SMARS robotlibrary

Release v1.0

Marcus Jacobs

Jun 27, 2024

CONTENTS:

CHAPTER ONE

API REFERENCE

This page contains auto-generated API reference documentation[1].

1.1 robotlibrary

1.1.1 Subpackages

robotlibrary.bluetooth

Submodules

robotlibrary.bluetooth.advertising

Module Contents

Functions

advertising_payload([limited_disc, br_edr, name, ...])	
decode_field(payload, adv_type)	
decode_services(payload)	

Attributes

_ADV_MAX_PAYLOAD	

`robotlibrary.bluetooth.advertising._ADV_MAX_PAYLOAD`

[1] Created with sphinx-autoapi

robotlibrary.bluetooth.advertising.**advertising_payload**(*limited_disc=False*, *br_edr=False*, *name=None*, *services=None*, *appearance=0*)

robotlibrary.bluetooth.advertising.**decode_field**(*payload*, *adv_type*)

robotlibrary.bluetooth.advertising.**decode_services**(*payload*)

robotlibrary.bluetooth.ble_flags

Module Contents

robotlibrary.bluetooth.ble_flags.**ADV_TYPE_FLAGS**

robotlibrary.bluetooth.ble_flags.**ADV_TYPE_NAME**

robotlibrary.bluetooth.ble_flags.**ADV_TYPE_UUID16_COMPLETE**

robotlibrary.bluetooth.ble_flags.**ADV_TYPE_UUID32_COMPLETE**

robotlibrary.bluetooth.ble_flags.**ADV_TYPE_UUID128_COMPLETE**

robotlibrary.bluetooth.ble_flags.**ADV_TYPE_UUID16_MORE**

robotlibrary.bluetooth.ble_flags.**ADV_TYPE_UUID32_MORE**

robotlibrary.bluetooth.ble_flags.**ADV_TYPE_UUID128_MORE**

robotlibrary.bluetooth.ble_flags.**ADV_TYPE_APPEARANCE**

robotlibrary.bluetooth.ble_flags.**ADV_TYPE_MANUFACTURER_DATA**

robotlibrary.bluetooth.ble_flags.**IRQ_CENTRAL_CONNECT**

robotlibrary.bluetooth.ble_flags.**IRQ_CENTRAL_DISCONNECT**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTS_WRITE**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTS_READ_REQUEST**

robotlibrary.bluetooth.ble_flags.**IRQ_SCAN_RESULT**

robotlibrary.bluetooth.ble_flags.**IRQ_SCAN_DONE**

robotlibrary.bluetooth.ble_flags.**IRQ_PERIPHERAL_CONNECT**

robotlibrary.bluetooth.ble_flags.**IRQ_PERIPHERAL_DISCONNECT**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_SERVICE_RESULT**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_SERVICE_DONE**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_CHARACTERISTIC_RESULT**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_CHARACTERISTIC_DONE**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_DESCRIPTOR_RESULT** robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_DESCRIPTOR_DONE**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_READ_RESULT**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_READ_DONE**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_WRITE_DONE**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_NOTIFY**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTC_INDICATE**

