

pyhula

Hula用Pythonパッケージ
以下バージョンに対応

pythonバージョン:3.6.7

インストール

ターミナルで以下のコマンドを実行してpyhulaをインストールします。
2つの方法でインストール可能です。

None

```
pip install pyhula
```

```
pip install pyhula-1.1.4-cp36-cp36m-win_amd64.whl
```

バージョンを確認する

ターミナルで「pip list」を実行してpyhulaのバージョンを確認します。
プログラムで以下のコードを実行する

Python

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

使用方法

以下のコードを参考にユーザーAPIインスタンスを作成してください。
そのインターフェースを利用して、Hulaを制御できます。

Python

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

api.single_fly_takeoff() #離陸
api.single_fly_touchdown() #着陸
```

API一覧

ドローンとの接続

```
connect(server_ip)
```

...

説明： ドローンの接続
 パラメーター： server_ip: ドローンのIPv4アドレス 入力しない場合は自動取得
 戻り値： True:成功 False:失敗

...

例：api.connect('192.168.1.118')
 例：api.connect()

離陸

```
single_fly_takeoff(led)
```

...

説明：ドローンを離陸させる

パラメータ： ledを入力しない場合のデフォルトは0,
 フォーマット：{'r':0, 'g':0, 'b':0, 'mode':1}
 r, g, b: 色域
 mode: 1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
 点滅

...

例：
 api.single_fly_takeoff()
 api.single_fly_takeoff({'r':16, 'g':15, 'b':100, 'mode':1})

着陸

```
single_fly_touchdown(led)
```

...

説明：ドローンを着陸させる

パラメータ： led入力しない場合のデフォルトは0

```

フォーマット : {'r':0, 'g':0, 'b':0, 'mode':1}
r, g, b : 色域
mode : 1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅
'''

```

```

例 : api.single_fly_touchdown()
api.single_fly_touchdown({'r':16, 'g':15, 'b':100, 'mode':1})

```

ホバリング

```

single_fly_hover_flight(time, led)

```

```

'''

```

説明 : ドローンのホバリング

パラメータ

time : ホバリング時間 (秒)

```

'''

```

例 :

```

api.single_fly_hover_flight(10)
api.single_fly_hover_flight(10, {'r':16, 'g':15, 'b':100, 'mode':1
})

```

前に飛ぶ

```

single_fly_forward(distance, led)

```

```

'''

```

説明 : ドローンを前に飛行させる

パラメータ

distance : 飛距離 (cm)

led : 入力しない場合のデフォルトは0

フォーマット : {'r':0, 'g':0, 'b':0, 'mode':1}

r, g, b : 色域

mode : 1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり

点滅

```

'''

```

```
例 :
api.single_fly_forward(100)
api.single_fly_forward(100,{'r':16,'g':15,'b':100,'mode':1})
```

後に飛ぶ

```
single_fly_back(distance,led)

'''
説明：ドローンを後に飛行させる

パラメータ
distance：飛行距離（cm）
led：入力しない場合のデフォルトは0
フォーマット：{'r':0,'g':0,'b':0,'mode':1}
r,g,b：色域
mode：1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅
'''

例：
api.single_fly_back(100)
api.single_fly_back(100,{'r':16,'g':15,'b':100,'mode':1})
```

左に飛ぶ

```
single_fly_left(distance,led)

'''
説明：ドローンを左に飛行させる

パラメータ
distance：飛行距離（cm）
led：入力しない場合のデフォルトは0
フォーマット：{'r':0,'g':0,'b':0,'mode':1}
r,g,b：色域、
mode：1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅
'''
```

```
例：
api.single_fly_left(100)
api.single_fly_left(100,{'r':16,'g':15,'b':100,'mode':1})
```

右に飛ぶ

```
api.single_fly_right(distance,led)

...
説明：ドローンを右に飛行させる

パラメータ
distance：飛行距離（cm）
led：入力しない場合のデフォルトは0
フォーマット：{'r':0,'g':0,'b':0,'mode':1}
r,g,b：色域
mode：1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅
...
```

例：

