ArduRPC Documentation

Release 0.4.0

PhiBo (DinoTools.org)

March 15, 2015

Contents

ArduRPC brings remote procedure calls(RPC) to microcontrollers. It has been developed for Arduino based projects but can also be used with other platforms. The protocol has been designed to be simple and flexible.

Users:

CHAPTER 1

Getting started

1.1 Setup

First of all download the base library from the ArduRPC repository and extract the ArduRPC directory into your Arduino library path.

After that you can download additional handlers from the ArduRPC handler repository or from any other source. Copy the handler files to your Arduino library path. Make sure you have downloaded and installed all dependencies before using a handler.

Example: Adafruit_NeoPixel

1. Download base library
   - Extract the files
   - Copy the ArduRPC directory to your Arduino library path
2. Download the ArduRPC handler repository
   - Extract the files
   - Copy the ArduRPC_Adafruit_NeoPixel directory to your Arduino library path
3. Download the Adafruit_NeoPixel library
   - Extract the files
   - Copy the Adafruit_NeoPixel directory to you Arduino library path.

1.2 Usage

The easiest way to get started is to use one of the available examples and to modify only the required parts.

To use the Adafruit_NeoPixel library with ArduRPC follow the examples below.

```
#include <Adafruit_NeoPixel.h>
#include <ArduRPC.h>
#include <ArduRPC_Adafruit_NeoPixel.h>

Adafruit_NeoPixel strip = Adafruit_NeoPixel(14, 6, NEO_GRB + NEO_KHZ800);

ArduRPC rpc = ArduRPC(2, 0);
ArduRPC_Serial rpc_serial = ArduRPC_Serial(Serial, rpc);

```

```
10 ArduRPC_Adafruit_Neopixel Strip_Wrapper(rpc, "strip", strip);
11
12 void setup() {
13   Serial.begin(9600);
14
15   strip.begin();
16   strip.show();
17 }
18
19 void loop() {
20   rpc_serial.loop();
21 }
```

Line 1: Include the Adafruit_NeoPixel library.

Line 2: Include the ArduRPC base library.

Line 3: Include the ArduRPC handler for the Adafruit_NeoPixel.

Line 5: Initialize the NeoPixel strip.

Line 7: Initialize the RPC manager with a limit of 2 handlers and 0 functions.

Line 8: Initialize the RPC data processor for a serial communication.

Line 10: Initialize the handler for the strip and name it 'strip'.

Line 13: Initialize the serial port and set baud to 9600.

Line 15 and 16: Setup the strips.

Line 18: Run the ArduRPC processing loop.

1.3 Additional examples

At least one example should be included with every library/handler.

CHAPTER 2

Handler

A handler provides one or more callable functions. Before the functions are accessible over the remote procedure call interface the handler must be connected to the system. A handler can be identified by the handler type.

2.1 Handler types

A handler type belongs always to a group with a base type. Functions defined in the base type must be available in all subtypes. This ensures that a client with support for a base type can at least access the basic functionality of a handler even if that specific subtype isn't supported.

The development process of an API can be classified by three states:

Experimental: Currently under development and the API might be changed in the next release.

Beta: The API should be changed only if there is a good reason for this.

Stable: The existing API must not be changed.

To register a new handler type feel free to open a new request by using the issue tracker on github.

Start	End	Mask	Name	Status
0x0100	0x01FF	8	*Base Pixel Strip*	Beta
0x0180	0x01FF	9	*Extended Pixel Strip*	Experimental
0x0200	0x02FF	8	*Base Matrix*	Beta
0x0280	0x02FF	9	*Extended Matrix*	Beta
0x0300	0x03FF	8	*Base Text-LCD*	Experimental
0x0380	0x03FF	8	*Extended Text-LCD*	Experimental
0x0400	0x04FF	8	*Base Sensor*	Experimental
0x0401			*Temperature Sensor*	Experimental
0x0402			*Humidity Sensor*	Experimental
0x0403			*Temperature-Humidity Sensor*	Experimental
0x0500	0x05FF	8	*Base Board*	Experimental
0x0501		16	*Arduino Board*	Experimental
0xFF00	0xFFFF	8	*Custom handlers*	n/a

