
HEYKUBE

Release v0.5.0

David Garrett

Jun 28, 2021

CONTENTS

1	Features	3
1.1	Installation	3
1.2	CLI	3
1.3	Usage	4
1.4	License	8
2	Indices and tables	9
	Python Module Index	11
	Index	13

This documents the HEYKUBE python library, letting you connect and program HEYKUBE

<http://www.heykube.com>

- Free software: MIT license
- Documentation: <https://heykube-python.readthedocs.io>.

FEATURES

- Currently only supported on the Raspberry Pi platform - TBD others

Contents:

1.1 Installation

1.1.1 Stable release

To install the HEYKUBE library, run this command in your terminal:

```
$ pip3 install heykube
```

This is the preferred method to install HEYKUBE, as it will always install the most recent stable release.

1.1.2 From sources

The sources for bleak can be downloaded from the Github repo

You can either clone the public repository:

```
$ git clone git://github.com/heykube/heykube-python
```

Once you have a copy of the source, you can install it with:

```
$ cd heykube-python  
$ pip3 install -e .
```

1.2 CLI

Note: HEYKUBE is still in beta, with lots of work to clean it up. Check the examples directory, or run the command line interface

This code shows how to run the command line interface (CLI) for HEYKUBE

```
$ cd scripts  
$ ./heykube_cli.py
```

The CLI initially scans for HEYKUBEs, and you can connect

```
Starting HEYKUBE Command line interface (CLI)
Scanning for HEYKUBEs
  HEYKUBE-28F1 : addr FC:AE:7C:F7:28:F1 at -56 dB RSSI
HEYKUBE> connect HEYKUBE-28F1
HEYKUBE-28F1> check_version
Software version: v0.99
BTLE disconnect: Remote User Terminated Connection
HEYKUBE-28F1> help

Documented commands (type help <topic>):
=====
check_battery    disconnect      help           prompt_face    write_instructions
check_version    enable_sounds  hints_off      quit
connect         get_instructions hints_on       reset
debug_level     get_moves      play_sound     scan
disable_sounds  get_orientation print_cube     track_cube

HEYKUBE-28F1>
```

1.3 Usage

Note: HEYKUBE is still in beta, with lots of work to clean it up. Check the examples directory, or run the command line interface

class `heykube.heykube`

Defines the HEYKUBE class

Includes ability to connect/disconnect from HEYKUBEs Program lights and sounds Send custom instructions
Query the cube state, register for moves and notifications

append_instructions(*instr_moves*)

Appends more instructions to the instructions queue

clear_instructions()

Clears the instructions queue, and returns to the internal solver

clear_notify()

Clear out old notify messages we do not need

connect(*device*)

Connects to a specified HEYKUBE device *device* is a BLEDevice from bleak, and should be used from
`get_device`

disable_match()

Disables the match from firing

disable_notifications()

Disables BTLE notifications

disable_sounds()

This method temporarily disables the sounds from the cube during the duration of the BTLE connection
session

disconnect()

Disconnects from a HEYKUBE device and clean-up connection

enable_match()

Enables the match to fire again since it disable after each match

enable_notifications(*notify_list*)

Registers for notifications from HEYKUBE

enable_pattern(*pattern*)

If HEYKUBE is solved, enables instructions for the specified pattern

enable_sounds(*major_sound=True, minor_sound=True*)

Reenables HEYKUBE sounds if they were previously disabled

flash_all_leds()

This method flashes all the LEDs on the HEYKUBE

get_device()

Scans input args and finds a HEYKUBE for connection Defines the HEYKUBE connection options

optional arguments:

- h, --help** show this help message and exit
- verbose** increase output verbosity
- n NAME, --name NAME** Directly defines name of a HEYKUBE for connection
- a ADDRESS, --address ADDRESS** Directly defines an HEYKUBE MAC address for connection
- s, --scan** Scans and reports all the available HEYKUBES
- d, --debug** Turns on debug prints

get_notify()

Get immediate notification from the queue if it is there, otherwise returns None

get_pattern_name(*index*)