robotlibrary.bluetooth.ble_flags.**IRQ_GATTS_INDICATE_DONE**

robotlibrary.bluetooth.ble_flags.**IRQ_MTU_EXCHANGED**

robotlibrary.bluetooth.ble_flags.**IRQ_L2CAP_ACCEPT**

robotlibrary.bluetooth.ble_flags.**IRQ_L2CAP_CONNECT**

robotlibrary.bluetooth.ble_flags.**IRQ_L2CAP_DISCONNECT**

robotlibrary.bluetooth.ble_flags.**IRQ_L2CAP_RECV**

robotlibrary.bluetooth.ble_flags.**IRQ_L2CAP_SEND_READY**

robotlibrary.bluetooth.ble_flags.**IRQ_CONNECTION_UPDATE**

robotlibrary.bluetooth.ble_flags.**IRQ_ENCRYPTION_UPDATE**

robotlibrary.bluetooth.ble_flags.**IRQ_GET_SECRET**

robotlibrary.bluetooth.ble_flags.**IRQ_SET_SECRET**

robotlibrary.bluetooth.ble_flags.**GATTS_NO_ERROR**

robotlibrary.bluetooth.ble_flags.**GATTS_ERROR_READ_NOT_PERMITTED**

robotlibrary.bluetooth.ble_flags.**GATTS_ERROR_WRITE_NOT_PERMITTED**

robotlibrary.bluetooth.ble_flags.**GATTS_ERROR_INSUFFICIENT_AUTHENTICATION**

robotlibrary.bluetooth.ble_flags.**GATTS_ERROR_INSUFFICIENT_AUTHORIZATION**

robotlibrary.bluetooth.ble_flags.**GATTS_ERROR_INSUFFICIENT_ENCRYPTION**

robotlibrary.bluetooth.ble_flags.**PASSKEY_ACTION_NONE**

robotlibrary.bluetooth.ble_flags.**PASSKEY_ACTION_INPUT**

robotlibrary.bluetooth.ble_flags.**PASSKEY_ACTION_DISPLAY**

robotlibrary.bluetooth.ble_flags.**PASSKEY_ACTION_NUMERIC_COMPARISON**

robotlibrary.bluetooth.ble_flags.**ADV_IND**

robotlibrary.bluetooth.ble_flags.**ADV_DIRECT_IND**

robotlibrary.bluetooth.ble_flags.**ADV_SCAN_IND**

robotlibrary.bluetooth.ble_flags.**ADV_NONCONN_IND** robotlibrary.bluetooth.ble_flags.**SCAN_RSP**

robotlibrary.bluetooth.ble_flags.**FLAG_READ**

robotlibrary.bluetooth.ble_flags.**FLAG_WRITE_NO_RESPONSE**

robotlibrary.bluetooth.ble_flags.**FLAG_WRITE**

robotlibrary.bluetooth.ble_flags.**FLAG_NOTIFY**

robotlibrary.bluetooth.ble_services_definitions

Module Contents

robotlibrary.bluetooth.ble_services_definitions.**MOTOR_TX_UUID**

robotlibrary.bluetooth.ble_services_definitions.**MOTOR_RX_UUID**

robotlibrary.bluetooth.ble_services_definitions.**ULTRASONIC_UUID**

robotlibrary.bluetooth.ble_services_definitions.**INFRARED_UUID**

robotlibrary.bluetooth.ble_services_definitions.**ROBOT_UUID**

robotlibrary.bluetooth.ble_services_definitions.**MOTOR_RX** = ()

robotlibrary.bluetooth.ble_services_definitions.**MOTOR_TX** = ()

robotlibrary.bluetooth.ble_services_definitions.**ROBOT_SERVICE** = ()

robotlibrary.bluetooth.central

Module Contents

Classes

BLECentral	

class robotlibrary.bluetooth.central.**BLECentral**(*add_robot_stuff=False*)

_handle_scan(*data*)

_handle_connect(*data*)

_handle_disconnect(*data*)

_handle_services(*data*)

_on_service_discovery_complete(*data*)

_handle_characteristics(*data*)

_handle_read(*data*)

_irq(*event: int*, *data*)

register_irq(*event: int*, *func*)

scan()

register_read_callback(*uuid*, *callback*)

send(*service_uuid*, *char_uuid*, *data*)

is_connected()

robotlibrary.bluetooth.main_central

Module Contents

Functions

read(buffer)
main()

robotlibrary.bluetooth.main_central.**read**(*buffer*)

robotlibrary.bluetooth.main_central.**main**()

robotlibrary.bluetooth.main_peripheral

Module Contents

Functions

main()

robotlibrary.bluetooth.main_peripheral.**main**()

robotlibrary.bluetooth.parser

Module Contents

Functions

decode_motor(data)	
encode_motor(→ bytes)	

robotlibrary.bluetooth.parser.**decode_motor**(*data: bytes*)

robotlibrary.bluetooth.parser.**encode_motor**(*speed: int*, *turn: int*, *forward: bool*) → bytes

robotlibrary.bluetooth.peripheral

Module Contents

Classes

BLEPeripheral	

class robotlibrary.bluetooth.peripheral.**BLEPeripheral**(*name='Theo'*, *add_robot_stuff=False*)

register_irq(*event*, *func*)

