediately. Remove retroSpeak and check the Pi again. If it doesn't boot with retroSpeak attached, you have a problem somewhere, and check your soldering again.

If all is well and the Pi boots, *carefully* probe for 3.3V and 5 volts at the chip sockets. Clip the negative probe of the multimeter to a ground point on the circuit. There's one on the retroSpeak GPIO header – use a flying wire for safety. Shorting 5V or 3.3V to ground will at the least crash the Pi, and may lead to filesystem corruption on the SD card.

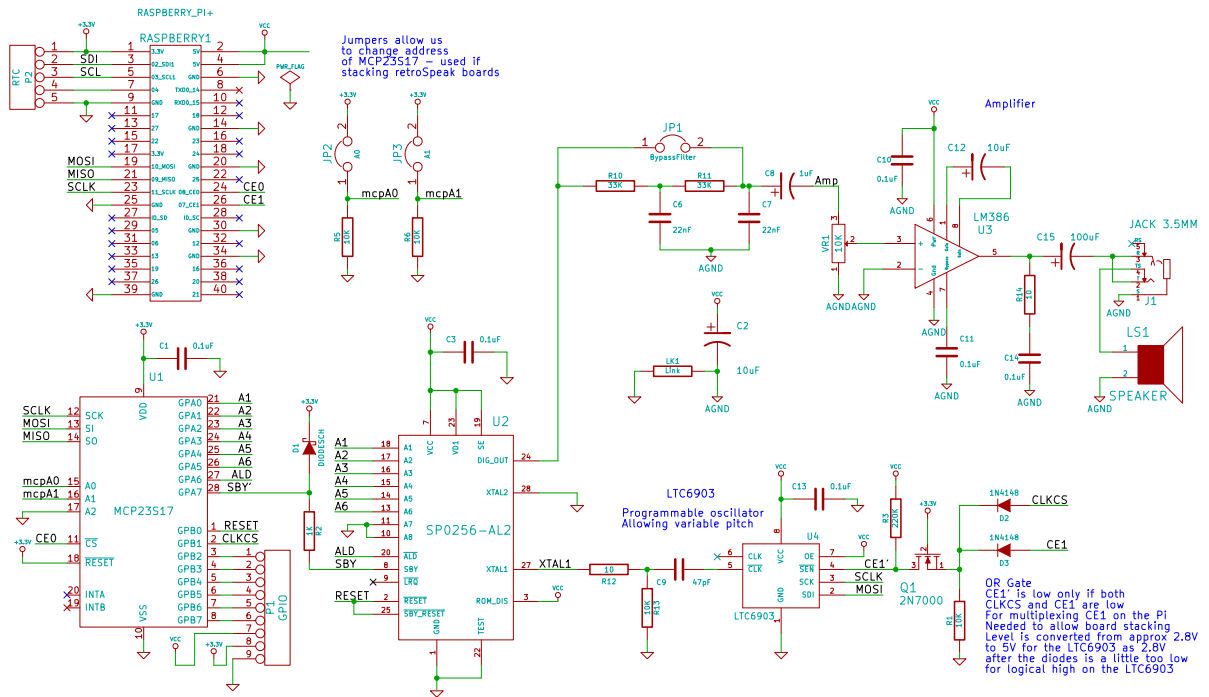
There should be voltages (3.3V or 5V or very close) at the following pins on the chip sockets:

- U1: narrow 28pin – 3.3V on pins 9 (VCC), 18 (/RESET)
- IC2: wide 28pin – 5V on pins 7 (VCC), 23 (VD1), 19 (SE)
- U3: 8pin – 5V on pin 6 (VCC)
- U4: tiny 8pin SMD – 5V on pins 8 (VCC), 7 (OE)

If everything is OK, shutdown the Pi and remove the power lead.

Now carefully insert the SP0256-AL2, MCP23S17 and LM386 ICs into their respective sockets, ensuring they are oriented correctly.

Schematic



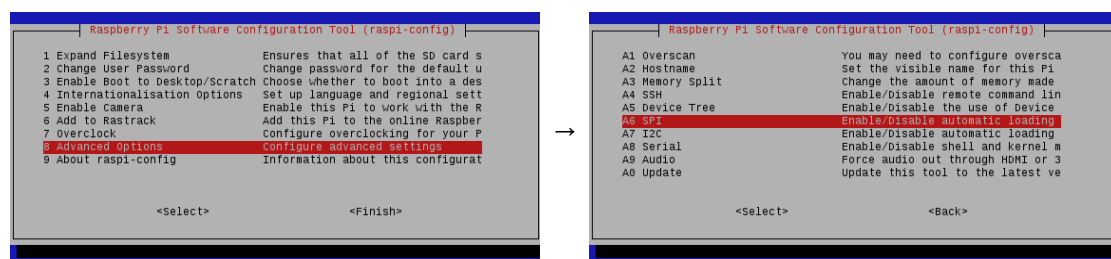
Setting up SPI on the Pi

RetroSpeak is tested on a Raspberry Pi A+, B+ and Model B version 2 using Raspbian downloaded from <http://www.raspberrypi.org/downloads/>. The NOOBS image was used. These instructions assume that a fresh install of NOOBS is used, so some steps may have already been taken on your version.