2.2 Base types

2.2.1 Base/Extended Pixel Strip

ID	Function	Strip Type	
		Base	Extended
0x01	`pixel_strip::getColorCount()`	X	x
0x02	`pixel_strip::getPixelCount()`	X	x
0x11	`pixel_strip::setPixelColor()`	X	x
0x12	`pixel_strip::setRangeColor()`	X	x

uint8_t `pixel_strip::`**`getColorCount`**`()`
Get the number of colors. Return value should be 1, 2 or 3.

Returns Number of colors.

uint16_t `pixel_strip::`**`getPixelCount`**`()`
Get the number of available pixels.

Returns Number of pixels

void `pixel_strip::`**`setPixelColor`**(uint16_t *n*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Set the color of a pixel. All color values MUST be given and spare colors will be ignored by the device.

Parameters

- **n** – The number of the LED. Range from 0 to pixel_count - 1
- **color1** – First color. Red if color_count = 3.
- **color2** – Second color. Green if color_count = 3.
- **color3** – Third color. Blue if color_count = 3.

void `pixel_strip::`**`setRangeColor`**(uint16_t *start*, uint16_t *end*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Set the color of a range of pixels.

Parameters

- **start** – The position to start. Range from 0 to pixel_count - 1
- **end** – The position to stop. Range from start to pixel_count - 1
- **color1** – First color. Red if color_count = 3.
- **color2** – Second color. Green if color_count = 3.
- **color3** – Third color. Blue if color_count = 3.

2.2.2 Base/Extended Matrix

The Base Matrix handler is inspired by the Adafruit_GFX library and it is intended to be used with libraries based on Adafruit_GFX. But it might also be possible to wrap any other library.

ID	Function	Matrix Type	
		Base	Extended
0x01	matrix_gfx::getColorCount()	X	X
0x02	matrix_gfx::getWidth()	X	X
0x03	matrix_gfx::getHeight()	X	X
0x11	matrix_gfx::drawPixel()	X	X
0x21	matrix_gfx::drawLine()	X	X
0x22	matrix_gfx::drawFastVLine()		X
0x23	matrix_gfx::drawFastHLine()		X
0x24	matrix_gfx::drawRect()		X
0x25	matrix_gfx::fillRect()		X
0x26	matrix_gfx::fillScreen()	X	X
0x27	matrix_gfx::invertDisplay()		X
0x31	matrix_gfx::drawCircle()		X
0x32	matrix_gfx::fillCircle()		X
0x33	matrix_gfx::drawTriangle()		X
0x34	matrix_gfx::fillTriangle()		X
0x35	matrix_gfx::drawRoundRect()		X
0x36	matrix_gfx::fillRoundRect()		X
0x41	matrix_gfx::drawChar()		X
0x42	matrix_gfx::setCursor()		X
0x43	matrix_gfx::setTextColor()		X
0x44	matrix_gfx::setTextColor()		X
0x45	matrix_gfx::setTextSize()		X
0x46	matrix_gfx::setTextWrap()		X
0x47	matrix_gfx::write()		X
0x51	matrix_gfx::setRotation()		X
0x52	matrix_gfx::swapBuffers()		X
0x53	matrix_gfx::setAutoSwapBuffers()		X
0x61	matrix_gfx::drawImage()		X

uint8_t matrix_gfx::**getColorCount**()
Get the number of colors. Return value should be 1, 2 or 3.

Returns Number of colors.

uint16_t matrix_gfx::**getWidth**()
Get width in pixels.

Returns Number of pixels

uint16_t matrix_gfx::**getHeight**()
Get height in pixels.