Returns a pattern name for a given index

get_pattern_names()

Returns all the pattern names

get_seq_num()

Reads the current sequence number from the cube

get_timestamp()

Reads the current timestamp from the cube

initialize()

This method resets the internal state of the HEYKUBE back to the solved state

is_solved()

This method checks if the HEYKUBE is in the solved state

Returns bool – Returns True if the cube is solved

light_led(*led_index*)

This method will manually light one of the LEDs on the cube

Parameters

- **face (*int.*)** – Picks one of 6 faces to light up

- **index** – Picks 1 of 6 indexes

play_sound(*select=0*)

This method plays a sound on the HEYKUBE device

Parameters **select** – Selects the sound index between 0-7

print_cube()

This method reads the current state of HEYKUBE and prints to the screen

read_accel()

This method returns the full orientation of the cube in space using the on-board 3D-accelerometer

Returns which face is up, along with X,Y and Z acceleration vector

read_battery()

Reads the battery status and charging state

read_instructions()

This method connects to the HEYKUBE and reads-out the current list of instructions that are currently in queue

Returns str – Returns the list of rotations

read_moves(*prev_seq_num=None*)

Reads up to the last 42 moves from HEYKUBE

read_status()

This method reads up to the last 3 status events registered in the HEYKUBE

Returns

list – Returns a list of the up to last 3 status events, None is status is empty status_dict :

'solution' ['scrambed:x' | 'bottom_cross:x' | 'bottom_layer:x' |]

'middle_layer:x' | 'top_layer_cross:x' | 'top_layer_face:x' | 'top_layer_corner:x'

| 'solved:0' - where x is [0-3],

'last_move' : 'o|O|w|W|r|R|y|Y|b|B|g|G', 'timestamp' : <running time in secs>,

'match' : True, 'instruction_empty' : True, 'instruction_max' : True, 'seq_num' : [0-255]}

read_version()

Reads the current SW version

send_hint(*index*)

This method plays a hint on the faces

send_prompt(*index*)

This method flashes the LEDs on the HEYKUBE, typically used when the user solves the cube

set_match(*match, enable=True*)

This method allows the user to set the match from the class Match object

software_reset()

This method issues a software reset through BTLE

turn_hints_off()

Turns hints off on HEYKUBE - they will return once solved

turn_hints_on()

Turns HEYKUBE hints back on

turn_off_led()

This method turns off all the LEDs on the HEYKUBE

wait(timeout=10)

This method is used to wait for specified time

wait_for_cube_state(prev_seq_num=None, timeout=10)

This method is used to wait for events from the HEYKUBE and includes a timeout mechanism if the event never happens

Parameters **timeout** (*float.*) – Specifies the timeout duration in seconds

Returns

dict – Returns a dictionary with notifications events, None is status is empty

'solution' ['scrambed:x' | 'bottom_cross:x' | 'bottom_layer:x' |

'middle_layer:x' | 'top_layer_cross:x' | 'top_layer_face:x' | 'top_layer_corner:x'
| 'solved:0' - where x is [0-3],

'last_move' : 'o|O|w|W|r|R|y|Y|b|B|g|G', 'timestamp' : <running time in secs>,
'match' : True, 'double_tap' : True, 'charger' : True, 'instruction_empty' : True,
'instruction_max' : True, 'seq_num' : [0-255]

wait_for_notify(prev_seq_num=None, timeout=10)

This method is used to wait for events from the HEYKUBE and includes a timeout mechanism if the event never happens

Parameters **timeout** (*float.*) – Specifies the timeout duration in seconds

Returns

dict – Returns a dictionary with notifications events, None is status is empty

'solution' ['scrambed:x' | 'bottom_cross:x' | 'bottom_layer:x' |] 'middle_layer:x' |
'top_layer_cross:x' | 'top_layer_face:x' | 'top_layer_corner:x' | 'solved:0' - where x
is [0-3],

'last_move' : 'U|L|F|R|B|D[']', 'match' : True, 'instruction_empty' : True, 'instruc-
tion_max' : True, 'seq_num' : [0-255] 'timestamp' : <running time in seconds>

write_cube_state(state)