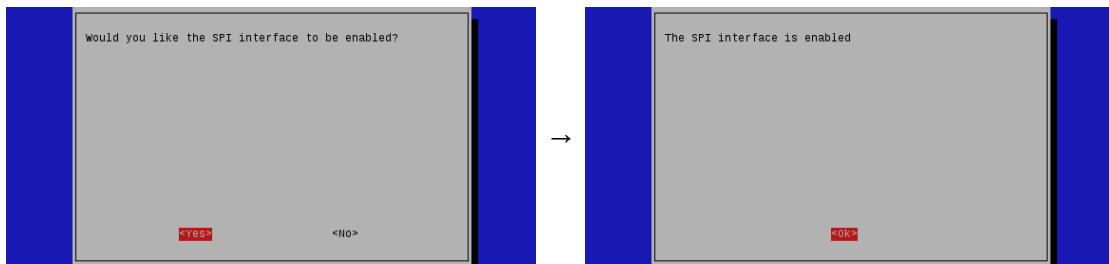
First step is to set up the Pi to be able to access SPI. You can easily do this in raspi-config in the advanced section.

To enable SPI start raspi-config from a command line:

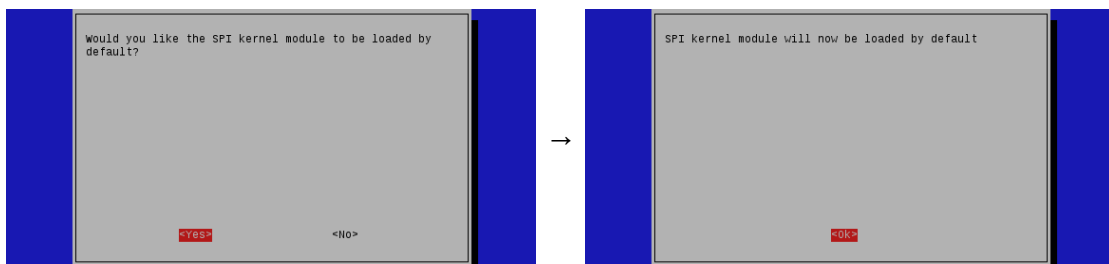
```
sudo raspi-config
```



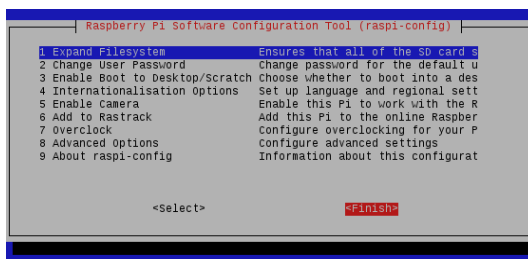
Use the arrow keys to select Advanced Options, press Enter, then use the arrow keys to select SPI and press Enter again.



Enable the SPI interface. If <No> is highlighted use the left/right arrow keys to select <Yes> and press Enter to select and again to confirm the <OK> prompt.



Then allow the kernel module to be loaded by default. If <No> is highlighted use the arrow keys to select <Yes> and press Enter to select and again to confirm the <OK> prompt.



Finally use the Tab key to move the highlight to the bottom row. Use the left/right arrow keys to select <Finish>.

Finally reboot your Pi with the command:

```
sudo reboot
```

Software

All the programs for retroSpeak are written in Python, though there is no reason why other languages like C++ couldn't be used. The wiringpi2 library makes it easy to interface the MCP23S17 and SPI port. Look at the Python code for clues about how it works. First of all WiringPi2 and WiringPi2-Python need to be installed.