Returns Number of pixels

void matrix_gfx::**drawPixel**(int16_t *x*, int16_t *y*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw a pixel.

Parameters

- **x** – Pixel x position
- **y** – Pixel y position
- **color1** – First color. Red if color_count = 3.
- **color2** – Second color. Green if color_count = 3.
- **color3** – Third color. Blue if color_count = 3.

void matrix_gfx::**drawLine**(int16_t *x0*, int16_t *y0*, int16_t *x1*, int16_t *y1*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw a line.

void matrix_gfx::**drawFastVLine**(int16_t *x*, int16_t *y*, int16_t *h*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw a vertical line.

void matrix_gfx::**drawFastHLine**(int16_t *x*, int16_t *y*, int16_t *w*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw a horizontal line.

void matrix_gfx::**drawRect**(int16_t *x*, int16_t *y*, int16_t *w*, int16_t *h*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw the boarder of rectangle.

void matrix_gfx::**fillRect**(int16_t *x*, int16_t *y*, int16_t *w*, int16_t *h*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw a filled rectangle.

void matrix_gfx::**fillScreen**(uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Fill the screen with the given color.

void matrix_gfx::**invertDisplay**(boolean *i*)
Invert the display.

void matrix_gfx::**drawCircle**(int16_t *x0*, int16_t *y0*, int16_t *r*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw the border of a circle.

void matrix_gfx::**fillCircle**(int16_t *x0*, int16_t *y0*, int16_t *r*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw a filled circle.

void matrix_gfx::**drawTriangle**(int16_t *x0*, int16_t *y0*, int16_t *x1*, int16_t *y1*, int16_t *x2*, int16_t *y2*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw the boarder of a triangle.

void matrix_gfx::**fillTriangle**(int16_t *x0*, int16_t *y0*, int16_t *x1*, int16_t *y1*, int16_t *x2*, int16_t *y2*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw a filled triangle.

void matrix_gfx::**drawRoundRect**(int16_t *x0*, int16_t *y0*, int16_t *w*, int16_t *h*, int16_t *radius*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw the boarder of a round rectangle.

void matrix_gfx::**fillRoundRect**(int16_t *x0*, int16_t *y0*, int16_t *w*, int16_t *h*, int16_t *radius*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Draw a filled round rectangle.

void matrix_gfx::**drawChar**(int16_t *x*, int16_t *y*, unsigned char *c*, uint8_t *color1*, uint8_t *color2*, uint8_t *color3*, uint16_t *bg*, uint8_t *size*)
Draw a character.

void matrix_gfx::**setCursor**(int16_t *x*, int16_t *y*)
Set the cursor position.

void matrix_gfx::**setTextColor**(uint8_t *color1*, uint8_t *color2*, uint8_t *color3*)
Set the text color.

void matrix_gfx::**setTextColor**(uint8_t *color1*, uint8_t *color2*, uint8_t *color3*, uint8_t *bg_red*, uint8_t *bg_green*, uint8_t *bg_blue*)
Set the text color.

void `matrix_gfx::`**`setTextSize`**(uint8_t *s*)
Set the text size.

void `matrix_gfx::`**`setTextWrap`**(boolean *w*)
Set the text wrap.

void `matrix_gfx::`**`setRotation`**(uint8_t *r*)
Set the rotation.

uint8_t `matrix_gfx::`**`swapBuffers`**(uint8_t *copy*)

Parameters copy 0 = False | 1 = True

Swap buffers and copy new front buffer into the back buffer.

uint8_t `matrix_gfx::`**`setAutoSwapBuffers`**(uint8_t *auto_swap*)

Parameters auto_swap 0 = False | 1 = True

Set option to swap buffers after each command.

void `matrix_gfx::`**`drawImage`**(int16_t *x*, int16_t *y*, int16_t *width*, int16_t *height*, uint8_t *color_encoding*, uint8_t* *image*)

