Qumcum PythonAPI Specification

バージョン	1.0
作　成　日	2022年4月15日
最終更新日	2022年4月15日

Qumcum サイト [https://qumcum.com/]

目次

1 はじめに

このたびは Qumcum PythonAPI をご利用いただき、ありがとうございます。

Qumcum PythonAPI は、クムクムロボットを PC から BLE 接続を用い、Python で制御するためのインターフェースを提供する API です。
当 API を使用することで、クムクムロボットの Python によるプログラミング学習が可能となります。

本書では、Qumcum API で用意している関数の説明と簡単な使用方法について記載します。

1.1 本書で必要となる知識と想定する読者

Python によるプログラミングの基本的な知識
クムクムロボットをお持ちで Python プログラミングに興味をお持ちの開発者・技術者

1.2 本書の構成

はじめにお読みください
「はじめに」

Qumcum PythonAPI の仕様を説明します
「API 仕様」

Qumcum PythonAPI の関数それぞれの詳細を説明します
「API 詳細」

2　API 仕様

2.1　概要

Qumcum PythonAPI は、下記の API を公開しています。

API 名	機能
connect	クムクムと接続し通信を開始する
end	クムクムとの通信を終了する
get_sensor_value	超音波センサ計測値を取得する
get_mic_value	マイク計測値を取得する
get_battery_value	電池残量の計測値を取得する
get_info	ロボットの基本情報を取得する
led_on	RGB-LED の指定した色を点灯させる
led_off	RGB-LED の指定した色を消灯させる
sound	指定した時間(msec)だけ指定した周波数の音を発生させる
voice_speed	発話する速度を指定する
voice	発話する
motor_power_on	モーター電源を ON にする
motor_power_off	モーター電源を OFF にする
motor_set_pos	モーターの角度(絶対位置)を指定する
motor_start	モーターを指定した角度(絶対位置)に移動させる
motor_pos_all	全モーター(7 つ)の角度をセットする
motor_angle_time	モーターの角度(絶対位置)、動作時間を指定する
motor_angle_multi_time	全モーター(7 つ)の角度をセットする
get_lib_ver	ライブラリのバージョン取得
wait	一定時間待つ

2.2 クムクムからのレスポンスについて

関数を呼び出して、クムクムから正常にコマンドを受け付けた時のレスポンスは、例のようなカンマ区切りの文字列で返ってきます。

例）

@,2,38,999,500

フォーマット

'@', <内部管理用番号> , <バッテリ残量の目安> , <超音波距離センサの計測値> , <マイクの計測値>

こちらは、下表のように定義されています。

名前	意味
'@'	固定で@になります
<内部管理用番号>	内部で使用されている管理用の番号です
<バッテリ残量の目安>	クムクムのバッテリ残量の目安(0～100：100 が満充電状態)
<超音波距離センサの計測値>	クムクムで最後に計測した超音波距離センサの計測値
<マイクの計測値>	クムクムで最後に計測したマイクの計測値

3　API 詳細

3.1　connect

API 名	connect
機能	クムクムと接続し通信を開始する
宣言	def connect(conn_str):
引数	conn_str・・・接続文字列(クムクムに割り当てられた 4 桁の 16 進文字列)
戻り値	処理結果(接続に成功した場合は 0、失敗した場合は-1)
説明	クムクムと通信を行うための接続処理を行います デバイスを探索してから接続を行うため時間がかかります
使用例	# 引数は自分のクムクムの下 4 桁の番号に # 書き換えてください qumcum.connect('12B6') # '12B6'が割り当てられたクムクムロボットと接続を行います

3.2　end

API 名	end
機能	クムクムとの通信を終了する
宣言	def end():
引数	なし
戻り値	処理結果(終了に成功した場合は 0、失敗した場合は-1)
説明	クムクムとの通信終了処理を行います。
使用例	qumcum.end()　　# クムクムから切断します

