
Darth-Vader-RPi

Release 0.1.0a0

Raul C.

Sep 15, 2020

CONTENTS

| | | |
|----------|------------------------------------|-----------|
| 1 | README | 3 |
| 2 | The main configuration file | 9 |
| 3 | Change the default settings | 21 |
| 4 | API Reference | 31 |
| 5 | Changelog | 41 |
| 6 | License: GPL3 | 43 |
| 7 | Indices and tables | 57 |
| | Python Module Index | 59 |
| | Index | 61 |

Darth-Vader-RPi (0.1.0a0) is a Python-based Raspberry Pi project about activating a Darth Vader action figure by turning on LEDs on his suit and lightsaber, and by playing sounds such as some of his famous quotes. See the [README](#) for more info about the project.

README

Darth-Vader-RPi (0.1.0a0) is a Python-based Raspberry Pi (RPi) project about activating a Darth Vader action figure by turning on LEDs on his suit and lightsaber, and by playing sounds such as some of his famous quotes.

Important: If you don't have an RPi, don't worry. You can still *test the script on your own computer* because the `darth_vader_rpi` package uses the [SimulRPi](#) library to simulate I/O devices connected to an RPi such as LEDs and push buttons by blinking red dots in the terminal and playing sounds when a keyboard key is pressed. Almost like testing with a real RPi!

Disclaimer: I also wrote the [SimulRPi](#) library

Table of contents

- *Introduction*
- *Connection diagram*
- *Dependencies*
- *Installation instructions*
- *Usage*
 - *Script `start_dv`*
 - * *List of options*
 - *Simulating on your computer*
- *How to uninstall*
- *Credits*
- *Resources*
- *References*

1.1 Introduction

The Darth Vader action figure is 11.5 inches tall (which is [this one from Hasbro](#)) and was modified to make it more lifelike by illuminating the lightsaber, chest control box, and belt. 3 push buttons control the following sounds and LEDs:

1. Some of his famous quotes
2. The *Imperial march* theme song
3. The lightsaber drawing, hum and retraction sounds
4. The lightsaber illumination (3 LEDs)

His iconic breathing sound plays in the background indefinitely almost as soon as the RPi is run with the `start_dv` script.

1.2 Connection diagram

Here's how the various LEDs and push buttons are connected to the Raspberry Pi:

- The lightsaber is illuminated by 3 LEDs connected in parallel
- The *Top*, *Middle*, and *Bottom* LEDs illuminate the slots in Darth Vader's chest control box. They blink in a specific sequence [as specified in the configuration file](#).
- When the *Lightsaber button* is first pressed, it produces the drawing sound, illuminates the sword, and a hum sound persists until the *Lightsaber button* is turned off. If the button is pressed again, it produces the sound of retracting the lightsaber and turns it off.

NOTE: the *Lightsaber button* can be pressed while the *Song button* or the *Quotes button* is playing audio since they use different audio channels.

- The *Song button* plays the [Imperial March song](#) by Jacob Townsend
- The *Quotes button* plays famous Darth Vader quotes when pressed. For testing purposes, the `darth_vader_rpi` package comes with two movie lines:
 - “I am your father”
 - “Nooooo”: it is also used for the [closing sound](#) when the `start_dv` script exits

However, you could [add more quotes](#) if you want.

1.3 Dependencies

- **Platforms:** macOS, Linux
- **Python:** 3.5, 3.6, 3.7, 3.8
- **Packages**
 - `dv_sounds` $\geq 0.1.0a0$: for retrieving the sound files (quotes, songs, and sound effects)
 - * **Ref:** [dv_sounds](#)
 - `pygame` $\geq 1.9.3$: for playing sounds
 - * **Ref:** [pygame](#)

- `pynput` >=1.6.8: needed by `SimulRPi` for monitoring the keyboard when simulating push buttons with keyboard keys, i.e. when running the `start_dv` script in simulation mode
 - * **Ref:** `pynput`
- `SimulRPi` >=0.1.0a0: for partly faking `RPI.GPIO` and simulating I/O devices connected to an RPi such as LEDs and push buttons in case that you don't have access to an RPi
 - * **Ref:** `SimulRPi`

Important: At the time of this writing (2020-08-28), I couldn't install `pygame` 1.9.6 (the latest stable release) with Python 3.5 and 3.8, on macOS. However, the latest pre-release development version (2.0.0.dev10) worked well with Python 3.5 and 3.8, on macOS.

1.4 Installation instructions

1. Make sure to update `pip`:

```
$ pip install --upgrade pip
```

2. Install the package `darth_vader_rpi` (latest release) with `pip`:

```
$ pip install git+https://github.com/raul23/Darth-Vader-RPi@v0.1.0a0#egg=Darth-  
↳Vader-RPi
```

It will install the dependencies if they are not already found in your system.

Important: Make sure that `pip` is working with the correct Python version. It might be the case that `pip` is using Python 2.x. You can find what Python version `pip` uses with the following:

```
$ pip -V
```

If `pip` is working with the wrong Python version, then try to use `pip3` which works with Python 3.x

Note: To install the **bleeding-edge version** of the `darth_vader_rpi` package:

```
$ pip install git+https://github.com/raul23/Darth-Vader-RPi#egg=Darth-Vader-RPi
```

However, this latest version is not as stable as the released version but you get the latest features being implemented.

Warning message

If you get the warning message from `pip` that the `start_dv` script is not defined in your `PATH`:

```
WARNING: The script start_dv is installed in '/home/pi/.local/bin' which is not on_  
↳PATH.
```

Add the directory mentioned in the warning to your `PATH` by editing your configuration file (e.g. `.bashrc`). See this [article](#) on how to set `PATH` on Linux and macOS.

Test installation

1. Test your installation by importing `darth_vader_rpi` and printing its version:

```
$ python -c "import darth_vader_rpi; print(darth_vader_rpi.__version__)"
```

2. You can also test that the dependencies were installed correctly:

```
$ python -c "import dv_sounds, pygame, pynput, SimulRPi"
```

1.5 Usage

1.5.1 Script `start_dv`

Once the `darth_vader_rpi` package is *installed*, you should have access to the `start_dv` script which turns on LEDs and plays sound effects on a Raspberry Pi (RPi).

Run the script on your **RPi** with *default values* for the GPIO channels and other settings:

```
$ start_dv
```

If you want to test the script on your **computer** (use the `-s` flag):

```
$ start_dv -s
```

Note: Both previous commands will use the default values from the *configuration file* (e.g GPIO channel numbers, channel volume).

Check *Change default settings* on how to modify these values.

Important: In order to stop the `start_dv` script at any moment, press `ctrl + c`.

List of options

To display the script's list of options and their descriptions:

```
$ start_dv -h
```

| | |
|-------------------------|--|
| --version | show program's version number and exit |
| -q, --quiet | Enable quiet mode, i.e. nothing will be printed. (default: False) |
| -s, --simulation | Enable simulation mode, i.e. <code>SimulRPi.GPIO</code> will be used for simulating <code>RPi.GPIO</code> . (default: False) |
| -v, --verbose | Print various debugging information, e.g. print traceback when there is an exception. (default: False) |

Edit a configuration file:

- e `cfg_name`, --edit `cfg_name`** Edit a configuration file. Provide **log_cfg** for the logging config file or **cfg** for the main config file. (default: None)
- a `APP`, --app-name `APP`** Name of the application to use for editing the file. If no name is given, then the default application for opening this type of file will be used. (default: None)

1.5.2 Simulating on your computer

If you don't have access to a Raspberry Pi (RPi) and want to try out the `start_dv` script, you can run it with the `-s` flag. It will make use of the [SimulRPi](#) library to simulate LEDs and push buttons connected to an RPi by blinking red dots in the terminal and monitoring pressed keyboard keys:

```
$ start_dv -s
```

NOTE: the last command makes use of default values. See [Change default settings](#) on how to change these values.