Parameters

- **x** – x-position
- **y** – y-position
- **width** – Image width
- **height** – Image height
- **color_encoding** – The color encoding. For more information have a look at the list below.
- **image** – The image data

Color encoding:

Mode 0: 8-Bit encoding. From MSB to LSB:

- 2-Bit - red
- 3-Bit - green
- 3-Bit - blue

Mode 1: 16-Bit encoding. From MSB to LSB:

- 5-Bit - red
- 6-Bit - green
- 5-Bit - blue

Mode 2: 24-Bit encoding. From MSB to LSB:

- 8-Bit - red
- 8-Bit - green
- 8-Bit - blue

2.2.3 Base/Extended Text-LCD

ID	Function	Text-LCD Type	
		Base	Extended
0x01	`text_lcd::getWidth()`	X	X
0x02	`text_lcd::getHeight()`	X	X
0x11	`text_lcd::clear()`	X	X
0x12	`text_lcd::home()`	X	X
0x13	`text_lcd::setCursor()`	x	X
0x21	`text_lcd::write()`	X	X
0x22	`text_lcd::print()`	X	X

uint8_t `text_lcd::`**`getWidth`**`()`
Get the width as number of characters.

uint8_t `text_lcd::`**`getHeight`**`()`
Get the height as number of characters.

void `text_lcd::`**`clear`**`()`
Clear the LCD screen and set the cursor position to the upper-left corner.

void `text_lcd::`**`home`**`()`
Set the cursor position to the upper-left corner.

void `text_lcd::`**`setCursor`**`(`uint8_t *col*, uint8_t *row*`)`

Parameters

- **col** – The column
- **row** – The row

Position the cursor.

void `text_lcd::`**`write`**`(`char *c*`)`

Parameters c The character to display

Print a single character to the LCD.

void `text_lcd::`**`print`**`(`uint8_t *num*, char *text*[]`)`

Parameters

- **num** – Number of characters
- **text** – The text to display

Print text to the LCD.

2.2.4 Base Sensor

2.2.5 Temperature/Humidity Sensor

The Temperature and the Humidity Sensors share the same API. Temperatures are always in Celsius.

ID	Function
0x11	`sensor_temperature::getMinValue()`
0x12	`sensor_temperature::getMaxValue()`
0x13	`sensor_temperature::getAccuracy()`
0x14	`sensor_temperature::getValue()`

float `sensor_temperature::getMinValue()`
Get the value of the lowest possible temperature/humidity measured by the sensor.

float `sensor_temperature::getMaxValue()`
Get the value of the highest possible temperature/humidity measured by the sensor.

float `sensor_temperature::getAccuracy()`
Get the best accuracy of the measured values.

float `sensor_temperature::getValue()`
Get the current temperature/humidity.

2.2.6 Temperature-Humidity Sensor

ID	Function
0x11	`sensor_temp_humidity::getMinTempValue()`
0x12	`sensor_temp_humidity::getMaxTempValue()`
0x13	`sensor_temp_humidity::getTempAccuracy()`
0x14	`sensor_temp_humidity::getTemperature()`
0x21	`sensor_temp_humidity::getMinHumidityValue()`
0x22	`sensor_temp_humidity::getMaxHumidityValue()`
0x23	`sensor_temp_humidity::getHumidityAccuracy()`
0x24	`sensor_temp_humidity::getHumidity()`

float `sensor_temp_humidity::getMinTempValue()`
Get the value of the lowest possible temperature measured by the sensor.

float `sensor_temp_humidity::getMaxTempValue()`
Get the value of the highest possible temperature measured by the sensor.

float `sensor_temp_humidity::getTempAccuracy()`
Get the best accuracy of the measured temperature.

float `sensor_temp_humidity::getTemperature()`
Get the current temperature.

float `sensor_temp_humidity::getMinHumidityValue()`
Get the value of the lowest possible humidity measured by the sensor.

float `sensor_temp_humidity::getMaxHumidityValue()`
Get the value of the highest possible humidity measured by the sensor.

float `sensor_temp_humidity::getHumidityAccuracy()`
Get the best accuracy of the measured humidity.

float `sensor_temp_humidity::getHumidity()`
Get the current humidity.