```
api.single_fly_right(100)
api.single_fly_right(100,{'r':16,'g':15,'b':100,'mode':1})
```

上に飛ぶ

```
single_fly_up(distance,led)

...
説明：ドローンを上に飛行させる

パラメータ
distance：飛行距離（cm）
led：入力しない場合のデフォルトは0
フォーマット：{'r':0,'g':0,'b':0,'mode':1}
r,g,b：色域、
mode：1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅
...
```

```

例：
api.single_fly_up(100)
api.single_fly_up(100,{'r':16,'g':15,'b':100,'mode':1})

```

下に飛ぶ

```

single_fly_down(distance,led)

...
説明：ドローンを下に飛行させる

パラメータ
distance：飛行距離（cm）
led：入力しない場合のデフォルトは0
フォーマット：{'r':0,'g':0,'b':0,'mode':1}
r,g,b：色域
mode：1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅
...

例：
api.single_fly_down(100)
api.single_fly_down(100,{'r':16,'g':15,'b':100,'mode':1})

```

左旋回

```

single_fly_turnleft(angle,led)

...
説明：ドローンを左に回転させる

パラメータ
angle：回転角度（度）
led：入力しない場合のデフォルトは0
フォーマット：{'r':0,'g':0,'b':0,'mode':1}
r,g,b：色域
mode：1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり

```

点滅
...

例：

```
api.single_fly_turnleft(90)
```

```
api.single_fly_turnleft(90,{'r':16,'g':15,'b':100,'mode':1})
```

右旋回

```
single_fly_turnright(angle,led)
```

...

説明：ドローンを右に回転させる

パラメータ

angle：回転角度（度）

led：入力しない場合のデフォルトは0

フォーマット：{'r':0,'g':0,'b':0,'mode':1}

r,g,b：色域

mode：1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり

点滅

...

例：

```
api.single_fly_turnright(90)
```

```
api.single_fly_turnright(90,{'r':16,'g':15,'b':100,'mode':1})
```

バウンス

```
single_fly_bounce(frequency, height,led)
```

...

説明：ドローンをバウンスさせる

パラメータ

```

frequency : バウンス回数
height : バウンス距離 (cm)
led : 入力しない場合のデフォルトは0
フォーマット : {'r':0, 'g':0, 'b':0, 'mode':1}
r, g, b : 色域
mode : 1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅
'''

例 :
api.single_fly_bounce(3, 50)
api.single_fly_bounce(3, 50, {'r':16, 'g':15, 'b':100, 'mode':1})

```

直線飛行

```

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

説明 : ドローンを直線飛行させる

パラメータ
x : 座標x (cm)  y : 座標y (cm)  z : 座標z (cm)
led : 入力しない場合のデフォルトは0
フォーマット : {'r':0, 'g':0, 'b':0, 'mode':1}
r, g, b : 色域
mode : 1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅
'''

例 : api.single_fly_straight_flight(100, 100, 100)
api.single_fly_straight_flight(100, 100,
100, {'r':16, 'g':15, 'b':100, 'mode':1})

```

サークル飛行

```

single_fly_radius_around(radius, led)

```

```
'''
```

説明：ドローンが弧を描くように飛行させる

パラメータ

radius: 半径 (cm、正: 反時計回り、負: 時計回り)

led: 入力しない場合のデフォルトは0

フォーマット: {'r':0, 'g':0, 'b':0, 'mode':1}

r, g, b: 色域

mode: 1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅

```
'''
```

例:

```
api.single_fly_radius_around(100)
```

```
api.single_fly_radius_around(100, {'r':16, 'g':15, 'b':100, 'mode':1})
```

360度旋回

```
single_fly_autogyratation360(num, led)
```

```
'''
```

説明：ドローンを時計回り・反時計回りで指定した回転数だけ
自転させる

パラメータ

num: (正: 反時計回り、負: 時計回り)

led: 入力しない場合のデフォルトは0

フォーマット: {'r':0, 'g':0, 'b':0, 'mode':1}

r, g, b: 色域

mode: 1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅

```
'''
```

例:

```
api.single_fly_autogyratation360(2)
```

```
api.single_fly_autogyratation360(2, {'r':16, 'g':15, 'b':100, 'mode':1})
```

宙返り

```
single_fly_somersault(direction)

...

