# WiPy Tools Documentation

***Release 0.0.9***

**Dwight Hubbard**

June 29, 2016

Contents

CHAPTER 1

# Table of Contents

## 1.1 Installation Instructions

### 1.1.1 Installing on CPython 3

Although micropython-microqueue is designed to function with micropython, it is supported on most python 3 interpreters.Use pip to install on Python3 or PyPy3.

```
$ pip install micropython-microqueue
```

### 1.1.2 Installing on micropython

The installation process differs depending on the version of micropython being used. However the **upip** module is used to do the installation from the Python package repositories.

#### Installing on micropython unix

Use the micropython **upip** module to install on micropython.

```
$ micropython -m upip install micropython-microqueue
```

#### Installing on micropython on the esp8266

To install on micropython embedded platforms:

##### Step 1. Change into the esp8266 build directory

```
$ cd esp8266
```

##### Step 2. Use upip to install the module into the scripts directory.

Set the MICROPYPATH environment variable to point to the scripts directory.

```
$ MICROPYPATH=scripts;micropython -m upip install micropython-microqueue
```

#### Step 3. Deploy the module to the esp8266.

```
$ make deploy
```

### 1.1.3 Example, of installing on ESP8266

Install using the upip module into the esp8266 scripts directory

```
$ MICROPYPATH=scripts;micropython -m upip install micropython-microqueue
Installing to: scripts/
Warning: pypi.python.org SSL certificate is not validated
Installing micropython-microqueue 0.0.6 from https://pypi.python.org/packages/07/55/c8cb5881a86906da5
Created scripts/microqueue/
Installing micropython-redis.list 0.0.57 from https://pypi.python.org/packages/4d/71/b12f84002e4d35ce
Created scripts/uredis_modular/
Installing micropython-redis.client 0.0.57 from https://pypi.python.org/packages/5a/b6/641f3f47f8ef69
Installing micropython-redis-modular 0.0.57 from https://pypi.python.org/packages/0a/68/1424002583bc7
```

Deploy to the esp8266

```
$ make PORT=/dev/ttyUSB0 deploy
Use make V=1 or set BUILD_VERBOSE in your environment to increase build verbosity.
Generating build/frozen.c
Generating build/genhdr/mpversion.h
GEN build/genhdr/qstr.i.last
GEN build/genhdr/qstr.split
GEN build/genhdr/qstrdefs.collected.h
QSTR not updated
CC ../py/modsys.c
CC moduos.c
CC build/frozen.c
CC ../lib/utils/pyexec.c
LINK build/firmware.elf
   text    data     bss     dec     hex filename
 527580    1044   56216  584840   8ec88 build/firmware.elf
Create build/firmware-combined.bin
esptool.py v1.2-dev
('flash    ', 34992)
('padding  ', 1872)
('irom0text', 493672)
('total    ', 530536)
Writing build/firmware-combined.bin to the board
esptool.py v1.2-dev
Connecting...
Running Cesanta flasher stub...
Flash params set to 0x0020
Writing 532480 @ 0x0... 532480 (100 %)
Wrote 532480 bytes at 0x0 in 46.2 seconds (92.2 kbit/s)...
Leaving...
Verifying just-written flash...
Verifying 0x81868 (530536) bytes @ 0x00000000 in flash against build/firmware-combined.bin...
-- verify OK (digest matched)
#@esptool.py --port /dev/ttyUSB0 --baud 115200 write_flash --flash_size=8m 0 build/firmware.elf-0x000
$
```

## 1.2 Usage

The microqueue module creates queues that are stored in redis lists. In order to establish a queue you will need a running redis server.

If the redis server host and port are not specified and the bootconfig module was used to configure the device with a redis_server and redis_port, that redis server willl be used by default.

### 1.2.1 Creating a queue worker on an esp8266

The queue.worker decorator runs the decorated function with the value from the queue passed as the argument to the function.

```
MicroPython v1.8.1-87-g7ddd85f-dirty on 2016-06-27; ESP module with ESP8266
Type "help()" for more information.
>>> from microqueue import MicroQueue
>>> queue = MicroQueue('queuename', host='192.168.1.183', port=6666)
>>>
>>> @queue.worker
... def print_message(message):
...     print(message)
...
>>> print_message()
Micropython Rocks!!!
```

### 1.2.2 Writing to the queue

The resulting queue is compatible with the python redis-hotqueue/hotqueue modules available. As long as the json serializer is used (pickle is not supported on micropython)

```
Python 2.7.11+ (default, Apr 17 2016, 14:00:29)
[GCC 5.3.1 20160413] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import json
>>> from hotqueue import HotQueue
>>> queue = HotQueue('queuename', host='192.168.1.183', port=6666, serializer=json)
>>> queue.put('Micropython Rocks!!!')
>>>
```

## 1.3 Resource usage

The microqueue module tries to keep memory utilization to a minimum.

The module currently uses 2240 bytes to import and 4064 bytes to create a microqueue object on the ESP8266 port.

Which means the following code uses 6304 bytes:

```
from microqueue import MicroQueue
q=MicroQueue('repl')
```

The redis protocol is binary safe which means the returned data doesn't have to be copied or processed before being returned. As a result the queue worker just needs sufficient memory to create the object being returned from the redis queue. Which means the queue worker does not require a significant amount of resouces while in use.

Here is an example function to show the memory free and a queue worker that displays the message returned and the amount of free memory:

```
>>> def free():
...     gc.collect()
...     return gc.mem_free()
...
>>> free()
14224
>>> @q.worker
... def print_free(message):
...     print("Received", message)
...     print(free())
...
>>> print_free()
Received hello
13664
Received hello
13632
Received hello
13632
Received hello
13632
Received hello
13632
```

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search