2.2.7 Base Board

At the moment only Arduino based boards are supported.

2.2.8 Arduino Board

A handler of this type is used to control Arduino based boards.

ID	Function
0x11	board_arduino::getAnalogInput()
0x21	board_arduino::getDigitalInput()
0x22	board_arduino::setDigitalOutput()
0x31	board_arduino::setPWMOutput()

uint16 board_arduino::**getAnalogInput**(uint8_t *pin*)
Get the value of an analog input pin.

Parameters pin Pin number

uint16 board_arduino::**getDigitalInput**(uint8_t *pin*)
Get the value of an digital input pin.

Parameters pin Pin number

uint16 board_arduino::**setDigitalOutput**(uint8_t *pin*, uint8_t *value*)
Set the value of a digital output pin.

Parameters

- **pin** – Pin number
- **value** – The value (0, 1)

uint16 board_arduino::**setPWMOutput**(uint8_t *pin*, uint8_t *value*)
Set the value of a PWM output pin.

Parameters

- **pin** – Pin number
- **value** – The PWM value (0-255)

2.2.9 Custom handlers

This range of IDs is reserved for custom handlers e.g. for testing or prototyping purposes.

CHAPTER 3

Buildin system handler

The system handler is available by default and is included in the base ArduRPC library.

3.1 Function overview

ID	Function
0x01	`getProtocolVersion()`
0x02	`getLibraryVersion()`
0x03	`getMaxPacketSize()`
0x10	`getFunctionList()`
0x20	`getHandlerList()`
0x21	`getHandlerName()`

3.2 Function details

uint8_t **`getProtocolVersion`**`()`
Get the protocol version. At the moment this should be 0.

RPC_ARRAY **`getLibraryVersion`**`()`
Return the library version as RPC_ARRAY with three elements of uint8 type.

1.Major version

2.Minor version

3.Patch level

uint16_t **`getMaxPacketSize`**`()`
Return the maximum packet size (header + data) in bytes.

RPC_VARRAY **`getFunctionList`**`()`
Get a list of all functions. The result has 2 columns.

1.The first column is a unsigned char and represents the internal ID.

2.The second column is a unsigned char and represents the type of the function.

RPC_VARRAY **`getHandlerList`**`()`
Get a list of all handlers. The result has 2 columns.

1.The first column is a unsigned char and represents the internal ID of the handler.

2.The second column is an unsigned short and represents the type of the handler.

RPC_CARRAY **getHandlerName** (uint8_t *handler_id*)
Get the handler name by a given ID.

CHAPTER 4

Additional handlers

Developers:

CHAPTER 5

Protocol

ArduRPC brings remote procedure calls to microcontrollers. The binary protocol has been designed to be simple and flexible. Basic and also complex data types are supported.

5.1 Communication

5.1.1 Request

Name	Type	Comment
Version	`uint8`	Protocol version (default: 0)
Handler ID	`uint8`	ID of the handler
Command ID	`uint8`	ID of the command to call
Length	`uint8`	Length of data in bytes
Data		List of parameters

Version: This is the version of the protocol. At the moment only version 0 is supported.

Handler ID: The ID of the handler to use.

Command ID: The ID of the command to call.

Length: Length of the data in bytes.

Data: A list of parameters. See Data Types for more information.

5.1.2 Response

Name	Type	Comment
Return code	`uint8`	The return code. See *Return codes*
Data	Mixed	The result

Data: The result data. Only one type of data is allowed.

5.2 Data Types

5.2.1 Basic data types

Basic data types have a fixed length like a signet Integer.

