



WIBOTIC

4545 Roosevelt Way NE Suite 400

Seattle, WA 98105

650-265-7987

<http://www.wibotic.com>

info@wibotic.com

Prepared by: WiBotic Inc

Date Updated: January 2019

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.

© 2019, WiBotic Inc. All rights reserved.

Contents

Description	6
Key Features	6
Development Kit Configuration	7
Inside the Box	7
Specifications	8
Power Level	8
Initial Hardware Setup	10
Connecting the Components	11
Configuring Ethernet Connectivity	13
Configuring Direct Ethernet Connectivity	13
Windows 7/8/10	14
Linux (Temporary Connection)	16
Linux (Permanent Connection) or Other OS	17
Ethernet Configuration Reset	17
Using the System – Bench Testing	18
Default Charge Settings	19
WiBotic Control Panel GUI Overview	21
Starting Wireless Power Transmission	21
Battery Charging Process	22
Power-Down Events	23
GUI Status Indicators	24
GUI Alert Messages and Troubleshooting	25
Troubleshooting GUI Connectivity	26
Updating Firmware	26
Process Description	26
Uploading the Firmware	27
Onboard Charger	29
Transmitter	30
APPENDIX A: NETWORK API	33

Getting Started	33
General Packet Format.....	34
Request Packet Types	34
Response Packet Types	34
API Requests.....	34
Building API Requests	34
Parsing API Responses	35
Parameters	35
Parameter Status Codes	37
Real-time ADC Packets	37
Device ID	38
APPENDIX B: PYTHON HELPER LIBRARY	39
Introduction.....	39
Setup	39
Verify Sample Code Operation	39
Python Development	40
Provided Code	41
APPENDIX C: ONBOARD API.....	43
CAN Interface	43
USB to CAN.....	44
CAN API	44
Basic Setup.....	45
Talking over CAN	46
Serializing Packets.....	49
WiBotic DSDL Definitions.....	50
Parameters	52
Parameter Status Codes	53

Description

WiBotic wireless power solutions enable autonomous wireless charging of aerial drones, mobile robots, unmanned aquatic vehicles, and other industrial automation devices across a wide range of applications. For simplicity and consistency, we refer to all of these vehicles, drones, and devices as “robots” for the remainder of this User Guide.

The WiBotic Development Kit is the first step in automating power delivery to robots or entire robot fleets. In addition to fully automating the battery charging function, remote monitoring of battery parameters and configurable charging profiles can extend the life of every battery. This configurability helps to ensure each and every robot is ready to take on its next mission.

The kit is assembled from standard WiBotic components, selected to meet customer needs for size, weight, and power as well as to maximize charging range flexibility. The WiBotic Web-Based Graphical User Interface (GUI) and Application Programming Interface (API) are also supplied for system configuration and monitoring purposes.

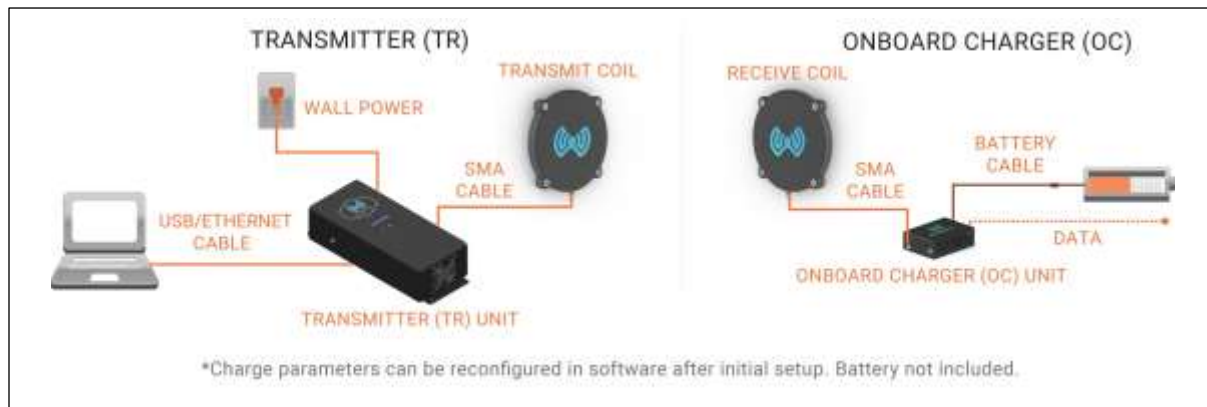
This manual describes setup and operation of the Development Kit and all additional production systems.

