pyhula

hula python dependency package. A Python pack used by hula.

Python version: 3.6.7

Installation

Execute the following commands in the terminal to install pyhula. Two installation methods are available. Input the following code in powershell(cmd.exe) to install pyhula.

```
pip install pyhula
pip install pyhula-1.0.4-cp36-cp36m-win_amd64.whl
```

Checking version

- Execute `pip list` in the terminal to check. Input "pip list" in powershell(cmd.exe) to get pyhula's version
- Execute in the program Using the following code.

```
import pyhula
ver = pyhula.get_version()
print(ver)
```

Usage

Use the following code to obtain a UserApi instance. You can then control the hula drone through the interfaces provided by UserApi. For interface specifications, please refer to the `doc/html/Chinese/index.html` file. Use the following codes to create a userApi instance. Its interfaces can be used to control fylo plane. Go to `doc/html/English/index.html` to see the interface specification.

```
import pyhula
api = pyhula.UserApi()
if not api.connect():
    print("connect error")
else:
    print('connection to station by wifi')
```

`api.single_fly_takeoff()` #Takeoff `api.single_fly_touchdown()` #Landing

Interface Description

Connecting to Drone

`connect(server_ip)`

... *Description:* Connects to the drone. *Parameters:* * `server_ip` : Drone IPv4 address (optional). If not specified, it will be automatically obtained. *Return Value:* * `True` : Success * `False` : Failure *Example:* `python api.connect('192.168.1.118') api.connect()`

Takeoff

`single_fly_takeoff(led)`

... *Description:* Real-time drone takeoff control. *Parameters:* * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_takeoff() api.single_fly_takeoff({'r':16,'g':15,'b':100,'mode':1})`

Landing

`single_fly_touchdown(led)`

... *Description:* Real-time drone landing control. *Parameters:* * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_touchdown() api.single_fly_touchdown({'r':16,'g':15,'b':100,'mode':1})`

Hover

`single_fly_hover_flight(time, led)`

... *Description:* Drone hovers. *Parameters:* * `time` : Hover time (seconds). *Example:* `python api.single_fly_hover_flight(10) api.single_fly_hover_flight(10,{'r':16,'g':15,'b':100,'mode':1})`

Move Forward

`single_fly_forward(distance, led)`

... *Description:* Real-time control of the drone moving forward. *Parameters:* * `distance` : Flight distance (cm). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_forward(100) api.single_fly_forward(100,{'r':16,'g':15,'b':100,'mode':1})`

Move Backward

`single_fly_back(distance, led)`

... *Description:* Real-time control of the drone moving backward. *Parameters:* * `distance` : Flight distance (cm). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_back(100) api.single_fly_back(100,{'r':16,'g':15,'b':100, 'mode':1})`

Move Left

`single_fly_left(distance, led)`

... *Description:* Real-time control of the drone moving left. *Parameters:* * `distance` : Flight distance (cm). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_left(100) api.single_fly_left(100,{'r':16,'g':15,'b':100,'mode':1})`

Move Right

`single_fly_right(distance,led)`

... *Description:* Real-time control of the drone moving right. *Parameters:* * `distance` : Flight distance (cm). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_right(100) api.single_fly_right(100,{'r':16,'g':15,'b':100,'mode':1})`

Move Up

`single_fly_up(distance,led)`

... *Description:* Real-time control of the drone moving up. *Parameters:* * `height` : Flight height (cm). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_up(100) api.single_fly_up(100,{'r':16,'g':15,'b':100,'mode':1})`

Move Down

`single_fly_down(distance, led)`

... *Description:* Real-time control of the drone moving down. *Parameters:* * `height` : Flight height (cm). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_down(100) api.single_fly_down(100,{'r':16,'g':15,'b':100, 'mode':1})`

Rotate Left

`single_fly_turnleft(angle, led)`

... *Description:* Real-time control of the drone rotating left. *Parameters:* * `angle` : Rotation angle (degrees). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_turnleft(90) api.single_fly_turnleft(90,{'r':16,'g':15,'b':100,'mode':1})`

