
ArduRPC Python Documentation

Release 0.3

DinoTools.org

August 02, 2014

1	ArduRPC	3
1.1	Requirements	3
1.2	Installation	3
2	Getting Started	5
2.1	Initialize the library	5
2.2	Setup a connection	5
2.3	Use the ArduRPC interface	5
3	Examples	7
3.1	Basic Example	7
3.2	Pixel Strip	8
4	License	9
5	Changelog	13
5.1	0.3.0 - 02.08.2014	13
5.2	0.2.0 - 21.02.2014	13
5.3	0.1.0 - no public release	13
6	API Documentation	15
6.1	ArduRPC	15
6.2	Handler	17
7	Indices and tables	21
	Python Module Index	23

ArduRPC is an extensible library to control microcontroller boards like [Arduino](#) using [Python](#) and the [ArduRPC](#) protocol.

1.1 Requirements

- Python 3.x (maybe 2.x but untested)
- pySerial

1.2 Installation

1.2.1 From PyPI

ArduRPC can be install from PyPI as described below.

```
pip install ardurpc
```

1.2.2 From source

Download the archive from <https://github.com/DinoTools/ArduRPC-python>

```
python setup.py install
```

Getting Started

2.1 Initialize the library

Load all included handlers to enable auto detection:

```
>>> import ardurpc
>>> ardurpc.load_handlers()
```

2.2 Setup a connection

First off all setup a serial connection:

```
>>> import serial
>>> ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)
```

Use the serial connection and initialize ArduRPC:

```
>>> rpc = ardurpc.ArduRPC(serial=ser)
```

2.3 Use the ArduRPC interface

Get the supported protocol version. This should be 0:

```
>>> rpc.getProtocolVersion()
```

Get the version of the ArduRPC library on the device. This should be a tuple with three elements:

```
>>> rpc.getLibraryVersion()
```

Get a list of all handler names available on the device:

```
>>> rpc.get_handler_names()
```

Get a handler named 'neopixel':

```
>>> handler = rpc.get_handler_by_name("neopixel")
```

Get the number of available pixels of the NeoPixel strip:

```
>>> handler.getPixelCount()
```

Additional examples can be found in the examples directory.

3.1 Basic Example

This is a very basic example to get started. It just connects to an Arduino running ArduRPC and extract some basic information.

```
#!/usr/bin/env python
"""
Basic example.

Connects to an Arduino using the ArduRPC protocol and displays some information.

Change the connect() function to reflect your settings.
"""

import ardurpc
from ardurpc.connector import Serial, UDP

def connect():
    # Connect to the serial port
    con = Serial("/dev/ttyACM0", 9600)

    # More examples:
    # con = Serial("/dev/ttyUSB0", 9600)
    # con = UDP(host="192.168.1.1", port=1234)

    # New instance
    rpc = ardurpc.ArduRPC(connector=con)

    print("Version(Protocol): {0}".format(rpc.getProtocolVersion()))
    print(
        "Version(Library): {0}".format(
            ".".join([str(i) for i in rpc.getLibraryVersion()])
        )
    )
    print(
        "Available handlers: {0}".format(
            ", ".join(rpc.get_handler_names())
        )
    )
```

```
)  
  
    return rpc  
  
if __name__ == "__main__":  
    connect()
```

3.2 Pixel Strip

This examples uses the connect() function from the basic example.

```
#!/usr/bin/env python  
"""  
Control a pixel strip  
  
This example uses the connect() function from the Basic example.  
"""  
  
from basic import connect  
  
def run():  
    # use the basic example  
    rpc = connect()  
  
    # Get a handler named 'strip'  
    handler = rpc.get_handler_by_name("strip")  
    if handler is None:  
        print("A handler with the given name does not exist")  
  
    # Get pixel count  
    pixel_count = handler.getPixelCount()  
    print("Strip has {0} pixels".format(pixel_count))  
  
    # Set color to red  
    for i in range(0, pixel_count):  
        handler.setPixelColor(i, (255, 0, 0))  
  
    # Set color to green  
    for i in range(0, pixel_count):  
        handler.setPixelColor(i, (0, 255, 0))  
  
    # Set color to blue  
    for i in range(0, pixel_count):  
        handler.setPixelColor(i, (0, 0, 255))  
  
if __name__ == "__main__":  
    run()
```

License

The code is licensed under the terms of GNU Lesser General Public License.