説明：ドローンをその場で前後左右に宙返りさせる

パラメータ
DIRECTION_FORWARD=0, /*前. | */
DIRECTION_BACK=1, /* 後. | */
DIRECTION_LEFT=2, /* 左. | */
DIRECTION_RIGHT=3, /* 右. | */
led：入力しない場合のデフォルトは0
フォーマット：{'r':0, 'g':0, 'b':0, 'mode':1}
r, g, b：色域
mode：1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅

...

例：
api.single_fly_somersault(0)
api.single_fly_somersault(0, {'r':16, 'g':15, 'b':10
0, 'mode':1})
```

曲線飛行

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

...

説明：ドローンを曲線飛行（x, y, z）経路上で飛行させる

パラメータ
x：座標x（cm）（機体の左右、右が正） y：座標y（cm）（機体の前後、前が正）
z：座標z（cm）（機体の上下、上が正）
direction： True：反時計回り、False：時計回り デフォルトはTrue
led：入力しない場合のデフォルトは0
フォーマット：{'r':0, 'g':0, 'b':0, 'mode':1}
r, g, b：色域
mode：1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅
```

```
'''
```

```
例 : api.single_fly_curvilinearFlight(100, 100, 0)
api.single_fly_curvilinearFlight(100, 100, 0,
{'r':16, 'g':15, 'b':100, 'mode':1})
```

障害物検知

```
single_fly_barrier_aircraft(mode)
```

```
'''
```

説明 : ドローンに障害物を検知させる

パラメータ

mode : True:オン False:オフ

```
'''
```

```
例 : api.single_fly_barrier_aircraft(True)
```

巡回ライン検出

```
single_fly_Line_walking(fun_id, dist, way_color)
```

```
'''
```

説明 : 巡回ライン検出

パラメータ

fun_id = 0 //0 : 前方巡線、無視

dist //距離、単位 : cm

way_color //巡回ライン色の色、0-黒色、255-白色

return result = 1; //指令実行の結果 : 0-失敗、1-成功、2-成功かつライン発見

```
'''
```

```
例 : api.single_fly_Line_walking(0, 100, 0)
```

ターゲット(タグ)認識

```
single_fly_AiIdentifies(mode)

...

説明：指定したターゲット（タグ）を認識させる

パラメータ
mode：0～9：数字タグ0～9を認識、10：左矢印を認識、11：右矢印を認識、12：
上矢印を認識、13：下矢印を認識、20：タスク終了、65～90：大文字アルファベッ
トA～Zを認識
実行後、認識プロセスは300ms継続。成功した場合即終了。

戻り値：
x：タグカードとドローンのX座標
y：タグカードとドローンのY座標
z：タグカードとドローンのZ座標
angle：タグカードとドローンの角度
result：True：認識成功 False：認識失敗

...

例：api.single_fly_AiIdentifies(1)
```

QRコードの位置合わせ(オプティカルフロー、ジンバルカメラ)

```
single_fly_Qrcode_align(mode, qr_id)

...

説明：オプティカルフロー（下方ビジョンセンサー）、ジンバルカメラ（前方カメラ）
によるQRコードの位置合わせ

パラメータ
qr_id：QRコードID [0～9]
mode：
mode = 0：オプティカルフローでQRコードに位置合わせ
mode = 1：ジンバルカメラでQRコードに位置合わせ
戻り値：
result：True：認識成功 False：認識失敗

...

例：api.single_fly_Qrcode_align(0, 1)
```