Rotate Right

`single_fly_turnright(angle, led)`

... *Description:* Real-time control of the drone rotating right. *Parameters:* * `angle` : Rotation angle (degrees). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_turnright(90) api.single_fly_turnright(90,{'r':16,'g':15,'b':100,'mode':1})`

Bounce

`single_fly_bounce(frequency, height,led)`

... *Description:* Real-time drone bounce control. *Parameters:* * `frequency` : Number of bounces. * `height` : Bounce distance (cm). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_bounce(3, 50) api.single_fly_bounce(3, 50,{'r':16,'g':15,'b':100,'mode':1})`

Straight Flight

`single_fly_straight_flight( x, y, z, led)`

... *Description:* Straight flight (x,y,z). *Parameters:* * `x` : Coordinate x (cm). * `y` : Coordinate y (cm). * `z` : Coordinate z (cm). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_straight_flight(100, 100, 100) api.single_fly_straight_flight(100, 100, 100, {'r':16,'g':15,'b':100,'mode':1})`

Circular Flight

`single_fly_radius_around(radius,led)`

... *Description:* Circular flight with a radius. *Parameters:* * `radius` : Radius of the circle (cm). Positive: counterclockwise; Negative: clockwise. * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_radius_around(100) api.single_fly_radius_around(100,{'r':16,'g':15,'b':100,'mode':1})`

Auto-Rotation

`single_fly_autogyration360(num,led)`

... *Description:* Auto-rotation (clockwise/counter-clockwise) for a certain number of rotations. *Parameters:* * `num` : (positive: counterclockwise; negative: clockwise). * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_autogyration360(2) api.single_fly_autogyration360(2, {'r':16,'g':15,'b':100,'mode':1})`

Roll

`single_fly_somersault(direction)`

... *Description:* Drone rolls in place (forward, backward, left, or right). *Parameters:* * `DIRECTION_FORWARD=0` * `DIRECTION_BACK=1` * `DIRECTION_LEFT=2` * `DIRECTION_RIGHT=3` * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_somersault(0) api.single_fly_somersault(0, {'r':16,'g':15,'b':100,'mode':1})`

Curvilinear Flight

`single_fly_curvilinearFlight(x, y, z, led)`

... *Description:* Curvilinear flight (x,y,z). *Parameters:* * `x` : X-axis coordinate (cm) (body left and right, positive to the right). * `y` : Y-axis coordinate (cm) (body front and back, positive to the front). * `z` : Z-axis coordinate (cm) (body up and down, positive upward). * `direction` : True: counterclockwise; False: clockwise; Default: True. * `led` : (optional) Default is 0. Format: `{'r':0,'g':0,'b':0,'mode':1}` . `r` , `g` , `b` : color range, `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Example:* `python api.single_fly_curvilinearFlight(100, 100, 0) api.single_fly_curvilinearFlight(100, 100, 0, {'r':16,'g':15,'b':100,'mode':1})`

Obstacle Avoidance On

`single_fly_barrier_aircraft(mode)`

... *Description:* Enables obstacle avoidance. *Parameters:* * `mode` : True: On; False: Off. *Example:* `python api.single_fly_barrier_aircraft(True)`

Line Following Detection

`single_fly_Line_walking(fun_id, dist, way_color)`

... *Description:* Line following detection. *Parameters:* * `fun_id = 0` : 0: Forward line following, ignoring intersections. * `dist` : Distance (cm). * `way_color` : Line color range, 0-black, 255-white. *Return Value:* * `result = 1` : Result of command execution: 0-failure, 1-success, 2-success, encountered an intersection. *Example:* `python api.single_fly_Line_walking(0, 100, 0)`

