py_websockets_bot Documentation

Release 0.1

Dawn Robotics Ltd

October 15, 2015

Contents

1 Table of Contents **3**
1.1 API Reference 3

2 Indices and tables **7**

Python Module Index **9**

This library uses websockets to talk to the raspberry_pi_camera_bot web server. This allows you to programmatically control the robot either from a script running locally on the robot's Raspberry Pi, or from a script that runs on a separate computer, connected to the robot by WiFi.

CHAPTER 1

Table of Contents

Contents:

1.1 API Reference

This package provides the *WebsocketsBot* class which uses websockets to communicate with and control a robot running the raspberry_pi_camera_bot robot_web_server.py

1.1.1 WebsocketsBot

class py_websockets_bot.**WebsocketsBot**

Interface class for talking to a robot running the raspberry_pi_camera_bot robot_web_server.py using websockets

__init__(*hostname='localhost'*, *port=80*, *camera_port=8080*)

Constructs a *WebsocketsBot*

Parameters

- **hostname** (*str*) – The ip address of the robot
- **port** (*int*) – The port of the robot's web server
- **camera_port** (*int*) – The port of the web server providing camera images

centre_neck()

Centres the neck of the robot

disconnect()

This routine should be called when the *WebsocketsBot* is no longer needed

estimate_robot_time_offset()

Estimates the time difference between our system clock and the robot's system clock. We use a simple algorithm based on the method given at http://www.mine-control.com/zack/timesync/timesync.html The estimation process will take about 10 to 20 seconds so should only be called occasionally (perhaps once every 30 minutes or so).

TODO: Get this to work asynchronously...

Returns The offset to add to the system time in order to get to the robot's time

Return type float

get_battery_voltage()

Reads the battery voltage from the robot.

Returns The battery voltage read from the robot, along with the client time when the battery voltage was received.

Return type (float, float)

get_latest_camera_image(*max_image_age=None*)
This routine will block until an image is obtained from the robot's camera.
Parameters **max_image_age** (*float*) – Specifies the maximum age of the image to retrieve in seconds. If this is None, then the routine will retrieve the first available image. If the currently available image is older than this age then the routine blocks until a newer image is retrieved.
Returns a tuple containing the image and its timestamp
Raises Exception if *start_streaming_camera_images()* has not been called

get_latest_motion_vectors(*max_motion_vectors_age=None*)
This routine will block until an image is obtained from the robot's camera.
Parameters **max_motion_vectors_age** (*float*) – Specifies the maximum age of the motion vectors to retrieve in seconds. If this is None, then the routine will retrieve the first available set of motion vectors. If the currently available motion vectors are older than this age then the routine blocks until a newer set of motion vectors is retrieved.
Returns a tuple containing the motion vectors and a timestamp
Raises Exception if *stop_streaming_motion_vectors()* has not been called

get_latest_small_camera_image(*max_image_age=None*)
This routine will block until an image is obtained from the robot's camera.
Parameters **max_image_age** (*float*) – Specifies the maximum age of the image to retrieve in seconds. If this is None, then the routine will retrieve the first available image. If the currently available image is older than this age then the routine blocks until a newer image is retrieved.
Returns a tuple containing the image and its timestamp
Raises Exception if *start_streaming_small_camera_images()* has not been called

get_robot_config()
Reads the current configuration of the robot.
Returns A `RobotConfig` object containing configuration parameters for the robot
Return type `RobotConfig`

get_robot_status_dict()
Gets a dictionary containing various status variables from the robot. If robot sensor data is returned then it will be returned as a dictionary of sensor readings in the entry called 'sensors'. Each sensor reading is a dictionary consisting of a 'data' entry and a 'timestamp' entry, where the timestamp is the system clock time of the robot in seconds when the reading was taken.
return A dictionary containing status variables from the robot, along with the client time when the status dictionary was received.

Return type (dict, float)

power_off_robot()
Instructs the Pi on the robot to shut down

set_motor_joystick_pos(*joystickX*, *joystickY*)
Passes a movement joystick command to the robot. The magnitude of the joystick input should be less than or equal to 1.0. So

$$\sqrt{\text{joystickX}^2 + \text{joystickY}^2} \leq 1.0,$$

however, if it's not then it will be constrained automatically.
Parameters

- **joystickX** (*float*) – Joystick X input from -1.0 (left) to 1.0 (right)
- **joystickY** (*float*) – Joystick Y input from -1.0 (backwards) to 1.0 (forwards)

set_motor_speeds (*left_motor_speed*, *right_motor_speed*)

Sets the motor speeds of the robot

Parameters

- **left_motor_speed** (*float*) – Left motor speed from 0 to 100%
- **right_motor_speed** (*float*) – Right motor speed from 0 to 100%

set_neck_angles (*pan_angle_degrees*, *tilt_angle_degrees*)

Sets the neck angles of the robot in degrees. The centre point for each motor is 90 degrees

Parameters

- **pan_angle_degrees** (*float*) – Pan servo angle from 0 to 180 degrees
- **tilt_angle_degrees** (*float*) – Tilt servo angle from 0 to 180 degrees

set_neck_joystick_pos (*joystickX*, *joystickY*)

Passes a neck joystick command to the robot. The magnitude of the joystick input should be less than or equal to 1.0. So

$$\sqrt{\text{joystickX}^2 + \text{joystickY}^2} \leq 1.0,$$

however, if it's not then it will be constrained automatically.

