

# Finch Python Package Description

---

## Description of Files

**finch.py** – Contains the Finch API functions for controlling Finch

**finchconnection.py** – Contains low level functions for sending and receiving data over USB, used by finch.py

**hidapi32/64.dll, libhidapi.dylib, libhidapi32/64.so, libhidapi.so** – compiled C libraries used by finchconnection.py to scan for USB devices (specifically, the Finch) and open a connection. **dll** files are for Windows, **dylib** is for OSX, and **so** is for Linux.

**alarm.py, lapswimmer.py, musicexample.py, racedriver.py, testfinchfunctions.py** – Example Python programs, open them for more information on what each one does.

**notes.py** – helper class used by musicexample.py to simplify playing lots of notes on the Finch buzzer.

## Finch API

### General

**close** – closes the connection to the Finch

Usage: call this when your program is exiting to cleanly close the Finch connection. This will shut off the motors and then send the Finch back to idle mode (color changing)

**halt** – convenience function to shut off the Finch LED and finch motors

Usage: call whenever you wish the Finch to stop moving and turn off its LED with one function call. Will not shut off the buzzer.

### Outputs

**led** – controls the Finch beak color

Usage: hex triplet string: `finch.led('#0000FF')` or 0-255 RGB values: `finch.led(0, 0, 255)`. The first value indicates the red intensity, second is green, third is blue.

**wheels** – Controls the power sent to the left and right wheels

Usage: values must range from -1.0 to 1.0 for each wheel, use `left=right=0.0` to stop. Examples:

```
finch.wheels(1.0, 1.0) # full forward
```

```
finch.wheels(-1.0, -1.0) # full reverse
```

```
finch.wheels(0.7, -0.3) # 70% forward on left wheel, 30% reverse on right wheel
```

**buzzer** – outputs sound over the Finch buzzer

Usage: pass two parameters, duration in seconds followed by frequency in Hertz. Frequencies between 20 and 20,000 are audible. Example (plays 440 Hz note for ½ second): `finch.buzzer(0.5, 440)`.

Note that this function does not delay program execution; to delay, use **buzzer\_with\_delay**. If two buzzer functions are called one after the other, only the second will play. For example, the following will only play a single 880 Hz note:

```
finch.buzzer(0.5, 220)
finch.buzzer(0.5, 880)
```

**buzzer\_with\_delay** – outputs sound over the Finch buzzer, then blocks the program for delay equal to the duration of the note. Same usage as **buzzer**.

## Sensors

**temperature** – returns temperature in degrees Celcius.

Usage: `my_temperature = finch.temperature()`

**light** – returns the left and right light sensor readings, values range from 0.0 (dark) to 1.0 (bright)

Usage: `left_light_sensor, right_light_sensor = finch.light()`

**obstacle** – returns obstacle sensor readings, values are False for no obstacle, True for obstacle detected

Usage: `left_obstacle, right_obstacle = finch.obstacle()`

**acceleration** – returns (x, y, z, tap, shake).

Usage: x, y, and z, are the acceleration readings in units of G's, and range from -1.5 to 1.5. When the finch is horizontal, z is close to 1, x, y close to 0. When the finch stands on its tail, y, z are close to 0, x is close to -1. When the finch is held with its left wing down, x, z are close to 0, y is close to 1.

tap, shake are boolean values -- true if the corresponding event has happened.