QRコード認識(オプティカルフロー、ジンバルカメラ)

```
single_fly_recognition_Qrcode(mode, qr_id)

...
説明：オプティカルフロー（下方ビジョンセンサー）、ジンバルカメラ（前方カメラ）
によるQRコード認識

パラメータ
qr_id: QRコードID [0~9]
mode:
mode = 0 オプティカルフローでQRコード認識
mode = 1: 前置カメラでQRコード認識
戻り値:
{
result: //False: 認識失敗 True: 認識成功
x: //ドローンとQRコードとの距離（X方向）
y: //ドローンとQRコードとの距離（Y方向）
z: //ドローンとQRコードとの距離（Z方向）
yaw: //ドローンとQRコードとの相対角度（ヨー角）
qr_id: // 認識されたQRコードのID
}
...

例: api.single_fly_Qrcode_align(0, 1)
```

QRコード追跡

```
single_fly_track_Qrcode(qr_id, time)

...
説明：QRコード[0-9]番を[time]秒間追跡する

パラメータ
qr_id: QRコードID [0~9]
time: 追跡時間（秒）
戻り値:
result: True: 認識成功 False: 認識失敗
...

例: api.single_fly_track_Qrcode(1, 10)
```

色認識:現在の映像ストリームのフレームから色を取得

```
single_fly_getColor()
...
説明：色認識（現在の映像ストリームのフレームの色を取得する）

パラメータ
mode：1：開始、1フレームだけ処理を実行
戻り値：
r,g,b：色域
state：0：失敗 1：成功
...

例：ret = api.single_fly_getColor()
#戻り値：r,g,b色域 state:0失敗 1成功
```

ライトの色とモードを設定する

```
single_fly_lamplight(r, g, b, time, mode)
...
説明：ライトの色とモードを設定する

パラメータ
r,g,b：色域
time：ライトの点灯時間/秒
mode：1/点灯、2/消灯、4/RGB 3色サイクル、16/レインボー、32/点滅、64/ふんわり
点滅
戻り値：True：実行成功 False：実行失敗
...

例：api.single_fly_lamplight(255, 0, 0, 1, 1) #ライトの色とモードを
設定する
```

レーザー発射 ※日本では使用できない

```
plane_fly_generating(type, data ,reserve)

...

説明：レーザーを発射させる

パラメータ
type:0: 単発  1: 連発  2: レーザー受信開始  3: レーザー受信終了
4: 弾数無制限の連発  5: 発射停止
data = 10    //レーザー連発頻度（回/秒、範囲1～14）
reserve = 100 //弾数（データ範囲1～255）

...

例: api.plane_fly_generating(0, 10, 100) #単発
    api.plane_fly_generating(2, 10, 100) #レーザー受信開始
```

レーザー受信判定

```
plane_fly_laser_receiving()

...

説明：レーザー受信器がレーザーに当たったかどうかを検出する。
戻り値:
True: 被弾（レーザーに当たった）
False: 未被弾（レーザーに当たっていない）

...

例: api.plane_fly_laser_receiving()
```

QRカーペットによる測位のオン/オフ

```
Plane_cmd_switch_QR(type)

...
```

説明：QRカーペット（2.2m/4.3m/6.4m/8.5m）による測位機能のON/OFF

パラメータ

type :

0：有効にする（ON）

1：無効にする（OFF）

...

例：api.Plane_cmd_switch_QR(0)

写真を撮る

Plane_fly_take_photo()

...

説明：写真を撮影する。ビデオストリームを有効にした後に呼び出す必要があります。

...

例：api.Plane_fly_take_photo() #写真を撮影する

ビデオを撮る

Plane_cmd_switch_video(type)

...

説明：録画を開始または終了する。

パラメータ

type : 0：録画開始 1：録画終了

...

