ardrone.h File Reference

Go to the source code of this file.

Enumerations

enum **drone_camera** { **FRONT_CAMERA**, **BOTTOM_CAMERA** }

Functions

int **drone_connect** (void)
Establishes a connection between the drone and the Link. This function must be called before any other ardrone functions. More...

void **drone_disconnect** (void)

void **drone_calibrate** (void)
Calibrates the drone's accelerometers to understand what "flat" is. More...

int **get_drone_version** (void)

void **drone_takeoff** (void)
Makes the drone takeoff and stabilize itself. This command will return immeadiately. More...

void **drone_land** (void)
This function will be used to land the drone at its current position. More...

int **get_drone_battery** (void)
retrieves the cached battery value More...

void **drone_clear_position** ()
Clears the accumulated absolute x, y, and z positions of the AR.Drone. More...

float **get_drone_x** (void)
Retrieves the x value relative to the drones starting position. Negative values indicate the drone has moved to the left of it's starting position. More...

float **get_drone_y** (void)
Retrieves the y value relative to the drones starting position. Negative values indicate the drone has moved backwards from it's starting position. More...

float **get_drone_z** (void)
Retrieves the y value relative to the drones starting position. Negative values indicate the drone has moved down from it's starting position. More...

float **get_drone_x_velocity** (void)
Retrieves the current velocity in the right or left direction. More...

float **get_drone_y_velocity** (void)
Retrieves the current velocity in the forward or backwards direction. More...

float **get_drone_z_velocity** (void)
Retrieves the current velocity in the upward or downwards direction. More...

float **get_drone_pitch** (void)
Retrieves the current pitch of the AR.Drone, in degrees. More...

float **get_drone_roll** (void)
Retrieves the current roll of the AR.Drone, in degrees. More...

float **get_drone_altitude** (void)
Retrieves the current altitude of the AR.Drone, in meters. More...

float	**get_drone_yaw** (void) Retrieves the current rotation in the clockwise (positive) or counterclockwise (negative) direction. More...
int	**drone_camera_open** (enum **drone_camera** camera) Opens the AR.Drone's camera as the camera input device. You must use **camera_close()** once finished. More...
int	**set_drone_mac_address** (const char *const address) Sets the Drone's MAC **Address** Pair to be the given string. More...
int	**drone_pair** (void) Automatically detects the host MAC **Address** and pairs the drone with it. More...
int	**set_drone_ssid** (const char *const ssid)
void	**drone_move** (float x_tilt, float y_tilt, float z_vel, float yaw_vel) Tells the drone to move with the given parameters. More...
void	**drone_hover** (void) Tells the drone that it should stop moving and hover at its current location. More...
void	**set_drone_emergency_stop_enabled** (int enabled)
int	**get_drone_emergency_stop_enabled** (void)

Detailed Description

Author

Braden McDorman

Enumeration Type Documentation

enum drone_camera

Enumerator	
FRONT_CAMERA	
BOTTOM_CAMERA	

Function Documentation

void drone_calibrate (void)

Calibrates the drone's accelerometers to understand what "flat" is.

int drone_camera_open (enum drone_camera camera)

Opens the AR.Drone's camera as the camera input device. You must use **camera_close()** once finished.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Parameters
camera FRONT_CAMERA for the horizontal camera, BOTTOM_CAMERA for the vertical camera.

Returns
1 on success, 0 on failure

void drone_clear_position ( )

Clears the accumulated absolute x, y, and z positions of the AR.Drone.

Precondition
drone_connect must have been previously called to esǎtablish a connection to the drone.

See Also
get_drone_x
get_drone_y
get_drone_z

int drone_connect (void)

Establishes a connection between the drone and the Link. This function must be called before any other ardrone functions.

void drone_disconnect (void)

void drone_hover (void)

Tells the drone that it should stop moving and hover at its current location.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

void drone_land (void)

This function will be used to land the drone at its current position.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Postcondition
The drone should slowly descend to the ground from its current height.

void drone_move (float x_tilt, float y_tilt, float z_vel, float yaw_vel)

```
void drone_move ( float  x_tilt,
                  float  y_tilt,
                  float  z_vel,
                  float  yaw_vel
                )
```

Tells the drone to move with the given parameters.