Data Types

Identifier	Data Type	Size in Bytes
0x00	None	0
0x01	Signed Char	1
0x02	Unsigned Char	1
0x03	Signet Short	2
0x04	Unsigned Short	2
0x05	Signed Long	4
0x06	Unsigned Long	4
0x07	Signed Long Long	8
0x08	Unsigned Long Long	8
0x09	Float [2]	4

Data structure:

Name	Type	Comment
Identifier	`uint8`	The identifier. See Data types for more information.
Data		The data specified by the identifier.

5.2.2 Complex data types

At the moment the following data types are supported.

Identifier	Data Type
0x10	Simple array
0x11	String
0x12	Multicolumn array
0x13	Value array

Array:

Name	Type	Comment
Identifier	`uint8`	Set to 0x10. See data types for more information.
Data Identifier	`uint8`	Basic data type of the elements.
Length	`uint8`	Length of the string in bytes.
Data		The string.

String:

Name	Type	Comment
Identifier	`uint8`	Set to 0x11. See data types for more information.
Length	`uint8`	Length of the string in bytes.
Data		The string.

Multi column array:

Name	Type	Comment
Identifier	`uint8`	Set to 0x12. See data types for more information.
Columns	`uint8`	Number of columns.
Column Identifier 1	`uint8`	The identifier for column 1
Column Identifier m	`uint8`	The identifier for column m
Length	`uint8`	Number of rows
Row 1		Data of row 1
Row n		Data of row n

[1] Float values MUST use the IEEE 754 binary32 representation format.
[2] Float values MUST use the IEEE 754 binary32 representation format.

Value array:

Name	Type	Comment
Identifier	`uint8`	Set to 0x13. See data types for more information.
Length	`uint8`	Length of the array in bytes.
Data		List of identifiers and values

5.3 Return codes

Code	Comment
0	Success
124	Function not found
125	Handler not found
126	Command not found
127	Failure no reason given

5.4 Example

5.4.1 Basic data types

Request:

Data	Comment
0x00	Protocol version (default: 0)
0x03	ID of the handler.
0x02	ID of the command to call
0x05	Length of data in bytes
0x10	Value: 16 (Type: uint8)
0x00 0x01	Value: 1 (Type: uint16)

Response:

Data	Comment
0x00	Success
0x01	Identifier for Unsigned Char
0x10	Value: 16

CHAPTER 6

Communication

6.1 Serial (Hex-Mode)

- Use a serial port like UART (Arduino: Serial or SoftwareSerial)
- Read/Write data line by line
- Every package/line must start with a colon (':')
- Lines without a colon must be ignored
- Data is encoded as hex string

Example:

Line 1: The Request

Line 2: A comment or debug information

Line 3: The response

```
:000302050110030001
This is a comment or debug info
:0110
```

Additional information:

CHAPTER 7

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.
```

CHAPTER 8

Changelog

8.1 Version 0.4.0 (02.08.2014)

- Docs: New handler type for temperature and humidity sensors
- Protocol: Support float types
- Feature: Set max packet size to 256 byte
- Feature: New ArduRPC_Ethernet lib
- Feature: Handlers have to be a class
- Core: Rename ArduRPCSerial to ArduRPC_Serial
- Docs: New handler type for Arduino based boards

8.2 Version 0.3.0 (25.02.2014)

- Improvements to lower memory usage
  - Shared buffers
- Feature: ArduRPCSerial class
- Feature: Macros to get version information
- Feature: Functions to write result data with type identifier

8.3 Version 0.2.0 (no public release)

- Protocol improvements
- Feature: Handler names
- Feature: Possibility to register functions
- Feature: None value
- Feature: Return codes

8.4 Version 0.1.0 (no public release)

- Initial version

CHAPTER 9

Handlers

- Official handler repository
  - Colorduino_GFX
  - Adafruit NeoPixel
  - ...

CHAPTER 10

Client libraries

Python:

- python-ardurpc

CHAPTER 11

Application

B

G

M

P

S

T