Here's how the keyboard keys are related **by default** to push buttons connected to an RPi:

- `cmd_l` —> lightsaber button
- `alt_l` —> song button
- `alt_r` —> quotes button

Check [Change keymap](#) if you want to change this default key-to-channel mapping.

Here is a video of what it looks like in a terminal when running the `start_dv` script on a computer instead of an RPi:

1.6 How to uninstall

To uninstall **only** the package `darth_vader_rpi`:

```
$ pip uninstall darth_vader_rpi
```

To uninstall the package `darth_vader_rpi` and its dependencies:

```
$ pip uninstall darth_vader_rpi dv_sounds pygame pynput simulrpi
```

You can remove from the previous command-line those dependencies that you don't want to uninstall.

Note: When uninstalling the `darth_vader_rpi` package, you might be informed that the configuration files `logging_cfg.json` and `main_cfg.json` won't be removed by `pip`. You can remove those files manually by noting their paths returned by `pip`. Or you can leave them so your saved settings can be re-used the next time you re-install the package.

Example:

```
$ pip uninstall darth-vader-rpi
Found existing installation: Darth-Vader-RPi 0.1.0a0
Uninstalling Darth-Vader-RPi-0.1.0a0:
  Would remove:
    /Users/test/miniconda3/envs/rpi_py37/bin/start_dv
    /Users/test/miniconda3/envs/rpi_py37/lib/python3.7/site-packages/Darth_Vader_RPi-
    ↪0.1.0a0.dist-info/*
    /Users/test/miniconda3/envs/rpi_py37/lib/python3.7/site-packages/darth_vader_rpi/*
  Would not remove (might be manually added):
    /Users/test/miniconda3/envs/rpi_py37/lib/python3.7/site-packages/darth_vader_rpi/
    ↪configs/logging_cfg.json
    /Users/test/miniconda3/envs/rpi_py37/lib/python3.7/site-packages/darth_vader_rpi/
    ↪configs/main_cfg.json
$ rm -r /Users/test/miniconda3/envs/rpi_py37/lib/python3.7/site-packages/darth_vader_
    ↪rpi
```

1.7 Credits

- **Darth Vader quotes:**
 - “I am your father”
 - “Nooooo”
- **Music:**
 - Imperial March song by Jacob Townsend is licensed under a [Creative Commons \(CC BY-NC-SA 3.0\) License](#)

NOTE: The original song file was reduced under 1 MB by removing the first 7 seconds (no sound) and the last 2 minutes and 24 seconds.
- **Sound effects:**
 - Darth Vader breathing sound
 - Darth Vader’s lightsaber sound effect
 - Darth Vader’s lightsaber retraction sound effect
- **Slot LEDs sequences:**
 - Empire Strikes Back chest box light sequence

1.8 Resources

- [Darth-Vader-RPi GitHub](#): source code

1.9 References

- [dv_sounds](#): a package for downloading the various sounds needed for the project, e.g. lightsaber sound effects
- [pygame](#): a package used for playing sounds
- [RPI.GPIO](#): a module to control RPi GPIO channels
- [SimulRPi](#): a package that partly fakes `RPi.GPIO` and simulates some I/O devices on a Raspberry Pi. It makes use of the [pynput](#) library for monitoring the keyboard for any pressed key.

THE MAIN CONFIGURATION FILE

The default settings used by the `start_dv` script are found in the [main configuration file](#). It is referred to as *main* because there is another config file you could edit, the [logging configuration file](#).

The *main* configuration file can be edited with the following command:

```
$ start_dv -e cfg
```

The logging configuration file could be instead edited with the `-e log_cfg` command-line option.

The previous command will open the configuration file with the default text editor that is associated with JSON files as specified in your system, e.g. *atom* on macOS or *vim* on Linux.

If you want to use another text editor you can specify it with the `-a APP` command-line option:

```
$ start_dv -e cfg -a TextEdit
```

In what follows, you will find an explanation for each setting found in the [main configuration file](#), presented in alphabetic order.

Important: Some of the settings (*quiet*, *simulation*, and *verbose*) in the configuration file can be also set through the script's command-line arguments. The command-line arguments override the settings found in the configuration file.

See also:

The script [start_dv](#)

2.1 audio_channels

Three audio channels are used for this project:

- **channel 0:** used for Darth Vader's breathing sound which plays in the background almost as soon as the `start_dv` script runs. Its volume is set by default at 0.2 since we don't want to overwhelm the other sounds playing in the other audio channels
- **channel 1:** used for playing the *Imperial March* song and all Darth Vader quotes. Its volume is set by default at 1.0
- **channel 2:** used for playing the lightsaber sound effects and the closing sound. Its volume is set by default at 1.0

The setting [audio_channels](#) in the configuration file defines these three audio channels with their default volume.

Listing 1: Audio channels and their default volume

```
"audio_channels": [  
  {  
    "channel_id": 0,  
    "channel_name": "breathing_sound",  
    "volume": 0.2  
  },  
  {  
    "channel_id": 1,  
    "channel_name": "song_and_quotes",  
    "volume": 1.0  
  },  
  {  
    "channel_id": 2,  
    "channel_name": "lightsaber_and_closing_sounds",  
    "volume": 1.0  
  }  
],
```

The Python package [pygame](#) is used for playing the various sounds used in this project.

Note:

- Volume takes values in the range 0.0 to 1.0 (inclusive).
- If value < 0.0, the volume will not be changed
- If value > 1.0, the volume will be set to 1.0

As per the [pygame](#) documentation.

2.2 default_led_symbols

The setting `default_led_symbols` in the configuration file defines the default LED symbols used by **all** output channels. A LED symbol is used for each output state (*ON* and *OFF*).

By **default**, the symbols used for representing LEDs in the terminal are the following:

```
"default_led_symbols": {  
  "ON": "",  
  "OFF": ""  
},
```

See also:

[Change LED symbols](#)

2.3 gpio_channels

The setting `gpio_channels` in the configuration file defines the GPIO pins connected to LEDs and push buttons.