例：api.Plane_cmd_switch_video(0) #録画を開始する

ビデオストリームの有効化

```
Plane_cmd_swith_rtp(type)
...
説明：ビデオストリームを有効または無効にする。

パラメータ
type：
0： ビデオストリームを有効
1： ビデオストリームを無効
...
例： api.Plane_cmd_swith_rtp(0) #ビデオストリームを有効にする
```

ビデオストリームウィンドウを開く

```
single_fly_flip_rtp()
...
説明：ドローンのビデオストリームを表示するためのウィンドウを開くパラメータ
...
例： api.single_fly_flip_rtp() #ビデオストリームのウィンドウを開く
```

ジンバルカメラの角度設定

```
Plane_cmd_camera_angle(type, data)
...
説明：メインカメラの俯仰（ピッチ）角度を設定する。

パラメータ
type：
0： 上（絶対）
```

```

1: 下（絶対）
2, 3: アルゴリズム制御
4: 校準（キャリブレーション）
5: 積木上（相対）
6: 積木下（相対）
data= 30; // 回転角度（0～90度）

...

例: api.Plane_cmd_camera_angle(0, 30) #主カメラの俯仰（ピッチ）角度
    を設定する。

```

アーム(低速プロペラ回転)

```

plane_fly_arm()

...
説明: モーターのアンロックパラメータ
...

例: api.plane_fly_arm() #低速プロペラ回転

```

ディスアーム(低速プロペラ回転の停止)

```

plane_fly_disarm()

...
説明: モーターのロックパラメータ
...

例: api.plane_fly_disarm() #低速プロペラ回転の停止

```

障害物回避情報の取得

```
Plane_getBarrier()
```

```
'''
```

説明：障害物回避情報のパラメータを取得する。

パラメータ

戻り値：辞書型（ディクショナリ）各方向の障害物の有無を示す

True：障害物あり False：障害物なし

```
{
'forward': True
'back': True,
'left': True,
'right': True,
}
```

```
'''
```

例：ret = api.Plane_getBarrier() #障害物回避情報を取得する

ドローンのバッテリー残量の取得

```
get_battery()
```

```
'''
```

説明：ドローンのバッテリー残量（パーセンテージ）を取得する

戻り値：整数型：バッテリーの残量をパーセンテージで示す。

```
'''
```

例：ret = api.get_battery() #ドローンのバッテリー残量（%）の取得

ドローンの座標(x, y, z)の取得

```
get_coordinate()
```

```
...  
説明：ドローンの座標[x, y, z]を取得する  
  
戻り値：[x, y, z]  
  
...  
  
例：ret = api.get_coordinate() #ドローンの座標[x, y, z]の取得する
```

ドローンの傾きの取得

```
get_yaw()  
  
...  
説明：ドローンの角度を取得する（度）  
  
戻り値  
整数型：[ヨー角、ピッチ角、ロール角]  
  
...  
  
例：ret = api.get_yaw()
```

ドローンの機体速度(X軸速度、Y軸速度、Z軸速度)の取得

```
get_plane_speed()  
  
...  
説明：ドローンの機体速度（X軸速度、Y軸速度、Z軸速度）を取得する  
  
戻り値  
整数型：[X,Y,Z]  
  
...  
  
例：ret = api.get_plane_speed()
```

ドローンのToF高度の取得

```
get_plane_distance()  
...  
説明：ドローンのToF高度を取得する  
  
戻り値  
整数型の配列：ドローンのToF高度  
  
...  
例：ret = api.get_plane_distance()
```

機体IDの取得

```
get_plane_id()  
...  
説明：ドローンのIDを取得する  
  
戻り値  
整数型の配列：ドローンのID  
  
...  
例：ret = api.get_plane_id()
```

電磁石の使用(拡張ポート使用時)

```
Plane_cmd_electromagnet(type)  
...  
説明：外部電磁石  
  
パラメータ  
type：  
2：電磁石を吸着する  
3：電磁石を弾出する
```

```
...
```

```
例: ret = api.Plane_cmd_electromagnet(2)
```

グリッパーの使用(拡張ポート使用時)

```
Plane_cmd_clamp(type, angle)
```

```
...
```

説明: 外部グリッパー

パラメータ

type:

0: グリッパーを閉じる

1: グリッパーを開く

angle: グリッパーの回転角度を指定する (0~180度)

```
...
```

```
例: ret = api.Plane_cmd_clamp(1, 180)
```