3.3 get_sensor_value

API 名	get_sensor_value
機能	超音波センサ計測値を取得する
宣言	def get_sensor_value():
引数	なし
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時は超音波センサの計測値、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	クムクムで計測した超音波センサの計測値を取得します
使用例	sens_val = get_sensor_value()　# 超音波センサの計測を行い、計測値を取得します print(sens_val)　# 取得した超音波センサの計測値を出力します

3.4 get_mic_value

API 名	get_mic_value
機能	マイク計測値を取得する
宣言	def get_mic_value():
引数	なし
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はマイクの計測値、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	クムクムで計測したマイクの計測値を取得します
使用例	mic_val = get_mic_value()　# マイクの計測を行い、計測値を取得します print(mic_val)　# 取得したマイクの計測値を出力します

3.5 get_battery_value

API 名	get_battery_value
機能	バッテリ残量の目安を取得する
宣言	def get_battery_value():
引数	なし
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はバッテリ残量の目安を表す値(0-100、100 が満充電状態)、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	クムクムで計測したバッテリ残量の目安を取得します
使用例	battery_val = get_battery_value()　# バッテリ残量の計測を行い、計測した目安の値を取得します print(battery_val)　# 取得したバッテリ残量の目安の値を出力します

3.6 get_info

API 名	get_info
機能	ロボットの基本情報を取得する
宣言	def get_info():
引数	なし
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムの基本情報を表す一連の文字列、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	現在接続しているクムクムの基本情報を取得します クムクムの基本情報は下記のカンマ区切り形式で文字列として取得されます INF,Qumcum,CRETARIA,<ファームウェアバージョン>,<ロボットの MAC アドレス>
使用例	info_str = qumcum. get_info()　# 現在接続しているクムクムの基本情報を取得します print(info_str)　# 取得した基本情報を出力します

3.7 led_on

API 名	led_on
機能	RGB-LED の指定した色を点灯させる
宣言	def led_on(color):
引数	color・・・点灯させる LED の色の番号
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	指定できる LED の色の番号は 1~3 で 1・・・赤(Red) 2・・・青(Blue) 3・・・緑(Green) です。 指定した色の LED がすでに点灯している場合は、変化しません
使用例	qumcum.led_on(1)　# 赤の LED を点灯します qumcum.wait(1)　　# <-- 赤の LED が点灯してから 1 秒待ちます qumcum.led_off(1)　# 赤の LED を消灯します

3.8 led_off

API 名	led_off
機能	RGB-LED の指定した色を消灯させる
宣言	def led_off(color):
引数	color・・・消灯させる LED の色の番号
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	指定できる LED の色の番号は 1~3 で 1・・・赤(Red) 2・・・青(Blue) 3・・・緑(Green) です。 指定した色の LED がすでに消灯している場合は、変化しません
使用例	qumcum.led_on(3)　# 緑の LED を点灯します qumcum.wait(1)　　# <-- 緑の LED が点灯してから 1 秒待ちます qumcum.led_off(3)　# 緑の LED を消灯します

3.9 sound

API 名	sound
機能	指定した時間の間、指定した周波数の音を発生させる
宣言	def sound(freq, time_ms):
引数	freq・・・発生させる周波数(Hz) time_ms・・・音を出力する時間(msec)
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	引数 freq(周波数)には 30~18,000(Hz)くらい(人の耳で聞こえる)までの音を出すことが可能です time_ms は一回の呼び出しで 9,999(msec)(9.999 秒)まで音を出すことが可能です
使用例	qumcum.sound(880,9999)　# 880Hz の音を 9.999 秒間出します qumcum.wait(10)　　# <-- 10 秒待ちます