Install latest WiringPi2 and WiringPi2-Python

Details are here: <https://github.com/Gadgetoid/WiringPi2-Python>

In a bash shell enter the following commands to install wiringPi and WiringPi2-Python:

```
sudo apt-get install python-dev python-setuptools
git clone git://git.drogon.net/wiringPi
cd wiringPi
sudo ./build
cd ..
git clone https://github.com/Gadgetoid/WiringPi2-Python.git
cd WiringPi2-Python
sudo python setup.py install
```


Install the retroSpeak library and scripts from GitHub

The Python scripts for controlling the retroSpeak board are at:

<https://github.com/jas8mm/retroSpeak/>

From a command line you can clone the repository by using:

```
git clone https://github.com/jas8mm/retroSpeak.git
```

Or alternatively use the  button to retrieve a zip file of the archive. Unpack this to a folder called retroSpeak in your user folder.

Yet another alternative is to just download the *.py Python scripts. Start with retroSpeak.py – use the download RAW option in GitHub.

Testing

Plug the retroSpeak board onto your Pi. Plug a speaker into the retroSpeak board and turn the volume control up. Then at a command prompt, enter the following:

```
cd retroSpeak
python retroSpeak.py
```

You should be greeted with the not-so dulcet tones of the SP0256-AL2 chip running at various speeds. What it is speaking and the allophones that make up the words are printed on the terminal.

To say the time and date, try:

```
python speakTime.py
```

There are some options on this command – so use:

```
python speakTime.py -h
```

to display them.

A very basic text-to-speech provided by the retroTTS.py script. It's not very good – but it is a start. Try:

```
python retroTTS.py -c 3.6 To be or not to be that is the question.
Whether tis Nobler in the mind to suffer The Slings and Arrows of out-
rageous Fortune Or to take Arms against a Sea of troubles.
```

It's no Olivier... but hey!

FAQS

The audio is very noisy.

Yes it is. It's a simple old-school circuit. The amplifier creates a lot of hiss, and picks up quite a lot of noise from the digital circuits on the Pi. The audio is generated by a PWM'd digital signal on the SP0256 – so isn't very hi-fi to start with. It's not so bad run through a speaker connected to the PCB or computer speakers. Through headphones it's a bit horrible. A non-amplified non-filtered signal can be taken from the filter-bypass jumper.

But eSpeak or <insert other speech synthesiser> is better

Yes it is. And better integrated to the system. And free. Their text-to-speech is much more sophisticated. Use that instead. retroSpeak is just a bit of fun. Old-school stuff.

It doesn't work – what is wrong?

Remove the ICs and follow the test procedure in the build guide above.

Check your soldering, and the voltages on the power pins of the chips. Make sure there are no shorts or missing components. Look for dry joints – reflow solder on dull joints etc. Take a close-up photo of the LTC6903, or use a magnifier and check the soldering.

Check the orientation of the chips, and polarised components – diodes and electrolytic capacitors.

An oscilloscope is very useful to check for the clock signal from the LTC6903. This arrives at pin 27 of the SP0256 socket. After power on the LTC6903 defaults to about 10kHz. When a retroSpeak program is run, it defaults to 3.12Mhz. A digital oscilloscope should count the frequency for you.

It could be the SP0256 is dead or faulty. I have built quite a few of the boards, and all work. I have come across DOA SP0256s from ebay sellers. But it's hardly surprising – this part is 30+ years old. There's also a possibility that Chinese sourced chips might be fake.

References

http://en.wikipedia.org/wiki/General_Instrument_SP0256

Archer/Radioshack datasheet: <http://www.futurebots.com/spo256.pdf> and at:
http://www.cpcwiki.eu/imgs/3/35/Sp0256al2_datasheet.pdf

Lots of useful info about the SP0256-AL2 chip:
<http://www.cpcwiki.eu/index.php/SP0256>

Speech synthesis on the Raspberry Pi: http://elinux.org/RPi_Text_to_Speech_%28Speech_Synthesis%29

Using an LTC6903 with a SP0256 and Arduino:
<http://elek101.blogspot.co.uk/2011/11/sp0256-al2-pitch-control.html>

Public domain text to speech in C using Naval Research Laboratory algorithm:
<ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/synthesis/english2phoneme.tar.gz>
by John A Wasser from 1985. I have converted the program to Python to give a basic text-to-speech capability to retroSpeak.

Naval Research Laboratory algorithm is described in detail (with SNOBOL code) in NRL Report 7948 January 21st, 1976 Elovitz et al (PDF):
<http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA021929>

Raspberry Pi HAT mechanical specs (PDF):
<https://github.com/raspberrypi/hats/blob/master/hat-board-mechanical.pdf?raw=true>

retroSpeak was designed using Kicad: <http://www.kicad-pcb.org>

Documents and illustrations created using LibreOffice
(<http://www.libreoffice.org/>) and Inkscape (<https://inkscape.org/>)

Raspberry Pi at <http://www.raspberrypi.org/>

© Jason Lane February 2015