# Guide Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Festival</td>
<td>3</td>
</tr>
<tr>
<td>Flite</td>
<td>3</td>
</tr>
<tr>
<td>Setting Up Your Pi</td>
<td>4</td>
</tr>
<tr>
<td>The Festival Speech Package</td>
<td>7</td>
</tr>
<tr>
<td>Speaking!</td>
<td>8</td>
</tr>
<tr>
<td>Speak Easier: Flite</td>
<td>9</td>
</tr>
<tr>
<td>Fun Uses for Speech</td>
<td>11</td>
</tr>
<tr>
<td>Reading to You</td>
<td>11</td>
</tr>
<tr>
<td>Reading Your Fortune</td>
<td>11</td>
</tr>
<tr>
<td>Emulating Talking Machines</td>
<td>11</td>
</tr>
<tr>
<td>WAV File Output</td>
<td>11</td>
</tr>
<tr>
<td>Reading the Weather</td>
<td>12</td>
</tr>
<tr>
<td>Integration With Python</td>
<td>14</td>
</tr>
<tr>
<td>Talking Clock</td>
<td>15</td>
</tr>
<tr>
<td>Clock Accuracy</td>
<td>15</td>
</tr>
<tr>
<td>Add a Button</td>
<td>15</td>
</tr>
<tr>
<td>Python Code</td>
<td>16</td>
</tr>
</tbody>
</table>
Overview

Talking computers were on fire 20 to 30 years ago. Movies like War Games and TV series like Knight Rider featured electronics speaking to their human operators. Speak-n-Spell machines taught a generation of children. Then the magic was out of the bottle and the focus drifted to other technologies.

Today, we have a new explosion of voice enabled devices. They announced this week the Amazon Echo will read Kindle books. You will see more devices speaking with the growth of the Internet of Things. Special speech synthesis chips of old are no longer required. The smallest of today’s Linux computers often has the capability to output speech. This includes the Raspberry Pi line of single board computers.

From the Raspberry Pi Zero to the A+/B+ to the Raspberry Pi 2, all have the capacity to run free software to turn text to speech.

This tutorial will show you how to have your Pi use the free software packages Festival and its derivative Flite to output voice.

Festival

Festival, written by The Centre for Speech Technology Research in the UK, offers a framework for building speech synthesis systems. It offers full text to speech through a number APIs: from shell level, via a command interpreter, as a C++ library, from Java, and an Emacs editor interface. Festival is multi-lingual (currently British English, American English, and Spanish. Other groups work to release new languages for the system. Festival is in the package manager for the Raspberry Pi making it very easy to install.

Flite

Flite is a lighter version of Festival built specifically for embedded systems. It has commands that make it easier to use than Festival on the command line. It runs faster than Festival. Unless you have the need for Festival’s complex scripting language or phoneme handling, Flite is the go-to program. Flite is also in the package manager.

There are other speech programs for Linux, including eSpeak (https://adafruit.it/l4e). If you find that Festival / Flite does not meet your needs, then check out eSpeak or other packages.
Setting Up Your Pi

If you have a fresh "out of the box" Raspberry Pi, you will need to connect a keyboard and display to the board and install the latest version of the Raspian operating system. The Adafruit Learning System has a series, Learn Raspberry Pi (https://adafruit.it/dDL), to assist in getting your Pi up and running successfully.

At this point, you have two ways to output audio on the Pi. All Raspberry Pi boards have an HDMI video out port which will also output audio if your display will also output audio. For most Raspberry Pi boards, there is also a headphone type audio output. The Models A and B use a dedicated 3.5 millimeter audio output jack. Later model A+, B+, and Raspberry Pi2 use a jack that outputs both stereo audio and video.

You should go ahead and select the HDMI or audio jack as the primary video output. The Raspbian distribution includes the raspi-config utility. From the command line, type

```
sudo raspi-config
```

If you are in the graphical environment, you can get to the command line by clicking the terminal icon circled in green below. That will pop up the black background terminal window. Then you can type commands.