GPIO channels for the following I/O devices are defined:

- **Lightsaber button:** it controls the LEDs and sound effects for the lightsaber
- **Song button:** when pressed it plays the *Imperial March* song
- **Quotes button:** when pressed it plays one of Darth Vader quotes
- **Slot LEDs:** they consist of three LEDs referred to as *Top*, *Middle*, and *Bottom* LEDs and are found on Darth Vader's chest control box
- **Lightsaber LEDs:** when the lightsaber button is pressed, these LEDs are turned ON/OFF

NOTE: on the Darth Vader's figurine, three LEDs are used to turn on the lightsaber as explained in the [Connection diagram](#). However, when simulating the Raspberry Pi with `SimulRPi`, only one LED is shown in the terminal

`gpio_channels` lists GPIO channel objects with the following properties:

- `channel_id`: this property should **not be modified** because it is used to uniquely identify the GPIO channels
- `channel_name`: it will be displayed in the terminal along with the LED symbol. By default, the channel number is displayed if `channel_name` is the empty string, i.e. `channel_name = ""`
- `channel_number`: it identifies the GPIO pin based on the numbering system you have specified (*BOARD* or *BCM*)
- `key`: it is only defined for button objects. It specifies the mapping between a keyboard key and a push button so you can simulate push buttons with your keyboard

Listing 2: **Example:** changing keymap for the Song button

```
{
  "channel_id": "song_button",
  "channel_name": "song_button",
  "channel_number": 24,
  "key": "shift_r"
}
```

- `led_symbols`: it is only defined for LED objects. It is a dictionary defining the symbols to be used when the LED is turned ON and OFF. If not found for a LED object, then the *default LED symbols* will be used

Listing 3: **Example:** changing the default LED symbols for the lightsaber LEDs

```
{
  "channel_id": "lightsaber_led",
  "channel_name": "lightsaber",
  "channel_number": 22,
  "led_symbols": {
    "ON": "\\033[1;31;48m(0)\\033[1;37;0m",
    "OFF": "(0) "
  }
}
```

Let's take a look at two GPIO channels found in the configuration file:

Listing 4: **Example:** GPIO channels for the lightsaber button and LEDs

```
"gpio_channels": [  
  {  
    "channel_id": "lightsaber_button",  
    "channel_name": "lightsaber_button",  
    "channel_number": 23,  
    "key": "cmd"  
  },  
  {  
    "channel_id": "lightsaber_led",  
    "channel_name": "lightsaber",  
    "channel_number": 22  
  }  
]
```

Thus, in this example, you have a push button connected to the GPIO pin 23 (based on the BCM mode), controlling the lightsaber by turning it ON/OFF and producing the lightsaber sound effects (drawing, retraction, and hum sounds). Also, the keyboard key `cmd` simulates the lightsaber push button when running the `start_dv` script on your computer.

Finally, the lightsaber LEDs are connected to GPIO pin 22 (BCM) and are turned ON/OFF when the corresponding push button (or `cmd` key) is pressed.

See also:

- [Change GPIO channel name and number](#)
- [Change keymap](#)
- [Change LED symbols](#)

2.4 mode

The setting `mode` in the configuration file defines the numbering system (*BOARD* or *BCM*) used to identify the GPIO channels. By default, *BCM* is used.

As per the [RPi.GPIO documentation](#):

BOARD refers to the pin numbers on the P1 header of the Raspberry Pi board. The advantage of using this numbering system is that your hardware will always work, regardless of the board revision of the RPi. You will not need to rewire your connector or change your code.

BCM is a lower level way of working - it refers to the channel numbers on the Broadcom SOC. You have to always work with a diagram of which channel number goes to which pin on the RPi board. Your script could break between revisions of Raspberry Pi boards.

2.5 quiet

The setting `quiet` in the configuration file is a flag (set to *false* by default) that allows you to run the `start_dv` script without printing anything on the terminal, not even the LED symbols when running the simulation nor the exceptions are printed.

However, you will still be able to hear sounds and interact with the push buttons or keyboard.

Listing 5: The setting `quiet` set to *false* by default

```
{
  "quiet": false,
  "simulation": false,
  "verbose": false,
  "mode": "BCM"
}
```

This flag can also be set directly through the script's `-q` command-line option:

```
$ start_dv -q
```

See also:

Script's list of options

2.6 quotes

The setting `quotes` in the configuration file defines all the Darth Vader's quotes used for this project.

By default, two movie lines are included:

- "I am your father"
- "Nooooo"

Each quote is represented in the configuration file as objects having the following properties:

- `id`: unique identifier
- `name`: it will be displayed in the terminal
- `filename`: it is relative to the directory *sounds_directory*
- `audio_channel_id`: all quotes should be played in **channel 1** as explained in *audio_channels*

Listing 6: **Example:** two Darth Vader quotes

```
"quotes": [
  {
    "id": "dont_make_me_destroy_you",
    "name": "Don't make me destroy you",
    "filename": "quote_dont_make_me_destroy_you.ogg",
    "audio_channel_id": 1
  },
  {
    "id": "give_yourself_to_the_dark_side",
    "name": "Give yourself to the dark side",
    "filename": "quote_give_yourself_to_the_dark_side.ogg",
```

(continues on next page)

(continued from previous page)

```
"audio_channel_id": 1
}
]
```

Important: All Darth Vader quotes should be played in **channel 1** as explained in [audio_channels](#)

See also:

- The setting [audio_channels](#)
- Add Darth Vader quotes
- Change channel volume
- Change paths to audio files

2.7 simulation

The setting [simulation](#) in the configuration file is a flag (set to *false* by default) that allows you to run the `start_dv` script on your computer, instead of a Raspberry Pi (RPi).

The `SimulRPi.GPIO` module is used in order to partly fake `RPi.GPIO` and simulate I/O devices connected to an RPi such as LEDs and push buttons by displaying LED symbols in the terminal and monitoring the keyboard for any pressed key.

This flag can also be set directly through the script's `-s` command-line option:

```
$ start_dv -s
```

Note: `SimulRPi.GPIO` makes use of the `pynput` package to monitor the keyboard for any pressed key.

See also:

[Script's list of options](#)

2.8 slot_leds

Three LEDs (labeled as *top*, *middle*, and *top*) illuminate the slots in Darth Vader's chest control box.

The setting [slot_leds](#) in the configuration file defines the sequence the slot LEDs should be turned ON/OFF. This sequence corresponds to Darth Vader's physiological state, e.g. if he is in a calm mood the slot LEDs will blink in a different pattern than if he was in action.

`slot_leds` is an object that takes the following properties:

- `delay_between_steps`: delay in seconds between each step in the sequence
- `time_per_step`: time in seconds each step will last
- `sequence`: the type of the sequence which can be either "*action*", "*calm*" or a *custom sequence*. The sequence will keep on repeating until the script exits

Listing 7: **Example:** a `slot_leds` object with the calm sequence

```
"slot_leds":{
  "delay_between_steps": 0.5,
  "time_per_step": 1,
  "sequence": "calm"
},
```

The user can also provide its own sequence by using a list of LED labels {*top*, *middle*, *bottom*} arranged in a sequence specifying the order the slot LEDs should turn ON/OFF.

Example: custom slot LEDs sequence

```
"sequence": [
  ["top", "bottom"],
  [],
  ["middle"],
  []
]
```

This simple sequence will turn ON/OFF the slot LEDs in this order:

1. top + bottom LEDs turned ON
2. All LEDs turned OFF
3. middle LED turned ON
4. All LEDs turned OFF

Each step in the sequence will lasts for `time_per_step` seconds and there will be a delay of `delay_between_steps` seconds between each step in the sequence. And the whole sequence will keep on repeating until the script exits.

Note: This is how the *action* and *calm* sequences are exactly defined:

Listing 8: **Action sequence**

```
"sequence": [
  ["top", "middle", "bottom"],
  ["top", "bottom"],
  ["top", "middle", "bottom"],
  ["top"],
  [],
  ["top", "middle", "bottom"],
  ["top"],
  ["top", "middle", "bottom"],
  ["middle", "bottom"],
  [],
  ["top", "bottom"],
  ["top", "middle", "bottom"],
  ["top", "bottom"],
  [],
  ["top"],
  []
]
```

Listing 9: **Calm sequence**

```
"sequence": [
  ["middle"],
  ["top"],
  ["middle"],
  ["top"],
  ["middle"],
  ["top"],
  ["top"],
  [],
  ["bottom"],
  []
]
```

Note: The default sequences of slot LEDs were obtained from this YouTube video: [Empire Strikes Back chest box light sequence](#).