3.10 voice_speed

API 名	voice_speed
機能	発話する速度を指定する
宣言	def voice_speed(speed):
引数	speed・・・発話速度
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	voice()で発話を行うときの発話の速さの設定を行います。 引数 speed には 30 から 300 までの数値を指定することができ、30 は「とてもゆっくり」、300 は「とてもはやく」発話するようになります 当関数を 1 度も指定してない場合の初期値は 100(ふつうのはやさ)となります
使用例	qumcum.voice_speed(30)　　# 発話の速度を 30(とてもゆっくり)に指定する qumcum.voice('konnitiwa')　# 「こんにちわ」と発話します qumcum.wait(1)　　# 発話が終わるまで待ちます qumcum.voice_speed(300)　　# 発話の速度を 300(とてもはやく)に指定する qumcum.voice('konnitiwa')　# 「こんにちわ」と発話します qumcum.wait(1)　　# 発話が終わるまで待ちます

3.11 voice

API 名	voice
機能	発話する
宣言	def voice(words):
引数	words・・・発話させたい文章など
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	クムクムの音声合成を用いて発話させます 引数 words には発話させたい文章などをローマ字で指定します。数字は専用のタグをつけて表現させる必要があります 「いちえん」と発話させたい場合に、引数に'1en'としても発話しません。 数字を発話させるには引数で'<NUM VAL=1>'とする必要があります また、100 円(「ひゃくえん」)などの桁を認識した発話の場合は、'<NUMK VAL=100>'という指定を行う必要があります A001(「えーぜろぜろいち」などの英数を発話させるには引数で'<ALPHA VAL=A001>'という指定を行う必要があります
使用例	qumcum.voice('konnitiwa') # 「こんにちわ」と発話します qumcum.wait(1) # 発話が終わるまで待ちます qumcum.voice('<NUM VAL=123>') # 「いちにさん」と発話します qumcum.wait(1) # 発話が終わるまで待ちます qumcum.voice('<NUMK VAL=123>en') # 「ひゃくにじゅうさんえん」と発話します qumcum.wait(1) # 発話が終わるまで待ちます qumcum.voice('<ALPHA VAL=C001>') # 「しーぜろぜろいち」と発話します qumcum.wait(1) # 発話が終わるまで待ちます

3.12 motor_power_on

API 名	motor_power_on
機能	モーター電源を ON にする
宣言	def motor_power_on(motor_time_ms):
引数	motor_time_ms・・・モータの動作時間(msec)
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	クムクムのモータの電源を ON にします。 (クムクムのモータはこの関数でモータの電源を ON しないと動作しません)
使用例	qumcum.motor_power_on(500)　# 動作時間 500msec でモータ電源を ON にします qumcum.motor_set_pos(4, 0)　# 4 番(頭)のモータを 0 度(右)の位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.wait(1)　　# 動作が終わるまで待ちます qumcum.motor_power_off()　# モータ電源を OFF にします

3.13 motor_power_off

API 名	motor_power_off
機能	モーター電源を OFF にする
宣言	def motor_power_off():
引数	なし
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	クムクムのモータの電源を OFF にします。 モータの電源は Python プログラムを終了してもロボットの電源が ON の間は自動で OFF にはなりません。プログラムを終了する直前に一度呼び出してモータ電源を OFF にするようにしてください モータ電源を OFF にし忘れると、充電池の消費が多くなりますのでご注意ください
使用例	qumcum.motor_power_on(500)　# 動作時間 500msec でモータ電源を ON にします qumcum.motor_set_pos(4, 0)　# 4 番(頭)のモータを 0 度(右)の位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.wait(1)　　# 動作が終わるまで待ちます qumcum.motor_power_off()　# モータ電源を OFF にします

3.14 motor_set_pos

API 名	motor_set_pos
機能	モーターの角度(絶対位置)を指定する
宣言	def motor_set_pos(no, pos):
引数	no・・・モータの番号 pos・・・モータの角度
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	モータを指定した角度に動くように指示をします。 引数 no のモータの番号は、1~7 までの数字で割り当ては下記のようになります。 1:右手、2:右足、3:右足首、4:頭、5:左足首、6:左足、7:左手 引数 pos のモータの角度は、0~180 までの数字です(足[右足、右足首、左足首、左足]については、ロボットの構造上 90±30 度[60~120]までしか動かせません) この関数を呼び出したタイミングではモータは動きません。この関数で角度を設定た後、 motor_start()関数を呼び出すとモータが動きます。
使用例	qumcum.motor_power_on(500)　# 動作時間 500msec でモータ電源を ON にします qumcum.motor_set_pos(4, 0)　# 4 番(頭)のモータを 0 度(右)の位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.wait(1)　　# 動作が終わるまで待ちます qumcum.motor_power_off()　# モータ電源を OFF にします