Tag Recognition

`single_fly_AiIdentifies(mode)`

... *Description:* Tag recognition. *Parameters:* * `mode` : 0-9 recognizes numeric tags 0-9; 10 recognizes the left arrow; 11 recognizes the right arrow; 12 recognizes the up arrow; 13 recognizes the down arrow; 20 ends the task; 65-90 recognizes uppercase letters A-Z. After triggering recognition, the recognition process lasts 300ms. If successful, it ends immediately. *Return Value:* * `x` : X-coordinate of the tag relative to the drone. * `y` : Y-coordinate of the tag relative to the drone. * `z` : Z-coordinate of the tag relative to the drone. * `angle` : Angle of the tag relative to the drone. * `result` : False: Recognition failed; True: Recognition succeeded. *Example:* `python api.single_fly_AiIdentifies(1)`

Optical Flow / Camera QR Code Alignment

`single_fly_Qrcode_align(mode, qr_id)`

... *Description:* Optical flow / camera QR code alignment. *Parameters:* * `qr_id` : QR code ID [0-9]. * `mode` : `mode = 0` : Optical flow alignment; `mode = 1` : Camera alignment. *Return Value:* * `result` : False: Alignment failed; True: Alignment succeeded. *Example:* `python api.single_fly_Qrcode_align(0, 1)`

Optical Flow / Camera QR Code Recognition

`single_fly_recognition_Qrcode(mode, qr_id)`

... *Description:* Optical flow / camera QR code recognition. *Parameters:* * `qr_id` : QR code ID [0-9]. * `mode` : `mode = 0` : Optical flow recognition; `mode = 1` : Camera recognition. *Return Value:* `{ result: False/True, x: distance, y: distance, z: distance, yaw: angle, qr_id: id }` *Example:* `python api.single_fly_recognition_Qrcode(0, 1)`

QR Code Tracking

`single_fly_track_Qrcode(qr_id, time)`

... *Description:* Tracks QR code [0-9] for [time] seconds. *Parameters:* * `qr_id` : QR code ID. * `time` : Tracking time. *Return Value:* * `result` : 0: Success; 1: Failure. *Example:* `python api.single_fly_track_Qrcode(1, 10)`

Color Recognition

`single_fly_getColor()`

... *Description:* Color recognition; obtains the color of a frame from the current video stream. *Parameters:* None *Return Value:* * `Mode` : 1: Start, runs one frame. * `r` , `g` , `b` : Color range. * `state` : 0: Failure; 1: Success. *Example:* `python ret = api.single_fly_getColor() # Returns: r, g, b: color range, state: 0-failure, 1-success`

Set Light Color and Mode (Non-Blocking)

`single_fly_lamplight(r, g, b, time, mode)`

... *Description:* Sets the light color and mode. *Parameters:* * `r` , `g` , `b` : Color range. * `time` : Light duration (s). * `mode` : 1/solid, 2/off, 4/RGB three-color cycle, 16/rainbow, 32/flashing, 64/breathing light. *Return Value:* * `True` : Successful execution. * `False` : Execution failed. *Example:* `python api.single_fly_lamplight(255, 0, 0, 1, 1) #Sets light color and mode`

Laser Emission

`plane_fly_generating(type, data, reserve)`

... *Description:* Laser emission. *Parameters:* * `type = 0` : Laser: 0-single shot, 1-burst, 2-start laser receiving, 3-stop laser receiving, 4-continuous burst. * `data = 10` : Laser burst frequency (times/second), range 1-14. * `reserve = 100` : Ammo, data range 1-255. *Example:* `python api.plane_fly_generating(0, 10, 100) #`

`Single shot api.plane_fly_generating(2, 10, 100) # Start laser receiving`

Laser Receiver Hit

`plane_fly_laser_receiving()`

... *Description:* Laser receiver hit. *Return Value:* * `True` : Hit. * `False` : Not hit. *Example:* `python api.plane_fly_laser_receiving()`

QR Code Positioning Switch

`Plane_cmd_switch_QR(type)`

... *Description:* QR code positioning switch. *Parameters:* * `type` : 0-Enable QR code positioning; 1-Disable QR code positioning. *Example:* `python api.Plane_cmd_switch_QR(0)`

Take Photo

`Plane_fly_take_photo()`

... *Description:* Takes a photo (requires video stream to be enabled). *Example:* `python api.Plane_fly_take_photo() # Take photo`