See also:

[Change slot LEDs sequence](#)

2.9 songs

The setting `songs` in the configuration file defines the songs that can be played as part of the project.

At the moment, only the [Imperial March song by Jacob Townsend](#) is supported.

The setting `songs` takes a list of song objects having the following properties:

- `id`: this property should **not be modified** because it is used to uniquely identify the songs
- `name`: the name of the song which will be shown in the terminal
- `filename`: it is relative to the directory [sounds_directory](#)
- `audio_channel_id`: all songs should be played in **channel 1** as explained in [audio_channels](#)

Listing 10: The **Imperial March** song playing in audio channel #1

```
"songs": [
  {
    "id": "imperial_march_song",
    "name": "Imperial March song",
    "filename": "song_the_imperial_march.ogg",
    "audio_channel_id": 1
  }
],
```

Important: All songs should be played in **channel 1** as explained in [audio_channels](#)

See also:

- The setting [audio_channels](#)

- Change channel volume
- Change paths to audio files

2.10 sound_effects

The setting `sound_effects` in the configuration file defines the following sounds:

- **Breathing sound:** almost as soon as the `start_dv` script runs, Darth Vader's breathing sound starts playing in the background until the script ends
- **Lightsaber drawing sound:** when the lightsaber button is pressed, the drawing sound is played first followed by the hum sound which goes on until the button is pressed again which will produce the retraction sound
- **Lightsaber hum sound:** plays immediately after the lightsaber drawing sound and goes on until the lightsaber button is pressed again
- **Lightsaber retraction sound:** plays when the lightsaber button is pressed while the hum sound is playing
- **Closing sound:** plays after the user presses `ctrl + c` to exit from the script. By default, it is not played at the end

`sound_effects` takes a list of sound objects having the following properties:

- `id`: this property should **not be modified** because it is used to uniquely identify the sound effects
- `name`: it is the name of the sound which will be displayed in the terminal
- `filename`: it is relative to the directory `sounds_directory`
- `audio_channel_id`: the audio channel used for playing the sound. See [audio_channels](#) to know what channel is used for each type of sounds
- `mute`: it is only defined for the breathing and closing sounds. If set to `true`, the sound will not be played
- `loops`: only defined for the breathing sound. It is the number of times the sound should be repeated. If set to `-1`, it will be repeated indefinitely

Listing 11: **Example:** two sound effects playing in different audio channels

```
"sound_effects": [
  {
    "id": "breathing_sound",
    "name": "Breathing sound",
    "filename": "darth_vader_breathing.ogg",
    "audio_channel_id": 0,
    "mute": false,
    "loops": -1
  },
  {
    "id": "closing_sound",
    "name": "Nooooo [Closing]",
    "filename": "quote_nooooo.ogg",
    "audio_channel_id": 2,
    "mute": true
  }
]
```

Important: The breathing sound should use channel 0, while the other sound effects should use channel 2. Hence, the breathing sound can be heard in the background while a sound effect is also being played (e.g. the drawing sound of the lightsaber). See [audio_channels](#).

See also:

- The setting [audio_channels](#)
- [Change closing sound](#)
- [Change paths to audio files](#)
- [Mute breathing sound](#)

2.11 sounds_directory

The setting `sounds_directory` in the configuration file defines the directory where all the audio files are saved.

By default, `sounds_directory` points to the path where the `dv_sounds` package is installed. `dv_sounds` is used to download the various sounds (e.g. sound effects) needed for the project.

All the audio filenames found in the configuration file are defined relative to `sounds_directory`.

Listing 12: **Example:** Filename for the breathing-sound audio file

```
"sound_effects": [  
  {  
    "id": "breathing_sound",  
    "name": "Breathing sound",  
    "filename": "darth_vader_breathing.ogg",  
    "audio_channel_id": 0,  
    "mute": false,  
    "loops": -1  
  }  
]
```

In this example, the audio file `darth_vader_breathing.ogg` is to be found in the directory `sounds_directory`.

See also:

[Change paths to audio files](#)

2.12 verbose

The setting `verbose` in the configuration file is a flag (set to *false* by default) that allows you to run the `start_dv` script by logging to the terminal all messages (logging level is set to `DEBUG` when `verbose` is *true*). Also, when there is an exception, a traceback is printed so you can pinpoint exactly where the error occurred in the code which is not the case when running the script without `verbose` (you only get a one-line error message).

Listing 13: **Example:** running the script **without verbose**

```
ERROR      AttributeError: 'Namespace' object has no attribute 'edits'  
ERROR      Program exited with 1
```

Listing 14: **Example:** running the script with verbose

```
ERROR      'Namespace' object has no attribute 'edits'
Traceback (most recent call last):
  File "start_dv.py", line 795, in main
    if args.edits:
AttributeError: 'Namespace' object has no attribute 'edits'
ERROR      Program exited with 1
```

This flag can also be set directly through the script's `-v` command-line option:

```
$ start_dv -v
```

See also:

[Script's list of options](#)

CHANGE THE DEFAULT SETTINGS

- *Important tips*
- *Add Darth Vader quotes*
- *Change channel volume*
- *Change closing sound*
- *Change GPIO channel name and number*
- *Change keymap*
- *Change LED symbols*
 - *Case 1: change `default_led_symbols`*
 - *Case 2: change `gpio_channels`*
- *Change paths to audio files*
- *Change slot LEDs sequence*
- *Mute breathing sound*
- *Run the script as quiet or verbose*

3.1 Important tips

- This is the command to edit the [configuration file](#) with a **default** text editor as defined in your system:

```
$ start_dv -e cfg
```

Or with a **specific** text editor:

```
$ start_dv -e cfg -a APP_NAME
```

where `APP_NAME` is the name of a text editor, e.g. `TextEditor`

- To end the `start_dv` script, press `ctrl + c`
- When adding audio files, don't use `mp3` as the file format. Instead, use `ogg` (compressed) or `wav` (uncompressed). The reason is that `mp3` won't work well with pygame's simultaneous playback capability.

Reference: [stackoverflow](#)

3.2 Add Darth Vader quotes

If you want to add more Darth Vader quotes, you have to edit the setting `quotes` in the configuration file. Open the configuration file with:

```
$ start_dv -e cfg
```

Each quote is represented in the configuration file as objects having the following properties:

- `id`: unique identifier
- `name`: it will be displayed in the terminal
- `filename`: it is relative to the directory `sounds_directory`
- `audio_channel_id`: all quotes should be played in **channel 1** as explained in `audio_channels`

Add your quote object to the list in `quotes`, like in the following example:

Listing 1: **Example:** adding a new quote

```
"quotes": [  
  {  
    "id": "there_is_no_escape",  
    "name": "There is no escape",  
    "filename": "quote_there_is_no_escape.ogg",  
    "audio_channel_id": 1  
  },  
]
```

See also:

- The setting `audio_channels`
- The setting `quotes`
- *Change channel volume*
- *Change paths to audio files*

3.3 Change channel volume

To change the volume for an audio channel, open the configuration file and edit the channel's `volume` found in the setting `audio_channels`:

```
$ start_dv -e cfg
```

Listing 2: **Audio channels and their default volume**

```
"audio_channels": [  
  {  
    "channel_id": 0,  
    "channel_name": "breathing_sound",  
    "volume": 0.2  
  },  
  {  
    "channel_id": 1,  
    "name": "song_and_quotes",  
    "volume": 1.0  
  }  
]
```

(continues on next page)

(continued from previous page)

```

    },
    {
      "channel_id": 2,
      "name": "lightsaber_and_closing_sounds",
      "volume": 1.0
    }
  ],

```

What each channel controls:

- **Channel 0** controls Darth Vader’s breathing sound
- **Channel 1** controls the *Imperial March song* and all Darth Vader quotes
- **Channel 2** controls the lightsaber sound effects and the closing sound

Note: Volume takes values in the range 0.0 to 1.0 (inclusive). As per the [pygame documentation](#).