GNU LESSER GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <<http://fsf.org/>>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates
the terms and conditions of version 3 of the GNU General Public
License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser
General Public License, and the "GNU GPL" refers to version 3 of the GNU
General Public License.

"The Library" refers to a covered work governed by this License,
other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided
by the Library, but which is not otherwise based on the Library.
Defining a subclass of a class defined by the Library is deemed a mode
of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an
Application with the Library. The particular version of the Library
with which the Combined Work was made is also called the "Linked
Version".

The "Minimal Corresponding Source" for a Combined Work means the
Corresponding Source for the Combined Work, excluding any source code
for portions of the Combined Work that, considered in isolation, are
based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the
object code and/or source code for the Application, including any data
and utility programs needed for reproducing the Combined Work from the
Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

- a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
- b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser

General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

Changelog

5.1 0.3.0 - 02.08.2014

- Support Text-LCD handlers
- Support handler for Arduino based boards
- Handle new return codes (ArduRPC 0.4)
- New Bluetooth connector
- New UDP connector

5.2 0.2.0 - 21.02.2014

- Public release
- Support ArduRPC 0.3
- Add examples

5.3 0.1.0 - no public release

- Initial version

API Documentation

6.1 ArduRPC

class `ardurpc.ArduRPC` (*handlers=None, **kwargs*)

Functions with camel-case names are directly mapped to a function on the device. Functions without camel-case names are not mapped. The second function type should be preferred.

getHandlerList ()

Get a handler list.

Function 'get_handlers()' should be preferred.

Returns Raw value returned by the device

getHandlerName (*handler_id*)

Get a handler name by its ID.

Parameters **handler_id** (*Integer*) – The ID of the handler

Returns The name of the handler

Return type String

getLibraryVersion ()

Get the version of the ArduRPC library running on the device.

Returns Raw value returned by the device

getMaxPacketSize ()

Get the max packet size supported by the device.

Returns Raw value returned by the device

getProtocolVersion ()

Get supported protocol version.

Returns Value returned by the device

get_handler (*handler_id*)

Return a instance of the handler.

Parameters **handler_id** (*Integer*) – The ID of the handler

Returns Class instance or None

Return type Instance

get_handler_by_name (*name*)

Return a instance of the handler.

Parameters **name** (*String*) – The name of the handler

Returns Class instance or None

Return type Instance

get_handler_cache ()

Get a list of all cached handlers.

Returns Dict: Key = Name, Value = Instance

Return type Dict

get_handler_names ()

Return a list of handler names present on the device.

Returns List of names

Return type List

get_handlers ()

Get a list of all available handlers.

Returns Dict: Key = Name, Value = Instance

Return type Dict

get_version ()

Get the version of the ArduRPC library running on the device.

Returns A Dict with major, minor and patch level

Return type Dict

`ardurpc.load_handlers` ()

Load build-in handlers.

`ardurpc.register` (*handler_type*, *handler*, *mask=16*)

Register a new handler.

Parameters

- **handler_type** (*Integer*) – The ID of the handler type
- **handler** (*Class*) – The handler class (Not an instance)
- **mask** (*Integer*) – The mask to group handlers

ArduRPC exceptions.

- Failure
 - FunctionNotFound
 - HandlerNotFound
 - CommandNotFound
- Timeout

exception `ardurpc.exception.ArduRPCException` (*value=''*)

Base ArduRPC Exception.