Parameters

- **joystickX** (*float*) – Joystick X input from -1.0 (left) to 1.0 (right)
- **joystickY** (*float*) – Joystick Y input from -1.0 (down) to 1.0 (up)

set_robot_config (*config*)

Sends configuration parameters to the robot. In order to avoid disrupting existing configuration settings it is recommended that you call *get_robot_config()* first, modify the configuration parameters you're interested in, and then call this routine with the modified robot configuration.

:param `RobotConfig` config: The configuration to send to the robot

start_streaming_camera_images (*image_callback=None*)

This routine must be called to start streaming images from the robot's camera. Images can either be obtained in a blocking fashion, by calling *get_latest_camera_image()*, or an image callback can be provided.

Parameters **image_callback** (*func*) – A function to call with an image and timestamp when an image is obtained.

start_streaming_motion_vectors (*motion_vectors_callback=None*)

This routine must be called to start streaming motion vectors from the robot's camera. Motion vectors can either be obtained in a blocking fashion, by calling *get_latest_motion_vectors()*, or a callback can be provided.

Parameters **motion_vectors_callback** (*func*) – A function to call with motion vectors and a timestamp when a new batch of motion vectors is obtained.

start_streaming_small_camera_images (*image_callback=None*)

This routine must be called to start streaming 'small' images from the robot's camera. Images can either be obtained in a blocking fashion, by calling *get_latest_small_camera_image()*, or an image callback can be provided.

A small image is the normal camera image, reduced to a size of 160x120. Processing smaller images can give a large speed up to a lot of computer vision algorithms, and this is very important when running on a computationally constrained platform such as the Raspberry Pi.

Parameters **image_callback** (*func*) – A function to call with an image and timestamp when an image is obtained.

stop_streaming_camera_images()
Stops streaming images from the robot's camera

stop_streaming_motion_vectors()
Stops streaming motion vectors from the robot's camera

stop_streaming_small_camera_images()
Stops streaming 'small' images from the robot's camera

update()
This routine must be called periodically to ensure that background commications with the robot (to keep camera images streaming for example) happen on a regular basis.

1.1.2 CameraStreamingData

class py_websockets_bot.**CameraStreamingData**

1.1.3 ImageReaderProcess

class py_websockets_bot.**ImageReaderProcess**

1.1.4 MotionVectorsReaderProcess

class py_websockets_bot.**MotionVectorsReaderProcess**

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

p

py_websockets_bot, 3

Symbols

__init__() (py_websockets_bot.WebsocketsBot method), 3

C

CameraStreamingData (class in py_websockets_bot), 6
centre_neck() (py_websockets_bot.WebsocketsBot method), 3

D

disconnect() (py_websockets_bot.WebsocketsBot method), 3

E

estimate_robot_time_offset() (py_websockets_bot.WebsocketsBot method), 3

G

get_battery_voltage() (py_websockets_bot.WebsocketsBot method), 3
get_latest_camera_image() (py_websockets_bot.WebsocketsBot method), 4
get_latest_motion_vectors() (py_websockets_bot.WebsocketsBot method), 4
get_latest_small_camera_image() (py_websockets_bot.WebsocketsBot method), 4
get_robot_config() (py_websockets_bot.WebsocketsBot method), 4
get_robot_status_dict() (py_websockets_bot.WebsocketsBot method), 4

I

ImageReaderProcess (class in py_websockets_bot), 6

M

MotionVectorsReaderProcess (class in py_websockets_bot), 6

P

power_off_robot() (py_websockets_bot.WebsocketsBot method), 4
py_websockets_bot (module), 3

S

set_motor_joystick_pos() (py_websockets_bot.WebsocketsBot method), 4
set_motor_speeds() (py_websockets_bot.WebsocketsBot method), 5
set_neck_angles() (py_websockets_bot.WebsocketsBot method), 5
set_neck_joystick_pos() (py_websockets_bot.WebsocketsBot method), 5
set_robot_config() (py_websockets_bot.WebsocketsBot method), 5
start_streaming_camera_images() (py_websockets_bot.WebsocketsBot method), 5
start_streaming_motion_vectors() (py_websockets_bot.WebsocketsBot method), 5
start_streaming_small_camera_images() (py_websockets_bot.WebsocketsBot method), 5
stop_streaming_camera_images() (py_websockets_bot.WebsocketsBot method), 5
stop_streaming_motion_vectors() (py_websockets_bot.WebsocketsBot method), 6
stop_streaming_small_camera_images() (py_websockets_bot.WebsocketsBot method), 6

U

update() (py_websockets_bot.WebsocketsBot method), 6

W

WebsocketsBot (class in py_websockets_bot), 3