See also:

The setting `audio_channels`

3.4 Change closing sound

When the `start_dv` script is exiting after the user presses `ctrl + c`, a sound is produced. By default, no closing sound is produced and if it were to play, it would be the “Nooooo” quote.

To change the default closing sound, edit the setting `sound_effects` in the configuration file which can be opened with:

```
$ start_dv -e cfg
```

At the end of the list in `sound_effects`, you will find the `closing_sound` object. These are the properties you can modify for this object:

- `name`: it is the name of the sound which will be displayed in the terminal
- `filename`: it is relative to `sounds_directory`
- `mute`: if `true`, nothing will be played at the end of the script. Otherwise, the closing sound will be played when the script is terminating.

Listing 3: **Example:** choosing another closing sound by changing filename

```

"sound_effects": [
  {
    "id": "closing_sound",
    "name": "Bye [Closing]",
    "filename": "bye.ogg",
    "audio_channel_id": 2,
    "mute": false
  },

```

Note: By default, the closing sound is not played at the end of the `start_dv` script. Set its property `mute` to `true` in order to play the closing sound when the script exits.

See also:

- The setting `audio_channels`
- The setting `sound_effects`
- *Change channel volume*
- *Change paths to audio files*

3.5 Change GPIO channel name and number

The `GPIO channels` are identified in the terminal by their `channel_name` along with their LED symbols. If `channel_name` is not available, then its `channel_number` is shown.

The `channel_number` is the GPIO channel number of a pin used for connecting an I/O device (e.g. LED) and is defined based on the numbering system you have specified (*BOARD* or *BCM*).

To change a GPIO channel's `channel_name` and `channel_number`, open the configuration file with:

```
$ start_dv -e cfg
```

And edit its properties `channel_name` and `channel_number`, like in the following example.

Listing 4: **Example:** changing the `channel_name` and `channel_number` for the bottom LED

```
"gpio_channels": [  
  {  
    "channel_id": "bottom_led",  
    "channel_name": "Bottom LED",  
    "channel_number": 15  
  },  
]
```

Important: Don't change the property `channel_id` since it is used to uniquely identify the GPIO channels.

See also:

The setting `gpio_channels`

3.6 Change keymap

If you want to change the default keymap used for the three push buttons, edit the setting `gpio_channels` in the configuration file which can be opened with:

```
$ start_dv -e cfg
```

Listing 5: Default keymap used for the three push buttons

```

"gpio_channels": [
  {
    "channel_id": "lightsaber_button",
    "channel_name": "lightsaber_button",
    "channel_number": 23,
    "key": "cmd"
  },
  {
    "channel_id": "song_button",
    "channel_name": "song_button",
    "channel_number": 24,
    "key": "alt"
  },
  {
    "channel_id": "quotes_button",
    "channel_name": "quotes_button",
    "channel_number": 25,
    "key": "alt_r"
  },
]

```

In order to change the default keymap, you will need to change the value for `key` which refers to the name of the keyboard key associated with a given push button.

The names of keyboard keys that you can use are those specified in the [SimulRPi's documentation](#), e.g. `media_play_pause`, `shift`, and `shift_r`.

Listing 6: **Example:** choosing `shift_r` for the *Quotes* button

```

{
  "channel_id": "quotes_button",
  "channel_name": "quotes_button",
  "channel_number": 25,
  "key": "shift_r"
},

```

Note: On mac, I recommend using the following keyboard keys because they don't require running the `start_dv` script with `sudo`: `alt`, `alt_r`, `cmd`, `cmd_r`, `ctrl`, `ctrl_r`, `media_play_pause`, `media_volume_down`, `media_volume_mute`, `media_volume_up`, `shift`, and `shift_r`.

Ref.: Platform limitations

See also:

The setting `gpio_channels`

3.7 Change LED symbols

You can either:

1. change the default LED symbols used by **all** output channels, or
2. change the LED symbols for **specific** output channels

3.7.1 Case 1: change default_led_symbols

To change the default LED symbols used by **all** output channels, edit the setting `default_led_symbols` by opening the configuration file:

```
$ start_dv -e cfg
```

Add your LED symbols for each output state:

```
"default_led_symbols": {  
  "ON": "",  
  "OFF": " "  
},
```

3.7.2 Case 2: change gpio_channels

To change the LED symbols for **specific** output channels, edit the setting `gpio_channels` by opening the configuration file:

```
$ start_dv -e cfg
```

You need to modify the property `led_symbols` for a given LED object defined in `gpio_channels`.

Listing 7: **Example:** changing the symbols for the lightsaber LED

```
"gpio_channels": [  
  {  
    "channel_id": "lightsaber_led",  
    "channel_name": "lightsaber",  
    "channel_number": 22,  
    "led_symbols": {  
      "ON": "\\033[1;31;48m(0)\\033[1;37;0m",  
      "OFF": "(0) "  
    }  
  }  
]
```

Note: If you omit `led_symbols` as a property for a LED object, the default LED symbols will be used instead.

Important: If you are having problems displaying the default LED symbols when running the `start_dv` script, such as this error:

```
ERROR      UnicodeEncodeError: 'ascii' codec can't encode character '\U0001f6d1' in_  
↪position 2: ordinal not in range(128)
```

Then, you might have your locale settings set incorrectly. Check [Display problems](#) for more info about how to change them properly or other solutions.

See also:

- The setting `gpio_channels`
- *Change slot LEDs sequence*

3.8 Change paths to audio files

The setting `sounds_directory` in the configuration file defines the directory where all audio files (e.g. quotes) are saved.

Each audio object defined in the settings `quotes`, `songs` and `sound_effects` have a `filename` property that you can modify. The filename for each audio file is defined with respect to the directory `sounds_directory`.

Listing 8: **Example:** filename for the *closing sound*

```
"sound_effects": [
  {
    "id": "closing_sound",
    "filename": "quote_noooooo.ogg",
    "audio_channel_id": 2,
    "mute": false
  },
```

Important: Don't change the `id` property for `songs` and `sound_effects` objects because it is used to uniquely identify them.

See also:

- The setting `quotes`
- The setting `songs`
- The setting `sound_effects`
- The setting `sounds_directory`

3.9 Change slot LEDs sequence

The setting `slot_leds` in the configuration file controls the blinking pattern of the three slot LEDs in Darth Vader's control box.

To change the default sequence, open the configuration file:

```
$ start_dv -e cfg
```

The `slot_leds` object defines the property `sequence` which can take a string value (`'action'` or `'calm'`) or a custom sequence.

The custom sequence consists of a list of LED labels { `'top'`, `'middle'`, `'bottom'` } arranged in a sequence specifying the order the slot LEDs should turn ON/OFF.