__weakref__

list of weak references to the object (if defined)

exception `ardurpc.exception.CommandNotFound` (*value=''*)
 The command is not available on the microcontroller.

exception `ardurpc.exception.Failure` (*value=''*)
 Something went wrong while executing a command.
 But no reason was given.

exception `ardurpc.exception.FunctionNotFound` (*value=''*)
 The function is not available on the microcontroller.

exception `ardurpc.exception.HandlerNotFound` (*value=''*)
 The handler is not available on the microcontroller.

exception `ardurpc.exception.InvalidHeader` (*value=''*)
 The header was malformed

exception `ardurpc.exception.InvalidRequest` (*value=''*)
 The request was malformed

exception `ardurpc.exception.Timeout` (*value=''*)
 A timeout occurred.

exception `ardurpc.exception.UnknownReturnCode` (*value=''*)

6.2 Handler

6.2.1 Text-LCD

6.2.2 Matrix

class `ardurpc.handler.matrix.Base` (***kwargs*)

Handler for the Base Matrix type

drawLine (*x0, y0, x1, y1, color*)

Draw a line.

drawPixel (*x, y, color*)

Draw a pixel.

Parameters

- **x** (*Integer*) – X-Position
- **y** (*Integer*) – Y-Position
- **color** (*IntegerTuple*) – The color

fillScreen (*color*)

Fill the screen with the given color.

getColorCount ()

Get the color count.

Returns Number of colors

Return type Integer

getHeight ()

Get the height in pixels

Returns Height

Return type Integer

getWidth ()

Get the width in pixels.

Returns Width

Return type Integer

class `ardurpc.handler.matrix.Extended` (**kwargs)

Handler for the Extended Matrix type

drawChar (*x, y, c, color, bg, size*)

drawCircle (*x, y, radius, color*)

drawFastHLine (*x, y, w, color*)

drawFastVLine (*x, y, h, color*)

drawImage (*x, y, image, encoding=2*)

drawRect (*x, y, w, h, color*)

drawRoundRect (*x, y, w, h, radius, color*)

drawTriangle (*x0, y0, x1, y1, x2, y2, color*)

fillCircle (*x, y, radius, color*)

fillRect (*x, y, w, h, color*)

fillRoundRect (*x, y, w, h, radius, color*)

fillTriangle (*x0, y0, x1, y1, x2, y2, color*)

invertDisplay (*i*)

setAutoSwapBuffers (*auto_swap=True*)

setCursor (*x, y*)

setRotation (*rotation*)

setTextColor (*color, bg=None*)

setTextSize (*size*)

setTextWrap (*wrap*)

swapBuffers (*copy=True*)

write (*s*)

6.2.3 Strip

class `ardurpc.handler.strip.Base` (**kwargs)

Wrapper.

getColorCount ()

Return the number of colors.

Returns Number of colors 1, 2 or 3

Return type Integer

getPixelCount ()

Return the number of pixels.

Returns Number of pixels

Return type Integer

setPixelColor (*n*, *color*)

Set the color of a pixel.

Parameters

- **n** (*Integer*) – The pixel index (0 - (16²)-1)
- **color** (*List/Integer*) – The color

Returns command result

setRangeColor (*start*, *end*, *color*)

Set the color of a pixel.

Parameters

- **start** (*Integer*) – The pixel to start
- **end** (*Integer*) – The pixel to stop
- **color** (*List/Integer*) – The color

Returns command result

class ardurpc.handler.strip.**Extended** (**kwargs)
Wrapper.

show ()

Transmit the current values to the LEDs.

class ardurpc.handler.**Handler** (*connector=None*, *handler_id=None*, *name=None*)
Handler base class.

__weakref__

list of weak references to the object (if defined)

_call (*command_id*, *fmt=None*, **data*)

Execute a command on the microcontroller.

Parameters

- **command_id** (*Integer*) – The ID of the command
- **fmt** (*String*) – The format of the data
- **data** – Parameters for the command

Returns Returns the result

Indices and tables

- *genindex*
- *modindex*
- *search*

a

`ardurpc`, 15

`ardurpc.exception`, 16

`ardurpc.handler`, 19