Overrides the internal cube state –expert only

write_instructions(instr_moves, append=False)

This method allows the user to send a custom list of instructions to the HEYKUBE device. The faces on the LEDs will light up in sequence the users defines

Parameters **Moves()** (*Class*) – Holds the list of moves

class heykube.Cube**class heykube.Match(init_set=None)**

Defines the Match() class which enables HEYKUBE to match with certain patterns, and provide a notification

add_cross(face)

Sets the match for the cross on that face

add_cross_color(face_name)

Sets the match for a cross - but just colors on that face

add_face(face)

Sets the match for the colors on that face

clear()
Clears the match object back to don't care on all Facelets

solved()
Sets the match to a fully solved cube

This block shows how to connect to an initial HEYKUBE

```
import heykube
import time
import random

# Connect to the cube
cube = heykube.heykube()
device = cube.get_device()

# If not found, just exit
if device is None:
    exit()

# Run the connection
if not cube.connect(device):
    print('Failed to connect with a HEYKUBE')
    exit()
```

1.4 License

Copyright 2021 22nd Solutions, LLC

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

h

heykube, [4](#)

A

add_cross() (*heykub.Match method*), 7
 add_cross_color() (*heykub.Match method*), 7
 add_face() (*heykub.Match method*), 7
 append_instructions() (*heykub.heykub method*), 4

C

clear() (*heykub.Match method*), 7
 clear_instructions() (*heykub.heykub method*), 4
 clear_notify() (*heykub.heykub method*), 4
 connect() (*heykub.heykub method*), 4
 Cube (*class in heykub*), 7

D

disable_match() (*heykub.heykub method*), 4
 disable_notifications() (*heykub.heykub method*), 4
 disable_sounds() (*heykub.heykub method*), 4
 disconnect() (*heykub.heykub method*), 4

E

enable_match() (*heykub.heykub method*), 5
 enable_notifications() (*heykub.heykub method*), 5
 enable_pattern() (*heykub.heykub method*), 5
 enable_sounds() (*heykub.heykub method*), 5

F

flash_all_lights() (*heykub.heykub method*), 5

G

get_device() (*heykub.heykub method*), 5
 get_notify() (*heykub.heykub method*), 5
 get_pattern_name() (*heykub.heykub method*), 5
 get_pattern_names() (*heykub.heykub method*), 5
 get_seq_num() (*heykub.heykub method*), 5
 get_timestamp() (*heykub.heykub method*), 5

H

heykub
 module, 4

heykub (*class in heykub*), 4

I

initialize() (*heykub.heykub method*), 5
 is_solved() (*heykub.heykub method*), 5

L

light_led() (*heykub.heykub method*), 5

M

Match (*class in heykub*), 7
 module
 heykub, 4

P

play_sound() (*heykub.heykub method*), 6
 print_cube() (*heykub.heykub method*), 6

R

read_accel() (*heykub.heykub method*), 6
 read_battery() (*heykub.heykub method*), 6
 read_instructions() (*heykub.heykub method*), 6
 read_moves() (*heykub.heykub method*), 6
 read_status() (*heykub.heykub method*), 6
 read_version() (*heykub.heykub method*), 6

S

send_hint() (*heykub.heykub method*), 6
 send_prompt() (*heykub.heykub method*), 6
 set_match() (*heykub.heykub method*), 6
 software_reset() (*heykub.heykub method*), 6
 solved() (*heykub.Match method*), 8

T

turn_hints_off() (*heykub.heykub method*), 6
 turn_hints_on() (*heykub.heykub method*), 6
 turn_off_led() (*heykub.heykub method*), 6

W

wait() (*heykub.heykub method*), 7
 wait_for_cube_state() (*heykub.heykub method*), 7

`wait_for_notify()` (*heykube.heykube method*), [7](#)
`write_cube_state()` (*heykube.heykube method*), [7](#)
`write_instructions()` (*heykube.heykube method*), [7](#)