When you type `sudo raspi-config`, you will get the text menu below:
Select Advanced Options (note this may be the eighth or ninth option depending on your raspi-config version you have). When selected, you will see:

To choose the audio output:

- **Auto**
  - 1 Force 3.5mm (‘headphone’) jack
  - 2 Force HDMI

I suggest if you have a specific output you are planning to use to select HDMI or Headphone jack. I had used HDMI then plugged in a headphone set. The voice was still over HDMI so the Auto is not quite as plug and play as on a PC or Mac.

Once you have made your selection, you can select <Ok> then <Finish> to get back to the command line.

At this point your Raspberry Pi should be ready to install the Festival software.
Note that you should use amplified speakers on your Pi. If you have very low volume on the audio jack, you probably need more amplification.
The Festival Speech Package

Festival is included in the Debian package library so installation is quite easy. At the command prompt type:

```
sudo apt-get install festival
```

And you should see something like:

```
pi@raspberrypi $ sudo apt-get install festival
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  festlex-cmu festlex-poslex festvox-kalpc16k libbestools2.1
Suggested packages:
  gedit-festival festival-freebf-utilities
The following NEW packages will be installed:
  festival festlex-cmu festlex-poslex festvox-kalpc16k libbestools2.1
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 144kB of additional disk space.
After this operation, 19.2 MB of additional disk space will be used.
Do you want to continue? [y/N] y
Get:1 http://mirror director. raspbian.org/raspbian/jessie/main festival-armhf 1:2.1.1-release-a [786kB]
Get:2 http://mirror director. raspbian.org/raspbian/jessie/main festival-armhf 1:2.1.6-release-8 [73kB]
Get:3 http://mirror director. raspbian.org/raspbian/jessie/main festival-cmu all 1.4.0-6 [88kB]
Get:4 http://mirror director. raspbian.org/raspbian/jessie/main festival-poslex all 1.4.0-5 [234kB]
Get:5 http://mirror director. raspbian.org/raspbian/jessie/main festvox-kalpc16k all 1.4.0-5 [4095kB]
Fetched 6,755 kB in 4min 11s (70.7 kB/s).
Selecting previously unused package libbestools2.1:armhf.
Reading database ... 123606 files and directories currently installed.
Preparing to unpack .../libbestools2.1:armhf 1:2.1.1-release-a ...
Unpacking libbestools2.1:armhf (1:2.1.1-release-a) ...
Selecting previously unused package festival.
Preparing to unpack .../festival 1:2.1.1-release-a-armhf.deb ...
Unpacking festival (1:2.1.1-release-a) ...
Selecting previously unused package festlex-cmu.
Preparing to unpack .../festlex-cmu 1.4.0-6_armhf.deb ...
Unpacking festlex-cmu (1.4.0-6) ...
Selecting previously unused package festlex-poslex.
Preparing to unpack .../festlex-poslex 1.4.0-6_armhf.deb ...
Unpacking festlex-poslex (1.4.0-6) ...
Processing triggers for menu (2.7.0.2-5) ...
Processing triggers for install-info (5.2.0.dfgsg.1.0) ...
```

Note that the install process asks if using more disk (SD card) space is ok (green circle), type **Y**.

The default install has a subset of available voice files. There are a number of US, British, and other language voices available. See the Debian package manager documentation (https://adafruit.io/l3d) for a current list. Each takes some disk space so you may not want them all. A demonstration of downloading an additional voice:

```
sudo apt-get install festvox-rablpc16k
```

Which adds a nicely sampled 16khz British Male speaker to the voice list.

**Before you go crazy downloading the different voices**, see the next page on the Flite package which comes with a nice selection of voices.

© Adafruit Industries https://learn.adafruit.com/speech-synthesis-on-the-raspberry-pi
Speaking!
Festival has a great amount of flexibility including different voices etc.

Let's start using Festival with some command line examples.

First, to test the installation, we will send a simple message to Festival via the Linux/unix echo command via a pipe. A Pipe can be thought of as receiving the result of the command to the left of it in a command. You may have seen using a pipe in verbose directory listings by typing "ls -l | more". The directory output was taken in by the more command to display one page at a time. So for speaking our first sentence, we'll type:

```
echo "Hello World!" | festival --tts
```

You should hear the Pi utter Hello World either over HDMI or the headphone jack. Adjust the volume if you do not hear it at first. If you have nothing (and no error message), be sure you have plugged in your audio (HDMI cable with audio out or headphones or amplified speaker on the audio jack). Then be sure you selected that audio output in raspi-config as listed on the previous page.