_irq(*event*, *data*)

_handle_connect(*data*)

_handle_disconnect(*data*)

_handle_read(*data*)

send(*service_uuid*, *char_uuid*, *data*)

is_connected()

advertise(*interval_us=500000*)

register_read_callback(*uuid*, *callback*)

1.1.2 Submodules

robotlibrary.config

This defines the parameters for the joystick. Don't change if you don't know what you are doing.

Module Contents

robotlibrary.config.**JS_X_MEDIAN = 29940**

robotlibrary.config.**JS_Y_MEDIAN = 30510**

robotlibrary.config.**JS_MAX_DUTY = 65535**

robotlibrary.config.**JS_MIN_DUTY = 260**

This defines the parameters for the motors.

MAX_DUTY: Set to lower than the maximum not to overload the motors.

MIN_DUTY: You can leave this at 0. Set MIN_SPEED instead.

MIN_SPEED: Set this to a value slightly below the speed that sets the robot in motion.

MAX_SPEED: If you want another scale than 0-100, set the maximum here.

robotlibrary.config.**MAX_DUTY = 60000**

robotlibrary.config.**MIN_DUTY = 0**

robotlibrary.config.**MIN_SPEED = 0**

robotlibrary.config.**MAX_SPEED = 100**

This defines the waiting time for the debouncing of the buttons. Leave as it is if you don't know what it means.

robotlibrary.config.**DEBOUNCE_WAIT = 30**

Use these constants to check for white or black with the IR-sensor.

robotlibrary.config.**WHITE_DETECTED = 0**

robotlibrary.config.**BLACK_DETECTED = 1**

Motors and ultrasonic sensor must use consecutive pins. Use >None< if you don't use the sensor.

robotlibrary.config.**ML = 12**

robotlibrary.config.**MR = 14**

robotlibrary.config.**US = 16**

robotlibrary.config.**IR**

robotlibrary.config.**SERVO**

robotlibrary.infrared

Module Contents

Classes

IR	This class manages the IR-sensor. Write your code in Robot.ir_detected()

class robotlibrary.infrared.**IR**(*pinNo*, *robot*)

This class manages the IR-sensor. Write your code in Robot.ir_detected()

reset_detected(*t*)

obstacle(*pin*)

This is called on any change in the IR-sensor.

robotlibrary.joystick

Module Contents

Classes

Joystick

Attributes

joystick

class robotlibrary.joystick.**Joystick**(*x*, *y*, *b*)

reset(*t*)

button_handler(*pin*)

get_speed(*s*)

get_direction(*d*)

robotlibrary.joystick.**joystick**

`robotlibrary.motor`

Module Contents

Classes

Motor	This class manages the motor. Don't edit!

class `robotlibrary.motor.`**Motor**(*pinNo*)

This class manages the motor. Don’t edit!

set_speed(*s*)

Sets the speed of the motor. Checks for sensible input.

change_speed(*sc*)

This defines an offset to the speed in motor. It is used with the remote control to turn the robot.

reset_offset()

off()

set_forward(*forward*)

Sets the motor to forward or backward without changing the speed.

`robotlibrary.rc`

Module Contents

Classes

RC	This class represents the remote control with two rotary encoders and a slider to set the speed. Don't edit unless you know what you are doing.

Functions

main()

class `robotlibrary.rc.`**RC**

This class represents the remote control with two rotary encoders and a slider to set the speed. Don’t edit unless you know what you are doing.

read(*a*)

send(*t*)

rotary_changed(*change*)

This is called when the direction knob is turned to determine the turn or spin.

button()

This is the button click.

set_speed(*t*)

This calculates the speed between MIN_SPEED and MAX_SPEED that is sent to the robot.

robotlibrary.rc.**main**()

robotlibrary.robot

Module Contents

Classes

Robot	This is the central class which manages and uses all the other components of the robot. The parameters are defined in config.py

class robotlibrary.robot.**Robot**(*rc*)

This is the central class which manages and uses all the other components of the robot. The parameters are defined in config.py

_drive(*dir_l*, *dir_r*)

This abstracted driving function is only called locally by the other functions with better names. It accelerates and decelerates to make driving more natural. Do not call directly!!

