# Table of Contents

**Overview** 3

- Festival
- Flite

**Setting Up Your Pi** 4

**The Festival Speech Package** 7

**Speaking!** 7

**Fun Uses for Speech** 8

- Reading to You
- Reading Your Fortune
- Reading the Weather

**Integration With Python** 11

- CircuitPython Library Support
- Saying Sounds and Using Buttons
- Speaking Temperature

**Talking Clock** 13

- Clock Accuracy
- Add a Button
- Python Code

**Bonus : Speak Easier: Flite** 16

- Building Flite
- Reading Text
- WAV File Output
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 so we have included as an optional bonus at the end of the tutorial. The downside to using Flite is that the current package as of April 2019 was not compiled correctly for Raspbian so it takes a few more minutes to get it up and running.
There are other speech programs that work well on Raspbian so if festival or flite do not meet your needs consider the following.

- espeak
  - multi-lingual software speech synthesizer
  - `sudo apt-get install espeak`

- say
  - converts text to audible speech
  - `sudo apt-get install gnustep-gui-runtime`

- spd-say
  - sends text-to-speech output request to speech-dispatcher
  - `sudo apt-get install speech-dispatcher`

---

**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://learn.adafruit.com/), 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.

For the Raspberry Pi Zero, only HDMI audio is available out of the box. If you wish to hack a separate dedicated audio output, see the Adafruit tutorial at [https://learn.adafruit.com/introducing-the-raspberry-pi-zero/audio-outputs](https://learn.adafruit.com/introducing-the-raspberry-pi-zero/audio-outputs)

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:

```
sudo raspi-config
```

Select Advanced Options (note this may be the 7th option depending on the raspi-config version you are running). When selected, you will see:
Select the Audio option and you will see:

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 -y libasound2-plugins festival
```

Speaking!

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 --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 and read up including the extensive manual.

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 https://ia800202.us.archive.org/14/items/uspresidentialin04938gut/inagu10m/Thomas_Jefferson.txt
```

Then you can have festival read the text file by typing the command line. You can interrupt a voice stream by pressing the Ctrl-C key sequence.

```
festival --tts Thomas_Jefferson.txt
```

---

**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**

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 Rasbian 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.

```bash
pdftotext mydocument.pdf mydocument.txt
festival --tts mydocument.txt
```

**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 | festival --tts
```

To read a selection of fortunes to see what output is generates, 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://www.raspberrypi.org/forums/).

**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:

```bash
sudo apt-get install weather-util
```
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:

```
weather nyz072
```

Using a broad query such as "washington" results is too ambiguous.

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

Ok, let's be more specific:

```
pig@raspberrypi:~ $ weather boise idaho
Searching via name...
Your search is ambiguous, returning 8 matches:
[fips 16001000345] Boise City, ID
[fips 1600900362] Boise County, ID
[fips 16001000361] Boise City, ID
[fips 16001000363] Boise City, ID
[fips 16001000362] Boise City, ID
[fips 16001000361] Boise City, ID
[fips 16001000360] Boise City, ID
[fips 16001000359] Boise City, ID
```

So to get the specific location, we can type the fips number:

```
pig@raspberrypi:~ $ weather fips16001000345
Searching via FIPS...
[fips 16001000345] Boise Air Terminal, ID
Current conditions at Boise Air Terminal, ID
Last updated Jan 23, 2010 - 07:58 AM MST / 2010.01.23 0758 UTC
Temperature: 41.0 F (5.0 C)
Relative Humidity: 72%
Wind: From the SE (140 degrees) at 15 MPH (13 KT)
Sky conditions: mostly clear
```

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:
and it will sound like:

weather2.wav

Integration With Python

Below we provide two examples of generating speech with the festival program.

1. Saying Star Wars quotes based on button presses.
2. Speaking Temperature using the MCP9808 sensor.

CircuitPython Library Support

If you are already running the latest Raspian distribution you will need to install these two packages to run the CircuitPython examples in this guide.