The Festival command line argument (https://adafru.it/nbn) --tts tells the program to treat the input as text to speech. With no arguments, Festival will put you into its interactive mode which you can exit by pressing the Ctrl key and simultaneously pressing the z key (abbreviated Ctrl-z) if you get stuck.

Typing festival --help gives you a quick listing of commands. But to get a full feeling of all Festival can do, you can go to the project main page (https://adafru.it/nbo) and read up including the extensive manual (https://adafru.it/nbp).

Something more useful is having Festival reading text from a file. You can have different files for different phrases, and invoke them in a shell script, Python, C or other programming environment.

For example, the text of US President Thomas Jefferson's inaugural address is available free by typing:

```
wget http://history.eserver.org/jefferson-inaugural.txt (https://adafru.it/nbc)
```

Then you can have festival read the text file by typing the command line

```
festival --tts jefferson-inaugural.txt
```

You can interrupt a voice stream by pressing the Ctrl-C key sequence.
Speak Easier: Flite

Flite (https://adafruit.it/nbr) (festival-lite) is a small, fast open source text to speech synthesis engine developed at CMU and primarily designed for small embedded machines and/or large servers. Flite is designed as an alternative text to speech synthesis engine to Festival for voices built using the FestVox suite (https://adafruit.it/nbs) of voice building tools.

Flite is also included in the Debian package library so installation is similar too Festival. At the command prompt type:

```
sudo apt-get install flite
```

And you should see something like:

```
pi@raspberrypi:~ $ sudo apt-get install flite
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  flite
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 234 kB of archives.
After this operation, 384 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ jessie/main flite armhf 1.4-release-12 [234 kB]
Fetched 234 kB in 4min 0s (973 B/s)
Selecting previously unselected package flite.
(Reading database ... 121582 files and directories currently installed.)
Preparing to unpack .../flite_1.4-release-12_armhf.deb ...
Unpacking flite (1.4-release-12) ...
Processing triggers for man-db (2.7.0.2-5) ...
Processing triggers for install-info (5.2.0.0-0.1.6) ...
Setting up flite (1.4-release-12) ...
```

You are now set to try it out. For simple text, use text after the -t flag:

```
flite -t "All good men come to the aid of the rebellion"
```

and you can have flite speak the contents of a file with -f

```
flite -f jefferson-inaugural.txt
```

Sounds pretty good, eh?

You are not limited to the default voice. If you type

```
flite -lv
```

you get a list of available voices like this:

```
pi@raspberrypi:~ $ flite -lv
voices available: kal awb_time kal16 awb rms slt
```

To use a different voice, use the -voice flag followed by one of the voices in the listing

```
pi@raspberrypi:~ $ flite -voice kal16 -t "All good men come to the aid of the rebellion"
```
flite -voice awb -t "The Raspberry Pi is a great Maker platform!"

The voices available as of this tutorial are:

- US English Male RMS (https://adafru.it/nbt)
- US English Male JMK (https://adafru.it/nbu)
- US English Male BDL (https://adafru.it/nbv)
- US English Male AEW (https://adafru.it/nbw)
- US English Female SLT (https://adafru.it/nbx)
- US English Female CLB (https://adafru.it/nby)
- US English Female LJM (https://adafru.it/nbz)
- US English Female EEY (https://adafru.it/nbA)
- US English (Scottish accent) Male AWB (https://adafru.it/nbB)
- US English (German accent) Male AHW (https://adafru.it/nbC)
- US English (German accent) Male FEM (https://adafru.it/nbD)
- US English Male RXR (https://adafru.it/nbE)
- Indian English Male AUP (https://adafru.it/nbF)
- Indian English Male GKA (https://adafru.it/nbG)
- Indian English Male KSP (https://adafru.it/nbH)
- Indian English Female AXB (https://adafru.it/nbl)

If you would like one of the voices listed but not installed by default, they can be downloaded from the Flite website (https://adafru.it/nbJ).
Fun Uses for Speech

Now to turn the cute to the useful. Organizations are predicting that voice will surpass typing in the future. If voice will be the primary input method, voice can be the informational output (it is much more useful to get information with earbuds than carrying a 24 inch LCD display with you).