Listing 9: **Example:** a `slot_leds` object with the **calm** sequence

```
"slot_leds":{
  "delay_between_steps": 0.5,
  "time_per_step": 1,
  "sequence": "calm"
},
```

Listing 10: **Example:** a `slot_leds` object with a **custom** sequence

```
"slot_leds":{
  "delay_between_steps": 0.5,
  "time_per_step": 1,
  "sequence":[
    ["top", "bottom"],
    [],
    ["middle"],
    []
  ]
},
```

This simple custom sequence will turn ON/OFF the slot LEDs in this order:

1. top + bottom LEDs turned ON
2. All LEDs turned OFF
3. middle LED turned ON
4. All LEDs turned OFF

Each step in the sequence will last for `time_per_step` seconds and there will be a delay of `delay_between_steps` seconds between each step in the sequence. And the whole sequence will keep on repeating until the script exits by pressing `ctrl + c`.

See also:

The setting `slot_leds`

3.10 Mute breathing sound

To mute Darth Vader's breathing sound which plays almost as soon as the `start_dv` script runs, edit the setting `sound_effects` in the configuration file which can be opened with:

```
$ start_dv -e cfg
```

Set the `breathing_sound` object's `mute` to `false`.

Listing 11: **Example:** Mute the breathing sound

```
"sound_effects": [
  {
    "id": "breathing_sound",
    "name": "Breathing sound",
    "filename": "darth_vader_breathing.ogg",
    "audio_channel_id": 0,
    "mute": false,
    "loops": -1
  }
]
```

(continues on next page)

(continued from previous page)

```
}  
]
```

See also:

- The setting `sound_effects`
- *Change channel volume*

3.11 Run the script as quiet or verbose

To run the `start_dv` script as quiet or verbose, open the configuration file with:

```
$ start_dv -e cfg
```

And set the setting `quiet` or `verbose` to `true`.

When running the `start_dv` script as `verbose`, the logging level is set to `DEBUG`. Thus, all messages will be displayed and when there is an exception, the traceback will be shown.

On the other hand, when running the `start_dv` script as `quiet`, nothing will be printed to the terminal, not even error messages. However, you will still be able to hear sounds and interact with the push buttons or keyboard.

Important: if `quiet` and `verbose` are both activated at the same time, only `quiet` will have an effect.

See also:

- The setting `quiet`
- The setting `verbose`

API REFERENCE

- `darth_vader_rpi.start_dv`
 - *Usage*
- `darth_vader_rpi.darth_vader`
- `darth_vader_rpi.ledutils`
- `darth_vader_rpi.utils`

4.1 `darth_vader_rpi.start_dv`

Script to turn on LEDs and play sound effects on a Raspberry Pi (RPI).

The LEDs illuminate a Darth Vader action figure's lightsaber and the three slots in the chest control box. 3 push buttons control the following sounds and LEDs:

1. Some of his famous quotes
2. The Imperial march theme song
3. The lightsaber drawing, hum and retraction sounds
4. The lightsaber illumination (3 LEDs)

His iconic breathing sound plays in the background indefinitely almost as soon as the RPi is run with the script.

The script allows you also to edit the [main config file](#) to setup among other things the RPi's GPIO pins connected to LEDs and push buttons.

By default the [RPi.GPIO](#) module is used, but if the simulation option (`-s`) is used with the `start_dv` script, then the [SimulRPi.GPIO](#) module will be used instead which simulates [RPi.GPIO](#) for those that don't have an RPi to test on.

4.1.1 Usage

Once the `darth_vader_rpi` package is installed, you should have access to the `start_dv` script:

```
start_dv [-h] [--version] [-q] [-s] [-v] [-e {log,main}] [-a APP]
```

Run the script on the **RPi** with default values for the GPIO channels and other settings:

```
$ start_dv
```

Run the script on your **computer** using `SimulRPi.GPIO` which simulates `RPi.GPIO`:

```
$ start_dv -s
```

Edit the main config file with `TextEdit` (e.g. on macOS):

```
$ start_dv -e main -a TextEdit
```

Edit the logging config file with a default application (e.g. `atom`):

```
$ start_dv -e log
```

Notes

More information is available at:

- [Darth-Vader-RPi GitHub](#)
- [SimulRPi GitHub](#)

Note: In `darth_vader`, `ledutils`, and `start_dv`, the default value for GPIO is `None` and will be eventually set to one of the two modules (`RPi.GPIO` or `SimulRPi.GPIO`) depending on the user's settings.

`RPi.GPIO` provides a class to control the GPIO pins on a Raspberry Pi.

If the *simulation* option (`-s`) is used with the `start_dv` script, the `SimulRPi.GPIO` module will be used instead.

`start_dv.edit_config(cfg_type, app=None)`

Edit a configuration file.

The user chooses what type of config file (`cfg_type`) to edit: 'log' for the [logging config file](#) and 'main' for the [main config file](#).

The configuration file can be opened by a user-specified application (`app`) or a default program associated with this type of file (when `app` is `None`).

Parameters

- **cfg_type**(*str*, {'log', 'main'}) – The type of configuration file we want to edit. 'log' refers to the [logging config file](#), and 'main' to the [main config file](#) used to setup the Darth-Vader-RPi project such as specifying the sound effects or the GPIO channels.
- **app**(*str*, optional) – Name of the application to use for opening the config file, e.g. `TextEdit` (the default value is `None` which implies that the default application will be used to open the config file).

Returns `retcode` – If there is a [subprocess](#) -related error, the return code is non-zero. Otherwise, it is 0 if the file can be successfully opened with an external program.

Return type `int`

`start_dv.main()`

Main entry-point to the script.

According to the user's choice of action, the script might:

- activate a Darth Vader figurine (turn on LEDs and play sound effects)
- edit a configuration file

Raises `ValueError` – Raised if an invalid configuration name is given to the command-line argument *edit*.

Notes

Only one action at a time can be performed.

`start_dv.setup_argparser()`

Setup the argument parser for the command-line script.

The important actions that can be performed with the script are:

- activate a Darth Vader figurine (turn on LEDs and play sound effects)
- edit a configuration file

Returns `parser` – Argument parser.

Return type `argparse.ArgumentParser`

4.2 `darth_vader_rpi.darth_vader`

Module for activating a Darth Vader figurine by turning on LEDs on his suit and playing sounds, all done via a Raspberry Pi (RPi).

The LEDs illuminate Darth Vader's lightsaber and the three slots in the chest control box. 3 push buttons control the following sounds and LEDs:

1. Some of his famous quotes
2. The Imperial march theme song
3. The lightsaber drawing, hum and retraction sounds
4. The lightsaber illumination (3 LEDs)

His iconic breathing sound plays in the background indefinitely almost as soon as the RPi is run with the script.

class `darth_vader.DarthVader` (*main_cfg*)

Bases: `object`

Class for activating a Darth Vader figurine by turning on LEDs on his suit and playing sounds, all done via a Raspberry Pi (RPi).

The *main config file* is used to setup the *start_dv* script, such as the GPIO pins and the sound files.

Parameters `main_cfg` (*dict*) – Dictionary containing the configuration data to setup the *start_dv* script, such as the GPIO pins and the sound files. See *main config file* for a detailed look into its content.

Variables `th_slot_leds` (*start_dv.ExceptionThread*) – Thread responsible for turning on the three slot LEDs in a precise sequence.

Its target function is `ledutils.turn_on_slot_leds()`.

activate ()

Activate a Darth Vader figurine by turning on LEDs on his suit and playing sounds, all done via an RPi.

While the method waits for a pressed button, you can exit by pressing `ctr + c`.