Start/Stop Recording

`Plane_cmd_switch_video(type)`

... *Description:* Start/stop video recording. *Parameters:* * `type` : 0: Start; 1: Stop. *Example:* `python api.Plane_cmd_switch_video(0) #Start recording`

Enable Video Stream

`Plane_cmd_swith_rtp(type)`

... *Description:* Enables/disables the video stream. *Parameters:* * `type` : 0: Enable; 1: Disable. *Example:* `python api.Plane_cmd_swith_rtp(0) # Enable video stream`

Open Video Stream Window

`single_fly_flip_rtp()`

... *Description:* Opens the video stream window (video stream must be enabled first). *Example:* `python api.single_fly_flip_rtp() # Open video stream window`

Set Main Camera Pitch Angle

`Plane_cmd_camera_angle(type, data)`

... *Description:* Sets the main camera's pitch angle. *Parameters:* * `type = 0` : Direction of rotation: 0-up, 1-down (absolute), 2 and 3 algorithm control, 4-calibration, 5-block up, 6-block down (relative). * `data = 30` : Angle of rotation: 0-90. *Example:* `python api.Plane_cmd_camera_angle(0,30) # Set main camera pitch angle`

Low-Speed Propeller Rotation (Arm)

`plane_fly_arm()`

... *Description:* Low-speed propeller rotation (arming). *Example:* `python api.plane_fly_arm() #Low-speed propeller rotation`

Stop Low-Speed Propeller Rotation (Disarm)

`plane_fly_disarm()`

... *Description:* Stops low-speed propeller rotation (disarming). *Example:* `python api.plane_fly_disarm() # Stop low-speed propeller rotation`

Get Obstacle Avoidance Information

`Plane_getBarrier()`

... *Description:* Gets obstacle avoidance information. *Return Value:* A dictionary indicating the obstacle status for each direction. True: Obstacle present; False: No obstacle. `{ 'forward': True, 'back': True, 'left': True, 'right': True }` *Example:* `python ret = api.Plane_getBarrier() # Get obstacle avoidance information`

Get Drone Battery Percentage

`get_battery()`

... *Description:* Gets the drone battery percentage. *Return Value:* An integer representing the battery percentage. *Example:* `python ret = api.get_battery() # Get drone battery percentage`

Get Drone Coordinates (x,y,z)

`get_coordinate()`

... *Description:* Gets the drone's coordinates [x,y,z]. *Return Value:* `[x, y, z]` *Example:* `python ret = api.get_coordinate() # Get drone coordinates [x, y, z]`

Get Drone Angles

`get_yaw()`

... *Description:* Gets the drone's angles. *Return Value:* An integer array: `[yaw, pitch, roll]` *Example:* `python ret = api.get_yaw()`

Get Drone Body Speed (X,Y,Z)

`get_plane_speed()`

... *Description:* Gets the drone's body speed (X, Y, Z). *Return Value:* An integer array: `[X, Y, Z]` *Example:* `python ret = api.get_plane_speed()`

Get Drone Time-of-Flight Height

`get_plane_distance()`

... *Description:* Gets the drone's Time-of-Flight height. *Return Value:* An integer representing the drone's ToF height. *Example:* `python ret = api.get_plane_distance()`

Get Drone ID

`get_plane_id()`

... *Description:* Gets the drone ID. *Return Value:* An integer representing the drone ID. *Example:* `python ret = api.get_plane_id()`

External Electromagnet

`Plane_cmd_electromagnet(type)`

... *Description:* Controls the external electromagnet. *Parameters:* * `type` : 2-Electromagnet attraction; 3-Electromagnet release. *Example:* `python ret = api.Plane_cmd_electromagnet(2)`

External Clamp

`Plane_cmd_clamp(type,angle)`

... *Description:* Controls the external clamp. *Parameters:* * `type` : 0-Clamp closed; 1-Clamp open. * `angle` : Angle to rotate the clamp to (0-180). *Example:* `python ret = api.Plane_cmd_clamp(1,180)`

Remember to replace this Markdown with a proper PDF using a suitable editor. The formatting might not be perfectly preserved depending on the editor you use.