Reading to You

As previously discussed, both Festival and Flite will read any text file:

```bash
festival -tts jefferson-inaugural.txt
flite -f jefferson-inaugural.txt
```

Speaking content in other file formats usually requires a conversion of the file to a text format. PDF is a popular format for documents and some eBooks. Fortunately the default Raspbian Jessie has a great utility, pdftotext, which has a lot of options if you need them (you can just type `pdftotext` to get the list). The basics are rather easy. Say you have `mydocument.pdf` on your Pi. You can have Flite read this using the command:

```bash
pdftotext mydocument.pdf mydocument.txt
flite -f mydocument.txt
```

It should work that you could type `pdftotext mydocument.pdf|flite` but it does not appear to work with the version of `pdftotext` on Raspbian at present.

Reading Your Fortune

Nearly all Unix and Linux systems have a utility called `fortune` which reads you a random quote for the day. I'm not sure why Raspbian does not have it by default but it is available to load:

```bash
sudo apt-get install fortune-mod
```

Then whenever you want to have a new fortune read, type:

```bash
fortune | flite
```

To read a selection of fortunes to see what output is generated, just type `fortune` at the command prompt. To read about adding your own quotes to the fortune database on your Pi, see this thread on the Rasberry Pi forums [https://adafruit.it/nbK](https://adafruit.it/nbK).

Emulating Talking Machines

You can have fun text files also. You can Google the phrases from your favorite talking object: K.I.T.T (the talking car in Knight Rider & the Hoff!), The W.O.P.R. missile control computer from the movie War Games, or hacking the Gibson from Hackers. You can have these spoken via the command shell or in your python program (more on that in a bit).

WAV File Output

If you have only a fixed set of phrases, it might be better to have the sentences in an audio file instead of generating
them every time you want them spoken. Fortunately Flite will save its voice to a wav file (a standard sound file format):

```
flite -t "Shall we play a game?" -o wargames1.wav
flite -t "Lets play global thermonuclear war." -o wargames2.wav
```

You can then use a number of Linux programs to output the text file when you need to. `aplay` is one such program already on the Raspberry Pi:

```
aplay wargames1.wav
aplay wargames2.wav
```

You can even copy these wav files to your PC/Mac or Arduino wave shield SD card and play the voices on other equipment.

Reading the Weather

There is a cool utility for Linux, weather-util, that is able to look up local weather given a location code. Install the package:

```
cd ~
wget http://fungi.yuggoth.org/weather/src/weather-2.2.tar.gz
tar -zxvf weather-2.2.tar.gz
cd weather-2.2/
./weather nyz072
```

When loaded, you can type `weather` to see what options there are. The program will accept airport identification codes, city and state names, and more. It will search an online database to find matching locations. If the number of locations is too big (typing `weather washington` for example), it will ask for a narrower search. If you get close but still too many, it will prefix each with a “fips number”. Typing weather followed by a specific fips number will pick that unique location. For example:

```
pi@raspberrypi:~ $ weather washington
Searching via name...
Your search is too ambiguous, returning 398 matches.
```

Ok, let's be more specific:

```
pi@raspberrypi:~ $ weather boise idaho
Searching via name...
Your search is ambiguous, returning 8 matches:
[fips:1600100345] Boise City CCD, ID
[fips:1600190369] Boise Hills CCD, ID
[fips:160015] Boise County, ID
[fips:16008830] Boise City, ID
[fips:4002500286] Boise City CCD, OK
[fips:40067200] Boise City city, OK
[fips:0001] Boise Mountains, ID
[kboi] Boise Air Terminal, ID, United States
```

So to get the specific location, we can type the fips number:
weather will now cache this location so it does not have to do such an extensive search next time. For speaking the weather, we do not want any of the text before the temperature, so caching one time and adding the -q (quiet) flag gets us the text we want to be read. So to speak the Boise weather, we can feed the weather output into flite:

```
weather -q fips1600190345 | flite
```

and it will sound like:

https://adafruit.it/nbM
Integration With Python

While there is a C application programming interface for Flite (https://adafruit.it/nbN), it might be easier to just use a system call to run Flite to do the talking in your Python code.