Returns

retcode – If the method is run without any `Exception`, the return code is 0. Otherwise, it is 1.

Also, even if there is an `Exception`, the method will try to clean up before exiting.

Return type `int`

cleanup (*gpio_channels*)

Clean up any resources such as threads and GPIO channels.

The cleanup consists in the following actions:

- turn off each LED
- stop the thread `th_slot_leds`
- stop each audio channel
- call `RPi.GPIO.cleanup()` which will return all GPIO channels back to inputs with no pull up/down
 - If in simulation mode, `SimulRPi.GPIO.cleanup` is called to stop the threads among other things

Parameters `gpio_channels` (*dict*) – Dictionary mapping channel id (*str*) to channel attributes (*dict*). The channel attributes consist in the following:

- `channel_number`
- `channel_name`
- `key`
- `led_symbols`

Note: These channel attributes are those found in the setting `gpio_channels` from the main configuration file.

class `darth_vader.ExceptionThread` (*verbose=False, *args, **kwargs*)

Bases: `threading.Thread`

A subclass from `threading.Thread` that defines threads that can catch errors if their target functions raise an exception.

Parameters

- **verbose** (*bool, optional*) – If *True*, print the traceback when there is an exception. Otherwise, print just a one-line error message, e.g. `KeyError: 'test'`
- **args** (*tuple, optional*) – Positional arguments given to the thread's target function.
- **kwargs** (*dict, optional*) – Keyword arguments given to the thread's target function.

Variables `exc` (`Exception`) – Represents the exception raised by the target function.

References

- [stackoverflow](#)

`run()`

Method representing the thread's activity.

Overridden from the base class `threading.Thread`. This method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

It also saves and logs any error that the target function might raise.

4.3 `darth_vader_rpi.ledutils`

Collection of LEDs-related utilities for the *Darth-Vader-RPi* project.

`ledutils.turn_off_led(channel)`

Turn off a LED from a given channel.

Parameters `channel` (`int`) – Channel number associated with a LED which will be turned off.

`ledutils.turn_on_led(channel)`

Turn on a LED from a given channel.

Parameters `channel` (`int`) – Channel number associated with a LED which will be turned on.

`ledutils.turn_on_slot_leds(top_led, middle_led, bottom_led, leds_sequence='action', delay_between_steps=0.5, time_per_step=0.5)`

A thread's **target function** that turn on/off the three slot LEDs in a precise sequence.

These three LEDs are associated with Darth Vader's three slots located on his chest control box. These LEDs are labeled as 'top', 'middle', and 'bottom', respectively.

The three LEDs are turned on according to a default or custom sequence which repeats itself. The accepted values for `leds_sequence` are 'action' and 'calm' which represent Darth Vader's physiological state as a sequence of LEDs blinking in a particular order.

The user can also provide its own `leds_sequence` by using a list of LED labels {'top', 'middle', 'bottom'} arranged in a sequence specifying the order the slot LEDs should turn on/off, e.g. [['top', 'bottom'], [], ['middle'], []] will turn on/off the slot LEDs in this order:

```
1. top + bottom LEDs turned on
2. All LEDs turned off
3. middle LED turned on
4. All LEDs turned off
```

Each step in the sequence will last for `time_per_step` seconds.

There will be a delay of `delay_between_steps` seconds between each step in the previous example.

The default sequences of slot LEDs were obtained from this [YouTube video](#).

Parameters

- `top_led` (`int`) – Channel number associated with the *Top* slot LED.
- `middle_led` (`int`) – Channel number associated with the *Middle* slot LED.

- **bottom_led** (*int*) – Channel number associated with the *Bottom* slot LED.
- **leds_sequence** (*str or list, optional*) – Sequence of slot LEDs on Darth Vader’s chest box.

If `leds_sequence` is a string, then it takes on one of these values which represent Darth Vader’s physiological state: {‘action’, ‘calm’}.

If `leds_sequence` is a list, then it must be a list of slot LED labels {‘top’, ‘middle’, ‘bottom’} arranged in a sequence as to specify the order the slot LEDs should turn on/off, e.g. `[‘top’, ‘bottom’]`, `[]`, `[‘middle’]`, `[]` will turn on/off the slot LEDs in this order:

```
1. top + bottom LEDs turn on
2. All LEDs turn off
3. middle LED turn on
4. All LEDs turn off
```

- **delay_between_steps** (*float, optional*) – Delay in seconds between each step in the sequence. The default value is 0.5 second.
- **time_per_step** (*float, optional*) – Time in seconds each step in the sequence will last. The default value is 0.5 second.

Important: This also affects the time all LEDs will remain turned off if a step in `leds_sequence` is an empty list.

Important: `turn_on_slot_leds()` should be run by a thread and eventually stopped from the main program by setting its `do_run` attribute to *False* to let the thread exit from its target function.

For example:

```
th = threading.Thread(target=turn_on_slot_leds,
                      args=(leds_channels))
th.start()

# Your other code ...

# Time to stop thread
th.do_run = False
th.join()
```

4.4 darth_vader_rpi.utils

Collection of utilities for the *Darth-Vader-RPi* project.

class `utils.SoundWrapper` (*sound_id, sound_name, sound_filepath, channel_id, mute=False*)
Bases: `object`

Class that wraps around `pygame.mixer.Channel` and `pygame.mixer.Sound`.

The `__init__` method takes care of automatically loading the sound file. The sound file can then be played or stopped from the specified channel `channel_id` with the `play()` or `stop()` method, respectively.

Parameters

- **sound_id** (*str*) – A unique identifier.
- **sound_name** (*str*) – Name of the sound file that will be displayed in the terminal.
- **sound_filepath** (*str*) – Path to the sound file.
- **channel_id** (*int*) – Channel id associated with an instance of `pygame.mixer.Channel` for controlling playback. It must take an `int` value starting from 0.
- **mute** (*bool*, *optional*) – If set to *True*, the sound will not be played. The default value is *False*.

Note: It is a wrapper with a very minimal interface to `pygame.mixer.Channel` where only two methods `play()` and `stop()` are provided for the sake of the project.

play (*loops=0*)

Play a sound on the specified Channel `channel_id`.

Parameters **loops** (*int*) – Controls how many times the sample will be repeated after being played the first time. The default value (zero) means the sound is not repeated, and so is only played once. If `loops` is set to -1 the sound will loop indefinitely (though you can still call `stop()` to stop it).

Reference: `pygame.mixer.Sound.play()`

stop ()

Stop playback on the specified channel `channel_id`.

`utils.add_spaces_to_msg` (*msg*, *nb_spaces=60*)

Add spaces at the end of a message.

Parameters

- **msg** (*str*) – Message to be updated with spaces at the end.
- **nb_spaces** (*int*) – Number of spaces to add at the end of the message. The default value is 60.

Returns **message** – The updated message with spaces added at the end.

Return type *str*

`utils.dumps_json` (*filepath*, *data*, *encoding='utf8'*, *ensure_ascii=False*, *indent=None*, *sort_keys=False*)

Write data to a JSON file.

The data is first serialized to a JSON formatted string and then saved to disk.