_drive_instantly(*dir_l*, *dir_r*)

This abstracted driving function is only called locally by the other functions with better names. It sets the speed immediatly. Do not call directly!!

set_speed_instantly(*s*)

Sets the new speed immediately. Doesn't change the driving mode of the robot.

set_speed(*s*)

Sets the new speed and accelerates and decelerates. Doesn't change the driving mode of the robot.

set_forward(*f*)

Sets the direction of the robot. True means forward.

spin_right()

Spin right indefinitely.

spin_left()

Spin left indefinitely.

turn_right()

This turns the robot to the right without it spinning on the spot. Each call makes the turn steeper.

turn_left()

This turns the robot to the right without it spinning on the spot. Each call makes the turn steeper.

go_straight()

Lets the robot go straight on. Usually called when a turn shall end.

spin_before_obstacle(*distance*)

This spins until the distance to an obstacle is greater than the given parameter __distance__.

toggle_spin(*d*)

Toggle turn for the given duration. With each call the oppsoite direction(clockwise / anti-clockwise) is used.

random_spin(*d*)

Randomly turn for the given duration.

stop()

Stop the robot slowly by deceleration.

emergency_stop()

Stop the robot immediately.

ir_detected(*pin*, *pin_num*)

If implemented this method is called when the IR-sensor has detected a change. Fill in your code accordingly

get_dist()

Get the distance from the ultrasonic sensor.

set_angle(*a*)

If implemented, turn the servo motor with the ultrasonic sensor to the given angle.

get_smallest_distance()

This returns the angle of the ultrasonic sensor where it measured the smallest distance

get_longest_distance()

This returns the angle of the ultrasonic sensor where it measured the longest distance

robotlibrary.rotary

Module Contents

Classes

Rotary	This class deals with the rotary encoders for the remote control. Don't use directly or edit.

class robotlibrary.rotary.**Rotary**(*dt*, *clk*, *sw*, *rc*)

This class deals with the rotary encoders for the remote control. Don’t use directly or edit.

ROT_CW = 1

ROT_CCW = 2

SW_PRESS = 4

SW_RELEASE = 8

rotary_change(*pin*)

switch_detect(*pin*)

robotlibrary.servo

Module Contents

Classes

Servo	This class manages the servo motor that turns the ultrasonic sensor. You need a servo motor installed to get use out of this.

class robotlibrary.servo.**Servo**(*pin*)

This class manages the servo motor that turns the ultrasonic sensor. You need a servo motor installed to get use out of this. Don't use directly or edit.

set_angle(*a*)

If installed, the servor motor will set the angle of the ultrasonic sensor. 90° ist straight ahead.

_get_duty(*angle*)

Internal function. Calculates the PWM duty for the given angle.

robotlibrary.ultrasonic

Module Contents

Classes

Ultra	This class manages the ultrasonic sensor. It returns the distance to an obstacle in cm.

class robotlibrary.ultrasonic.**Ultra**(*pinNo*)

This class manages the ultrasonic sensor. It returns the distance to an obstacle in cm.

get_dist()

This returns the measured distance in cm. (float)

1.2 conf

1.2.1 Module Contents

`conf.project = 'SMARS robotlibrary'`

`conf.copyright = '2024, Marcus Jacobs'`

`conf.author = 'Marcus Jacobs'`

`conf.release = 'v1.0'`

`conf.extensions = []`

`conf.templates_path = ['_templates']`

`conf.exclude_patterns = []`

`conf.html_theme = 'alabaster'`

`conf.html_static_path = ['_static']`

`conf.autoapi_dirs = ['/home/marcus/Schule/Informatik/Robotik/SMARS/robotlibrary/',...`

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX