API

Tip

If you are having a hard time, you can always have a look at the examples page where the classes, methods and parameters are used in practice.

WebMethod

class `edurov.core.WebMethod`(*index_file, video_resolution='1024x768', fps=30, server_port=8000, debug=False, runtime_functions=None, custom_response=None*) [source]

Starts a video streaming from the rasparry pi and a webserver that can handle user input and other requests.

Parameters:

- **index_file** (*str*) – Absolute path to the frontpage of the webpage, must be called `index.html`. For more information, see Displaying the video feed.
- **video_resolution** (*str, optional*) – A string representation of the wanted video resolution in the format WIDTHxHEIGHT.
- **fps** (*int, optional*) – Wanted framerate, may not be achieved depending on available resources and network.
- **server_port** (*int, optional*) – The web page will be served at this port
- **debug** (*bool, optional*) – If set True, additional information will be printed for debug purposes.
- **runtime_functions** (*callable or list, optional*) – Should be a callable function or a list of callable functions, will be started as independent processes automatically. For more information, see Controlling motors (or anything).
- **custom_response** (*callable, optional*) – If set, this function will be called if default web server is not able to handle a GET request, should return a str or None. If returned value starts with `redirect=` followed by a path, the server will redirect the browser to this path. The callable must accept two parameters whereas the second one is the requested path. For more information, see Custom responses.

Examples

```
>>> import os
>>> from edurov import WebMethod
>>>
>>> file = os.path.join(os.path.dirname(__file__), 'index.html', )
>>> web_method = WebMethod(index_file=file)
>>> web_method.serve()
```

`serve`(*timeout=None*) [source]

Will start serving the web page defined by the index_file parameter

Parameters: **timeout** (*int*, *optional*) – if set, the web page will only be served for that many seconds before it automatically shuts down

Notes

This method will block the rest of the script.

ROVSyncer

class `edurov.sync.ROVSyncer` [source]

Holds all variables for ROV related to control and sensors

Examples

```
>>> import Pyro4
>>>
>>> with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
>>>   while rov.run:
>>>       print('The ROV is still running')
```

`actuator`

Dictionary holding actuator values

Getter:	Returns actuator values as dict
Setter:	Update actuator values with dict
Type:	dict

`run`

Bool describing if the ROV is still running

Getter:	Returns bool describing if the ROV is running

Setter: Set to False if the ROV should stop

Type: bool

`sensor`

Dictionary holding sensor values

Getter: Returns sensor values as dict

Setter: Update sensor values with dict

Type: dict

KeyManager

class `edurov.sync.KeyManager` [source]

Keeps control of all user input from keyboard.

Examples

```
>>> import Pyro4
>>>
>>> with Pyro4.Proxy("PYRONAME:KeyManager") as keys:
>>> with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
>>>     keys.set_mode(key='l', mode='toggle')
>>>     while rov.run:
>>>         if keys.state('up arrow'):
>>>             print('You are pressing the up arrow')
>>>         if keys.state('l'):
>>>             print('light on')
>>>         else:
>>>             print('light off')
```

Note

When using the methods below a **key identifier** must be used. Either the keycode (int) or the KeyASCII or Common Name (str) from the table further down on this page can be used. Using keycode is faster.

`arrow_dict`

Dictionary with the state of the keys *up arrow*, *down arrow*, *left arrow* and *right arrow*

`keydown`(*key*, *make_exception=False*) [source]

Call to simulate a keydown event

Parameters:
- **key** (*int or str*) – key identifier as described above
- **make_exception** (*bool*, *optional*) – As default an exception is raised if the key is not found, this behavior can be changed be setting it to *False*

`keyup`(*key*, *make_exception=False*) [source]

Call to simulate a keyup event

Parameters:
- **key** (*int or str*) – key identifier as described above
- **make_exception** (*bool*, *optional*) – As default an exception is raised if the key is not found, this behavior can be changed be setting it to *False*

`qweasd_dict`

Dictionary with the state of the letters q, w, e, a, s and d

`set`(*key*, *state*) [source]

Set the state of the key to True or False

Parameters:
- **key** (*int or str*) – key identifier as described above
- **state** (*bool*) – *True* or *False*

`set_mode`(*key*, *mode*) [source]

Set the press mode for the key to *hold* or *toggle*

Parameters:
- **key** (*int or str*) – key identifier as described above
- **mode** (*str*) – *hold* or *toggle*

`state`(*key*) [source]

Returns the state of *key*

Parameters: **key** (*int or str*) – key identifier as described above

Returns: **state** – *True* or *False*

Return type: bool

Keys table

```
KeyASCII      ASCII   Common Name           Keycode
K_BACKSPACE   \b      backspace             8
K_TAB         \t      tab                   9
K_CLEAR               clear
K_RETURN      \r      return                13
K_PAUSE               pause
K_ESCAPE      ^[      escape                27
K_SPACE               space                 32
K_EXCLAIM     !       exclaim
K_QUOTEDBL    "       quotedbl
K_HASH        #       hash
K_DOLLAR      $       dollar
K_AMPERSAND   &       ampersand
K_QUOTE               quote
K_LEFTPAREN   (       left parenthesis
K_RIGHTPAREN  )       right parenthesis
K_ASTERISK    *       asterisk
K_PLUS        +       plus sign
K_COMMA       ,       comma
K_MINUS       -       minus sign
K_PERIOD      .       period
K_SLASH       /       forward slash
K_0           0       0                     48
K_1           1       1                     49
K_2           2       2                     50
K_3           3       3                     51
K_4           4       4                     52
K_5           5       5                     53
K_6           6       6                     54
K_7           7       7                     55
K_8           8       8                     56
K_9           9       9                     57
K_COLON       :       colon
K_SEMICOLON   ;       semicolon
K_LESS        <       less-than sign
K_EQUALS      =       equals sign
K_GREATER     >       greater-than sign
K_QUESTION    ?       question mark
K_AT          @       at
K_LEFTBRACKET [       left bracket
K_BACKSLASH   \       backslash
K_RIGHTBRACKET ]      right bracket
K_CARET       ^       caret
K_UNDERSCORE  _       underscore
K_BACKQUOTE   `       grave
K_a           a       a                     65
K_b           b       b                     66
K_c           c       c                     67
K_d           d       d                     68
K_e           e       e                     69
K_f           f       f                     70
K_g           g       g                     71
K_h           h       h                     72
K_i           i       i                     73
K_j           j       j                     74
K_k           k       k                     75
K_l           l       l                     76
K_m           m       m                     77
K_n           n       n                     78
K_o           o       o                     79
K_p           p       p                     80
K_q           q       q                     81
K_r           r       r                     82
K_s           s       s                     83
K_t           t       t                     84
K_u           u       u                     85
K_v           v       v                     86
K_w           w       w                     87
K_x           x       x                     88
```

```
K_y           y       y                     89
K_z           z       z                     90
K_DELETE              delete
K_KP0                 keypad 0
K_KP1                 keypad 1
K_KP2                 keypad 2
K_KP3                 keypad 3
K_KP4                 keypad 4
K_KP5                 keypad 5
K_KP6                 keypad 6
K_KP7                 keypad 7
K_KP8                 keypad 8
K_KP9                 keypad 9
K_KP_PERIOD   .       keypad period
K_KP_DIVIDE   /       keypad divide
K_KP_MULTIPLY *       keypad multiply
K_KP_MINUS    -       keypad minus
K_KP_PLUS     +       keypad plus
K_KP_ENTER    \r      keypad enter
K_KP_EQUALS   =       keypad equals
K_UP                  up arrow              38
K_DOWN                down arrow            40
K_RIGHT               right arrow           39
K_LEFT                left arrow            37
K_INSERT              insert                45
K_HOME                home                  36
K_END                 end                   35
K_PAGEUP              page up               33
K_PAGEDOWN            page down             34
K_F1                  F1
K_F2                  F2
K_F3                  F3
K_F4                  F4
K_F5                  F5
K_F6                  F6
K_F7                  F7
K_F8                  F8
K_F9                  F9
K_F10                 F10
K_F11                 F11
K_F12                 F12
K_F13                 F13
K_F14                 F14
K_F15                 F15
K_NUMLOCK             numlock
K_CAPSLOCK            capslock
K_SCROLLOCK           scrollock
K_RSHIFT              right shift
K_LSHIFT              left shift
K_RCTRL               right control
K_LCTRL               left control
K_RALT                right alt
K_LALT                left alt
K_RMETA               right meta
K_LMETA               left meta
K_LSUPER              left Windows key
K_RSUPER              right Windows key
K_MODE                mode shift
K_HELP                help
K_PRINT               print screen
K_SYSREQ              sysrq
K_BREAK               break
K_MENU                menu
K_POWER               power
K_EURO                Euro
```

Utilities

Different utility functions practical for ROV control

`edurov.utils.cpu_temperature()` [source]

Checks and returns the on board CPU temperature

Returns:	**temperature** – the temperature
Return type:	float

`edurov.utils.free_drive_space`(*as_string=False*) [source]

Checks and returns the remaining free drive space

Parameters:	**as_string** (*bool*, *optional*) – set to True if you want the function to return a formatted string. 4278 -> 4.28 GB
Returns:	**space** – the remaining MB in float or as string if *as_string=True*
Return type:	float or str

`edurov.utils.receive_arduino`(*serial_connection*) [source]

Returns a message received over *serial_connection*

Expects that the message received starts with a 6 bytes long number describing the size of the remaining data. “0x000bhello there” -> “hello there”.

Parameters:	**serial_connection** (*object*) – the `serial.Serial` object you want to use for receiving
Returns:	**msg** – the message received or None
Return type:	str or None

`edurov.utils.receive_arduino_simple`(*serial_connection*, *min_length=1*) [source]

Returns a message received over *serial_connection*

Same as `receive_arduino` but doesn’t expect that the message starts with a hex number.

Parameters:

- **serial_connection** (*object*) – the `serial.Serial` object you want to use for receiving
- **min_length** (*int*, *optional*) – if you only want that the function to only return the string if it is at least this long.

Returns: **msg** – the message received or None

Return type: str or None

`edurov.utils.send_arduino(`*msg, serial_connection*`)` [source]

Send the *msg* over the *serial_connection*

Adds a hexadecimal number of 6 bytes to the start of the message before sending it. “hello there” -> “0x000bhello there”

Parameters:

- **msg** (*str or bytes*) – the message you want to send
- **serial_connection** (*object*) – the `serial.Serial` object you want to use for sending

`edurov.utils.send_arduino_simple(`*msg, serial_connection*`)` [source]

Send the *msg* over the *serial_connection*

Same as `send_arduino`, but doesn’t add anything to the message before sending it.

Parameters:

- **msg** (*str or bytes*) – the message you want to send
- **serial_connection** (*object*) – the `serial.Serial` object you want to use for sending

`edurov.utils.serial_connection(`*port='/dev/ttyACM0', baudrate=115200, timeout=0.05*`)` [source]

Establishes a serial connection

Parameters:

- **port** (*str, optional*) – the serial port you want to use
- **baudrate** (*int, optional*) – the baudrate of the serial connection
- **timeout** (*float, optional*) – read timeout value

Returns: **connection** – a `serial.Serial` object if successful or None if not

Return type: class or None