Parameters

- **filepath** (*str*) – Path to the JSON file where the data will be saved.
- **data** – Data to be written to the JSON file.
- **encoding** (*str*, *optional*) – Encoding to be used for opening the JSON file in write mode (the default value is *'utf8'*).
- **ensure_ascii** (*bool*, *optional*) – If `ensure_ascii` is *False*, then the return value can contain non-ASCII characters if they appear in strings contained in `data`. Otherwise, all such characters are escaped in JSON strings. See the `json.dumps` docstring description (the default value is *False*).
- **indent** (*int or None*, *optional*) – If `indent` is a non-negative integer, then JSON array elements and object members will be pretty-printed with that indent level. An

indent level of 0 will only insert newlines. `None` is the most compact representation. See the `json.dumps()` docstring description. (the default value is `None`).

- **sort_keys** (*bool, optional*) – If `sort_keys` is `True`, then the output of dictionaries will be sorted by key. See the `json.dumps` docstring description. (the default value is `False`).

Raises `OSError` – Raised if any I/O related error occurs while writing the data to disk, e.g. the file doesn't exist.

`utils.get_cfg_dirpath()`

Get the path to the directory containing the config files.

Returns `dirpath` – The path to the directory containing the config files.

Return type `str`

`utils.get_cfg_filepath(file_type)`

Get the path to a config file used by the script `start_dv`.

`file_type` accepts the following values:

- **default_log**: refers to the `default logging configuration file` used to setup the logging for all custom modules.
- **default_main**: refers to the `default main configuration file` used to setup the script `start_dv`.
- **log**: refers to the user-defined logging configuration file which is used to setup the logging for all custom modules.
- **main**: refers to the user-defined main configuration file used to setup the script `start_dv`.

Parameters `file_type` (`str`, `{'default_log', 'default_main', 'log', 'main'}`) – The type of config file for which we want the path.

Returns `filepath` – The path to the config file.

Return type `str`

Raises `AssertionError` – Raised if the wrong type of config file is given to the function. Only `{'default_log', 'default_main', 'log', 'main'}` are accepted for `file_type`.

`utils.load_json(filepath, encoding='utf8')`

Load JSON data from a file on disk.

If using Python version between 3.0 and 3.6 (inclusive), the data is returned as `collections.OrderedDict`. Otherwise, the data is returned as `dict`.

Parameters

- **filepath** (*str*) – Path to the JSON file which will be read.
- **encoding** (*str, optional*) – Encoding to be used for opening the JSON file in read mode (the default value is `'utf8'`).

Returns `data` – Data loaded from the JSON file.

Return type `dict` or `collections.OrderedDict`

Raises `OSError` – Raised if any I/O related error occurs while reading the file, e.g. the file doesn't exist.

References

Are dictionaries ordered in Python 3.6+? (stackoverflow)

`utils.override_config_with_args` (*config*, *parser*)

Override a config dictionary with arguments from the command-line.

Parameters

- **config** (*dict*) – Dictionary containing configuration options.
- **parser** (*argparse.ArgumentParser*) – Argument parser.

Returns

retval – Contains two lists:

1. *args_not_found*: saves command-line arguments not found in the config dictionary
2. *config_opts_overridden*: saves config options overridden by command-line arguments as a three-tuple (option name, old value, new value)

Return type `collections.namedtuple`

`utils.run_cmd` (*cmd*)

Run a shell command with arguments.

The shell command is given as a string but the function will split it in order to get a list having the name of the command and its arguments as items.

Parameters **cmd** (*str*) – Command to be executed, e.g.

```
open -a TextEdit text.txt
```

Returns **retcode** – Returns code which is 0 if the command was successfully completed. Otherwise, the return code is non-zero.

Return type `int`

Raises **FileNotFoundError** – Raised if the command *cmd* is not recognized, e.g. `$ TextEdit {filepath}` since *TextEdit* is not an executable.

CHANGELOG

5.1 Version 0.1.0a0

September 15, 2020

- Initial release
- Tested the `start_dv` script on various platforms and environments.

Here are the results:

- macOS:
 - * The `start_dv -s` command runs without errors on Python 3.7 and 3.8
 - * On Python 3.5 and 3.6, I had to set my locale settings to `LANG="en_US.UTF-8"` to make the `start_dv -s` command work. Thus, it was not an error with the script but with how my system environment was setup. Python 3.5 and 3.6 don't assume **UTF-8** based local settings like the other more recent Python versions.
- Raspberry Pi (Python 3.5):
 - * Running the `start_dv` command without errors.
 - * Running the `start_dv -s` command without errors.
- SSH from macOS to RPi (Python 3.5):
 - * Running the `start_dv` command without errors.
 - * Running the `start_dv -s` command produces a warning about `pynput` not being able to be imported (as expected) but the rest of the code that doesn't depend on keyboard keys being detected works, i.e. blinking of LED symbols in the terminal.

Note: For more detailed information about these tests, check [Test results for start_dv](#)

LICENSE: GPL3

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <<https://fsf.org/>>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for
software and other kinds of works.

The licenses for most software and other practical works are designed
to take away your freedom to share and change the works. By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users. We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors. You can apply it to
your programs, too.

When we speak of free software, we are referring to freedom, not
price. Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights. Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received. You must make sure that they, too, receive
or can get the source code. And you must show them these terms so they
know their rights.

Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains

(continues on next page)

(continued from previous page)

that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

(continues on next page)

(continued from previous page)

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited

(continues on next page)

(continued from previous page)

permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

(continues on next page)

(continued from previous page)

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord

(continues on next page)

(continued from previous page)

with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a

(continues on next page)

(continued from previous page)

requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

(continues on next page)

(continued from previous page)

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a

(continues on next page)

(continued from previous page)

covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone

(continues on next page)

(continued from previous page)

to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this

(continues on next page)

(continued from previous page)

License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided

(continues on next page)

(continued from previous page)

above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program. If not, see <https://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see [<https://www.gnu.org/licenses/>](https://www.gnu.org/licenses/).

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with

(continues on next page)

(continued from previous page)

the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read [<https://www.gnu.org/licenses/why-not-lgpl.html>](https://www.gnu.org/licenses/why-not-lgpl.html).

INDICES AND TABLES

- `genindex`
- `modindex`

PYTHON MODULE INDEX

d

`darth_vader`, 33

l

`ledutils`, 35

s

`start_dv`, 31

u

`utils`, 36

INDEX

A

`activate()` (*darth_vader.DarthVader method*), 34
`add_spaces_to_msg()` (*in module utils*), 37

C

`cleanup()` (*darth_vader.DarthVader method*), 34

D

`darth_vader`
 module, 33
`DarthVader` (*class in darth_vader*), 33
`dumps_json()` (*in module utils*), 37

E

`edit_config()` (*in module start_dv*), 32
`ExceptionThread` (*class in darth_vader*), 34

G

`get_cfg_dirpath()` (*in module utils*), 38
`get_cfg_filepath()` (*in module utils*), 38

L

`ledutils`
 module, 35
`load_json()` (*in module utils*), 38

M

`main()` (*in module start_dv*), 33
module
 darth_vader, 33
 ledutils, 35
 start_dv, 31
 utils, 36

O

`override_config_with_args()` (*in module utils*), 39

P

`play()` (*utils.SoundWrapper method*), 37

R

`run()` (*darth_vader.ExceptionThread method*), 35
`run_cmd()` (*in module utils*), 39

S

`setup_argparser()` (*in module start_dv*), 33
`SoundWrapper` (*class in utils*), 36
`start_dv`
 module, 31
`stop()` (*utils.SoundWrapper method*), 37

T

`turn_off_led()` (*in module ledutils*), 35
`turn_on_led()` (*in module ledutils*), 35
`turn_on_slot_leds()` (*in module ledutils*), 35

U

`utils`
 module, 36