Adafruit has a great tutorial on hooking buttons to your Raspberry Pi and using Python to output sounds: Playing sounds and using buttons with Raspberry Pi (https://adafruit.it/nbO). That tutorial uses the mpg123 program to play sounds. We'll change the code to use Flite and say programmed sentences:

```python
#!/usr/bin/env python
import os
from time import sleep
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN)
GPIO.setup(24, GPIO.IN)
GPIO.setup(25, GPIO.IN)
while True:
    if (GPIO.input(23) == False):
        os.system('flite -t "Use the force Luke!" ')
    if (GPIO.input(24) == False):
        os.system('flite -voice SLT -t "Some rescue!" ')
    if (GPIO.input(25) == False):
        os.system('flite -t "I find your lack of faith disturbing." ')
    sleep(0.1);
```

Use the Playing Sounds tutorial to see how the buttons are wired to the Pi GPIO pins. Note: If you switch pins from the tutorial, you'll need to switch pin numbers in the program. I have used some Star Wars phrases, you can use any sentences you want.

You can also programmatically change the sentences depending on your code. We'll modify the Adafruit tutorial MCP9808 Temperature Sensor Python Library (https://adafruit.it/nbP). The tutorial shows wiring the sensor to your Pi and printing out values. Instead of printing the values, suppose we have Flite read the value to you. The code would change to:

```python
import os
# Loop reading measurements every minute.
print 'Press Ctrl-C to quit.'
while True:
    temp = sensor.readTempC()
    os.system("flite -t 'The temperature is " + temp + " degrees'" &)
time.sleep(60.0)
```

The sky is the limit as far as having sentences change depending on sensors, time of day, the weather, and more.
How many times have you wished you could have something just speak the time of day? Well with just a more work with Flite, a talking clock is yours.

The Unix/Linux date command provides a near infinite amount of flexibility in providing the date and time.  

```bash
date --help
```
lists all the available options.  Here are some handy formatting for date that work well spoken:

- `date +%T` the time in hh:mm:ss format
- `date +%H:%M` the time in hour and minute (24 hour time)
- `date "%H:%M %P"` the time in hour and minute with AM or PM at the end
- `date +%D` the date in mm/dd/yy format
- `date "+%A %B %d %Y"` the date like Sunday January 23 2016
- `date "%A %d %B %Y"` the date like Sunday 23 January 2016

So take your favorites and you can add the commands to a shell script or python program like we did in the previous page:

```bash
date "+%H:%M %P" | flite                             (for just a clock)
date "+%H:%M %P %A %d %B %Y" | flite    (adds the date)
```

Clock Accuracy

There are some options.  If your Raspberry Pi is connected to the Internet, the operating system (Raspbian) gets the time from the Internet from a Network Time Protocol (NTP) server.  So you have a good time sync, accurate within a second or two.  The Pi does not have a battery backed real-time clock (RTC) to keep time in case it is not connected to the Internet or if it loses power and does not have an Internet connection.  Several third-party add-on boards (sometimes called HATS) provide a hardware RTC that will keep very accurate time, sometimes even temperature compensated.  Finally, a GPS receiver can not only provide position data, but a GPS connected board can get the atomic accurate time from satellites as well.  GPS is not often used for time as it is more expensive and less common.

Add a Button

Reading the time of day in a loop is rather tiring. We need a trigger to read the time when an event happens. The standard, safe way to add a button to a Pi GPIO pin is shown in the schematic below. Safe in that no overcurrent will be placed on the GPIO when the button is pressed or released.
The Fritzing diagram for the button connected to the Pi:

Python Code

The code to check the switch in a loop and say the time if the button is pressed:
import time
import RPi.GPIO as GPIO
import os

GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.IN)

#initialise a previous input variable to 0 (assume button not pressed last)
prev_input = 0
while True:
    #take a reading
    input = GPIO.input(17)
    #if the last reading was low and this one high, print
    if ((not prev_input) and input):
        print("Button pressed")
        os.system("date '+%I:%M %P' | flite")
    #update previous input
    prev_input = input
    #Slight pause to debounce
    time.sleep(0.05)

And here it is in action:

https://adafruit.it/nbQ