3.15 motor_start

API 名	motor_start
機能	モーターを指定した角度(絶対位置)に移動させる
宣言	def motor_start(no_wait=False):
引数	no_wait(省略可)・・・モータの動作完了を待たないか(True:動作完了を待たない、False:動作完了を待つ)
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	当関数を呼び出す前に motor_set_pos 関数などで指定したモータの角度への動作を開始します 引数 no_wait は「動作を待たないか」を指定するもので、True にすると当関数呼び出し後すぐに処理が返ってきて次の処理が実行されるようになります(デフォルトは False で「モータの動作完了を待ちます」)
使用例	qumcum.motor_power_on(500)　# 動作時間 500msec でモータ電源を ON にします qumcum.motor_set_pos(4, 0)　# 4 番のモータを 0 度の位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.motor_set_pos(4, 90)　# 4 番のモータを 90 度の位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.motor_power_off()　# モータ電源を OFF にします

3.16 motor_pos_all

API 名	motor_pos_all
機能	全モーター(7 つ)の角度をセットする
宣言	def motor_pos_all(pos1, pos2, pos3, pos4, pos5, pos6, pos7):
引数	pos1・・・モーター1(右手)の角度(0~180) pos2・・・モーター2(右足)の角度(60~120) pos3・・・モーター3(右足首)の角度(60~120) pos4・・・モーター4(頭)の角度(0~180) pos5・・・モーター5(左足首)の角度(60~120) pos6・・・モーター6(左足)の角度(60~120) pos7・・・モーター7(左手)の角度(0~180)
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	クムクムの標準で搭載されているモータ 7 つすべての角度を指定します (ロボットの構造上足のモータは 90±30 度[60~120]までしか動かせません) この関数を呼び出したタイミングではモータは動きません。この関数で角度を設定た後、 motor_start()関数を呼び出すとモータが動きます。 この関数を呼び出した後に motor_start()関数を呼び出すと 7 つのモータが同時に動き出しますのでモータの角度を十分確認してから動かすようにしてください
使用例	qumcum.motor_power_on(500)　# 動作時間 500msec でモータ電源を ON にします qumcum.motor_pos_all(80, 80, 80, 80, 80, 80, 80)　# 7 つのモータを 80 度の位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.motor_pos_all(100, 100, 100, 100, 100, 100, 100)　# 7 つのモータを 100 度の位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.motor_pos_all(0, 90, 90, 90, 90, 90, 180)　# まっすぐ起立したときの位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします

	qumcum.motor_power_off()　# モータ電源を OFF にします

3.17 motor_angle_time

API 名	motor_angle_time
機能	モーターの角度(絶対位置)、動作時間を指定する
宣言	def motor_angle_time(no, pos, motor_time):
引数	no・・・モーターの番号 pos・・・モーターの角度 motor_time・・・モーターの動作時間
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	モータを指定した角度に動くように指示をします。 引数 no のモータの番号は、1~7 までの数字で割り当ては下記のようになります。 1:右手、2:右足、3:右足首、4:頭、5:左足首、6:左足、7:左手 引数 pos のモータの角度は、0~180 までの数字です(足[右足、右足首、左足首、左足]については、ロボットの構造上 90±30 度[60~120]までしか動かせません) 引数 motor_time はミリ秒単位で 100~9999(msec)まで指定できます この関数を呼び出したタイミングではモータは動きません。この関数で角度を設定た後、 motor_start()関数を呼び出すとモータが動きます。
使用例	qumcum.motor_power_on(500)　# 動作時間 500msec でモータ電源を ON にします qumcum.motor_angle_time(4, 0, 500)　　# 4 番(頭)のモータを 500msec で 0 度(右)の位置へ動かします qumcum.motor_start()　　# モータを動かします qumcum.motor_angle_time(4, 180, 2000)　# 4 番(頭)のモータを 2000msec で 180 度(左)の位置へ動かします qumcum.motor_start()　　# モータを動かします