Parameters

enable	A value indicating if movement is enabled. 0 - True 1 - False
x_tilt	A value from zero to one indicating the percentage of maximum tilt in the left or right direction negative values are left and positive values are right. Ex: -.5 means Half of the total tilt left
y_tilt	A value from zero to one indicating the percentage of maximum tilt in the forward or backward direction negative values are left and positive values are right. Ex: -.5 means Half of the total tilt backwards.
yaw_vel	A value indicating the rotational velocity of the dronein milieters per second
z_vel	A value indicating the change in altitude in milimeters per second

int drone_pair (void)

Automatically detects the host MAC **Address** and pairs the drone with it.

Returns
1 for success, 0 for failure

See Also
set_drone_mac_address

void drone_takeoff (void)

Makes the drone takeoff and stabilize itself. This command will return immeadiately.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Postcondition
The drone should reach its normal operating height

See Also
drone_takeoff_block

float get_drone_altitude (void)

Retrieves the current altitude of the AR.Drone, in meters.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
A float indicating the altitude of the AR.Drone in meters.

int get_drone_battery (void)

retrieves the cached battery value

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
An integer representing the current battery level

int get_drone_emergency_stop_enabled (void)

Gets the previously set emergency stop enabled flag.

See Also
set_drone_emergency_stop_enabled

Returns
1 if emergency stop is enabled, 0 otherwise

float get_drone_pitch (void)

Retrieves the current pitch of the AR.Drone, in degrees.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
A float indicating the pitch of the AR.Drone in degrees.

float get_drone_roll (void)

Retrieves the current roll of the AR.Drone, in degrees.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
A float indicating the roll of the AR.Drone in degrees.

int get_drone_version (void)

Returns
The version of the currently connected drone. For example, an AR.Drone 1 will return the integer 1. The value -1 is returned upon error.

float get_drone_x (void)

Retrieves the x value relative to the drones starting position. Negative values indicate the drone has moved to the left of it's starting position.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
x value away from the drones starting position in milimeters verify it is in fact milimeters

float get_drone_x_velocity (void)

Retrieves the current velocity in the right or left direction.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
A float indicating the millimeters per second

float get_drone_y (void)

Retrieves the y value relative to the drones starting position. Negative values indicate the drone has moved backwards from it's starting position.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
y value away from the drones starting position in milimeters

float get_drone_y_velocity (void)

Retrieves the current velocity in the forward or backwards direction.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
A float indicating the velocity in millimeters per second

float get_drone_yaw (void)

Retrieves the current rotation in the clockwise (positive) or counterclockwise (negative) direction.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
A float indicating the degrees rotated from the original orientation

float get_drone_z (void)

Retrieves the y value relative to the drones starting position. Negative values indicate the drone has moved down from it's starting position.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
z value away from the drones starting position in milimeters

float get_drone_z_velocity (void)

Retrieves the current velocity in the upward or downwards direction.

Precondition
drone_connect must have been previously called to establish a connection to the drone.

Returns
A float indicating the velocity in millimeters per second

void set_drone_emergency_stop_enabled (int enabled)

When developing programs for the AR.Drone, it is often useful to be able to "emergency land". This will turn the Link's side button into a dedicated AR.Drone "kill switch". Note that using side_button in conjunction with this function may result in undefined behavior.

Parameters
enabled 0 for off, 1 for on

int set_drone_mac_address (const char *const address)

Sets the Drone's MAC **Address** Pair to be the given string.

Parameters
macAddress A string representing the MAC **Address** to pair

Returns
1 for success, 0 for failure

See Also
drone_pair

int set_drone_ssid (const char *const ssid)

Sets the SSID of the Drone to the given ssid.

Attention
This setting will not take effect until the AR.Drone is restarted.

Returns
1 for success, 0 for failure.