```
sudo pip3 install adafruit-blinka
```

```
sudo pip3 install adafruit-circuitpython-mcp9808
```

Saying Sounds and Using Buttons
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://learn.adafruit.com/playing-sounds-and-using-buttons-with-raspberry-pi). That tutorial uses the omxplayer program to play sounds. We'll change the code to use festival and say programmed sentences:

```python
import time
import os
import board
import digitalio

button1 = digitalio.DigitalInOut(board.D23)
button1.direction = digitalio.Direction.INPUT
button1.pull = digitalio.Pull.UP

button2 = digitalio.DigitalInOut(board.D24)
button2.direction = digitalio.Direction.INPUT
button2.pull = digitalio.Pull.UP

button3 = digitalio.DigitalInOut(board.D25)
button3.direction = digitalio.Direction.INPUT
button3.pull = digitalio.Pull.UP

print("press a button!")

while True:
    if not button1.value:
        os.system('echo "Use the force Luke!" | festival --tts')
    if not button2.value:
        os.system('echo "Some rescue!" | festival --tts')
    if not button3.value:
        os.system('echo "I find your lack of faith disturbing." | festival --tts')

time.sleep(0.1)
```

We can easily copy this code onto our Pi using the 'wget' command and run it using the following python syntax.

```
sudo python3 ./saying_sounds_using_buttons.py
```

### Speaking Temperature

You can also programmatically change the sentences depending on your code. We'll modify the Adafruit tutorial [MCP9808 Temperature Sensor Python Library](https://learn.adafruit.com/mcp9808-temperature-sensor-python-library). 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:
import os
import time
import board
import busio
import adafruit_mcp9808

# This example shows how to get the temperature from a MCP9808 board
i2c_bus = busio.I2C(board.SCL, board.SDA)
mcp = adafruit_mcp9808.MCP9808(i2c_bus)

while True:
    # print precise temperature values to console
    tempC = mcp.temperature
    tempF = tempC * 9 / 5 + 32
    print(f'Temperature: {tempC} C {tempF} F')

    # drop decimal points and convert to string for speech
    tempC = str(int(tempC))
    tempF = str(int(tempF))
    os.system(f'echo "The temperature is {tempC} degrees" | festival --tts')

    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.

We can easily copy this code onto our Pi using the 'wget' command and run it using
the following python syntax.

wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/
    master/Speech_Synthesis_on_the_Raspberry_Pi/speaking_temperature.py
sudo python3 ./speaking_temperature.py

Talking Clock

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. `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 "+%I:%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 (http://adafru.it/2345) (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 (http://adafru.it/2324) 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 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:

```python
# SPDX-FileCopyrightText: 2019 Mikey Sklar for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import os
import board
import digitalio

button = digitalio.DigitalInOut(board.D18)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP

while True:
    if not button.value:
        os.system("date '+%I:%M %P' | festival --tts")

    # slight pause to debounce
    time.sleep(0.2)
```

We can easily copy this code onto our Pi using the 'wget' command and run it using the following python syntax.

```
sudo python3 ./talking_clock.py
```

And here it is in action:
Bonus : Speak Easier: Flite

Flite (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 of voice building tools.

While flite can be installed using the usual 'apt-get install flite' the current version will not work on Raspberry Pi's as of April 2019. The package was built without support for alsa which is required for Raspberry Pi. We will walk you through building your own version from scratch along with a few example commands for usage. You can replace festival in our code examples with "flite" should you need a faster more flexible program.

Building Flite

Method 1:

There should be a package available. At the command line:

```
sudo apt install flite
```

And try the Reading Text section if it appears to work. If this method doesn't work, try the following to get the source and install it:

Method 2:
Move to the Downloads folder and install the necessary libasound2-dev package that flite requires:

```
    cd ~/Downloads
    sudo apt-get install libasound2-dev
```

Next we will grab the latest version of the flite "tarball".

```
```

Now we can uncompress the files, change into the new source code directory and start the configure process.

```
    tar -xvf flite-2.1-release.tar.bz2
    cd flite-2.1-release
    ./configure --with-audio=alsa --with-vox=awb
```
It will take several minutes to compile the source even on today's fastest Raspberry Pi B+. We will use the make command command to start the compilation.

```
make
```

Once the make command has completed without errors we can install the files using the following command:

```
sudo make install
```

---

**Reading Text**

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 Thomas_Jefferson.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 kall6 awb rms slt
```

To use a different voice, use the `-voice` flag followed by one of the voices in the listing
flite -voice awb -t "The Raspberry Pi is a great Maker platform!"

If you would like one of the voices listed but not installed by default, they can be downloaded from the Flite website.

There is a large selection of voices to choose from in the download area of the Flite website.

**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.