	qumcum.motor_angle_time(4, 0, 100)　　# 4 番(頭)のモータを 100msec で 0 度(右)の位置へ動かします qumcum.motor_start()　　# モータを動かします qumcum.motor_angle_time(4, 90, 500)　　# 4 番(頭)のモータを 500msec で 90 度(正面)の位置へ動かします qumcum.motor_start()　　# モータを動かします qumcum.motor_power_off()　# モータ電源を OFF にします

3.18 motor_angle_multi_time

API 名	motor_angle_multi_time
機能	全モーター(7 つ)の角度と動作時間をセットする
宣言	def motor_angle_multi_time(pos1, pos2, pos3, pos4, pos5, pos6, pos7, motor_time):
引数	pos1・・・モーター1(右手)の角度(0~180) pos2・・・モーター2(右足)の角度(60~120) pos3・・・モーター3(右足首)の角度(60~120) pos4・・・モーター4(頭)の角度(0~180) pos5・・・モーター5(左足首)の角度(60~120) pos6・・・モーター6(左足)の角度(60~120) pos7・・・モーター7(左手)の角度(0~180) motor_time・・・モーターの動作時間
戻り値	タプル(要素数 2)が返ります int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'という文字列が格納されます
説明	クムクムの標準で搭載されているモータ 7 つすべての角度とモータの動作時間を指定します (ロボットの構造上足のモータは 90±30 度[60~120]までしか動かせません) この関数を呼び出したタイミングではモータは動きません。この関数で角度を設定た後、 motor_start()関数を呼び出すとモータが動きます。 この関数を呼び出した後に motor_start()関数を呼び出すと 7 つのモータが同時に動き出しますのでモータの角度を十分確認してから動かすようにしてください
使用例	

	qumcum.motor_power_on(500)　# 動作時間 500msec でモータ電源を ON にします qumcum.motor_angle_multi_time(180, 80, 80, 80, 80, 80, 0, 500)　# 両手を上げて他のモータを 80 度の位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.wait(1) qumcum.motor_angle_multi_time(0, 100, 100, 100, 100, 100, 180, 100)　# 両手を下げて他のモータを 100 度の位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.wait(1) qumcum.motor_angle_multi_time(180, 90, 90, 90, 90, 90, 0, 1000)　# ばんざいをする位置へ 1000msec(1 秒)で動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.motor_angle_multi_time(0, 90, 90, 90, 90, 90, 180, 500)　# まっすぐ起立したときの位置へ動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.motor_power_off()　# モータ電源を OFF にします

3.19 get_lib_ver

API 名	get_lib_ver
機能	ライブラリのバージョンを取得
宣言	def get_lib_ver():
引数	なし
戻り値	ライブラリバージョンの文字列(“0.1.0”など)
説明	当ライブラリのバージョンを表現する文字列を返します
使用例	lib_ver = qumcum.get_lib_ver()　　# lib_ver にバージョン番号の文字列が格納されます print(lib_ver)　　# ライブラリバージョンを表示します

3.20 wait

API 名	wait
機能	一定時間待ちます
宣言	def wait(time_sec):
引数	time_sec・・・待つ時間(単位：秒)
戻り値	なし(0 のみ)
説明	引数 time_sec に 1 を指定した場合は 1 秒待ちます。 内部で time.sleep を呼ぶだけです
使用例	qumcum.led_on(1)　# 赤の LED を点灯します qumcum.wait(1)　　# <-- 赤の LED が点灯してから 1 秒待ちます qumcum.led_off(1)　# 赤の LED を消灯します

Memo