

WiBotic Inc.

4545 Roosevelt Way NE – Suite 400

Seattle, WA 98105

650-722-0679

<http://www.wibotic.com>

info@wibotic.com

Prepared by: WiBotic Inc

Date Updated: May 2018

Notice: This document is provided for informational purposes only. It represents WiBotic's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessments of the information in this document and any use of WiBotic's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from WiBotic.

© 2018, WiBotic Inc. All rights reserved.

Table of Contents

TABLE OF CONTENTS	2
1 PYTHON HELPER LIBRARY	3
1.1 Introduction.....	3
1.2 Setup.....	3
1.3 Verify Sample Code Operation.....	3
1.4 Python Development	4
1.4.1 Provided Code	4

1 Python Helper Library

1.1 Introduction

A Python helper library and sample code is provided as part of the SDK to help the user integrate faster with the Wibotic system and to demonstrate usage patterns. Both a lower level interface and higher level interface are provided.

1.2 Setup

The libraries and sample code require Python 3.6 (or newer) to be installed and the sample code was developed tested to run on Windows, though may run on Linux or Mac OSX with small modifications. Beyond the basic installation of Python, two libraries will be required: websockets and asyncio.

- Install Python 3.6 (or greater). <https://www.python.org/downloads/>
- [Ensure python is correctly configured in your PATH environment variable.](#)
- Install the websockets package. Note that the *pip* tool referenced is included with Python 3.6. <https://websockets.readthedocs.io/en/stable/intro.html>
- Install the asyncio package. <https://pypi.org/project/asyncio/>
- If not familiar with asyncio or Python coroutines, it is recommended that the user briefly read the following information.
<http://cheat.readthedocs.io/en/latest/python/asyncio.html>

If not already done, the user should install the Wibotic UI as included and documented elsewhere in this development kit and ensure that the Windows laptop is correctly configured to talk to the transmitter via the Ethernet connection. If the Wibotic UI is not able to communicate with the transmitter, the Python scripts will also fail to operate!

1.3 Verify Sample Code Operation

At a command prompt window, navigate to the directory that contains all the wibotic sample code and other python files and execute the following: `python wiboticsample.py`

You should see output similar to the following:

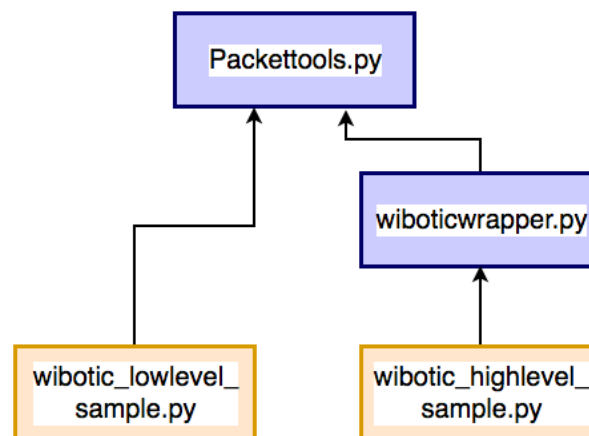
```
Successfully set the ChargeEnable setting to True
Transmitter Build Hash: 66f2ed6d
Receiver Build Hash: 66f2ed6d
Transmitter MAC Address: 34d95400003f
Receiver MAC Address: 34d954000040
=====
Transmitter data 28ms old
Transmitter Power Level: 64000
Receiver data 16ms old
Receiver Battery Voltage: 16.49V
Receiver Battery Charge State: 4
Receiver Battery Charge Current: 0.00A
Receiver Temperature: 24.9C
=====
Transmitter data 55ms old
Transmitter Power Level: 64000
Receiver data 47ms old
```

```
Receiver Battery Voltage: 16.52V
Receiver Battery Charge State: 4
Receiver Battery Charge Current: 0.00A
Receiver Temperature: 24.9C
```

If you do not see this output, it is possible that either your TCP/IP settings are not configured correctly (is the Wibotic UI working?) or that the Transmitter is not configured to be at the default IP address of 192.168.2.20. In the latter case, you can modify `wiboticsample.py` to change the IP address to the address you have configured.

1.4 Python Development

The following diagram shows the dependency relationships between the various python files.



1.4.1 Provided Code

- `packettools.py`: low level library code to help create and parse packets used in communication with the Transmitter.
- `wibotic_lowlevel_sample.py`: Example code showing how to open the transmitter websocket directly, and communicate to the transmitter over that socket, using the low level `packettools` library to parse and create data packets. Because this sample connects directly to the asynchronous websocket interface it is the method of choice when requiring high frequency (10Hz) ADC data packets to be received from the Transmitter and any connected Receiver.
- `wiboticwrapper.py`: high level library code that implements and encapsulates the transmitter websocket interface and provides a more synchronous command/response call flow from client applications. ADC packets can still be received via this library but it requires the consumer of this sample code poll the interface to receive the most recently received ADC packet. This library is ideal for a simpler command/response implementation but less suitable for asynchronously receiving ADC data at a high frequency.
- `wibotic_highlevel_sample.py`: Example code showing how to use the higher level library code referenced above and implements some example commands. It sets the `ChargeEnable` setting to `True`, then determines the Transmitter and Receiver

build versions and MAC addresses. Finally, it enters a loop to show the most recently received Transmitter and Receiver ADC packets every second.

It is recommended that you use the low level packettools python interface if you require access to high frequency (10Hz) data as it is coming off the websocket and are also comfortable with asynchronous websocket programming. If more simple setting control and occasional data gathering are required it may be easier to use the higher level wiboticwrapper interface.