Key Features

- ✓ **100% autonomous wireless battery charging**
 - Stand-by “heartbeat mode” with auto sensing for charge initiation
 - Auto tuning for wide transmit-receive coil range and alignment flexibility
 - Multiple configurations from 90W to 300W to provide the best combination of size, weight and power for each application
- ✓ **Flexible charging modes**
 - Supports most common robot battery chemistries including LiPo, LiFePO₄, SLA, Ni-MH
 - Constant current/constant voltage charging modes for lithium chemistries and a programmable mode for non-standard batteries
 - Fully programmable charging current and voltage via the GUI or APIs
- ✓ **Optional wired battery charging**
 - Multiple options from 5A/90W to 30A/750 watts delivered to the battery when directly wired for non-wireless operation
- ✓ **Lightweight Onboard Charger system for weight-sensitive installations**

Development Kit Configuration

The following diagram shows the operational configuration of the system as well as the individual components supplied with every WiBotic Development Kit:



Inside the Box

- | | |
|----------------------------|---------------------------|
| 1. Ethernet Cable | 6. Receive Coil |
| 2. USB to Ethernet Adapter | 7. Transmit Coil |
| 3. Power Cable | 8. Coaxial SMA Cable (x2) |
| 4. Power Transmitter | 9. Battery/Load Connector |
| 5. Onboard Charger | |



Specifications

All WiBotic Wireless Power systems share the following standard specifications:

PARAMETER	RANGE
Wireless Power Operating Frequency	6.78 ± 0.015 MHz
Communication Link Frequency	2.433 - 2.483 GHz
Ambient Operating Temperature	-20° to 40° C
Transmitter AC Input Voltage	90 - 264 VAC
Transmitter AC Input Frequency	50 – 60 Hz
Transmitter Data Port	Ethernet (RJ45)
Transmitter Data Communications	WiBotic API via WebSockets
Onboard Charger Auxiliary DC Input Voltage*	36 - 55 VDC
Onboard Charger Data Communications	WiBotic UAVCAN API over CAN-bus

*The standard WiBotic Development Kit includes an AC/DC power supply built into the transmitter. Direct DC input versions can also be accommodated. Contact WiBotic for details.

Power Level

Development Kits are available in configurations that combine Transmitters and Onboard Chargers to provide the best combination of size, weight and power for each application. The following table shows the available combinations for each desired power level. All systems utilize the same 200mm Transmitter and 100mm Receiver antennas unless custom antenna sizes/shapes are ordered.

Up to:	90W	125W	250W	300W
Transmitter	TR-110	TR-110	TR-300	TR-300
Onboard Charger	OC-110	OC-210	OC-250	OC-300

Onboard Chargers (OCs) are paired with the correct Transmitter to achieve their maximum power level, so the OC ultimately determines the amount of power delivered to the battery. Each OC has a different maximum power level (watts), maximum current level (amps), and output voltage range as shown in chart below:

ONBOARD CHARGERS	OUTPUT VOLTAGE (TO BATTERY)*	MAX OUTPUT POWER	OUTPUT CURRENT
OC-110	7.92 - 30.1 VDC	90W	0.5 - 5A
OC-210	12.03 – 36.0 VDC	125W	0.5 - 10A
OC-250**	12.03 – 36.0 VDC	250W	0.5 - 10A
OC-300	0 - 59.6 VDC	300W	0.5 - 30A

*DC Output voltage and current is configurable within the specified range via WiBotic software or API.

**The OC-250 can be configured for higher output voltage if required.

The total amount of power delivered to the battery is determined by both the maximum power output and the maximum current output, based upon the voltage of the battery being charged. Since there is a straightforward equation that relates voltage, amperage and wattage (*volts x amps = watts*) we can use simple calculations to determine which factor will ultimately determine your system's power level and charge speed:

Example:

The user wishes to configure the OC-110 to charge a battery that reaches 12.6V when fully charged. In this case, the OC-110 will reach its 5A limit before the 90W limit is reached, and power delivery will not exceed 63W.

$$12.6V \times 5A = 63W$$

A different user wishes to configure the same OC-110 to charge a battery that reaches 25.2V when fully charged. In this case, the OC-110 will reach its maximum 90W limit before it can reach the 5A limit. Power delivery will not exceed 90W and amperage will be a maximum of 3.57A.

$$90W / 25.2V = 3.57A$$

Note that the above values are representative only, and other factors such as antenna position can also affect system efficiency and power delivery.

Finally, there is no risk in inadvertently setting the desired charge current above the maximum level. In that case, the system firmware will limit operation to the maximum power level that is safely achievable.

After fully reading this User Guide, please contact WiBotic if there are any questions about the maximum power level available for your specific system.

Initial Hardware Setup

For initial system familiarization, WiBotic highly recommends bench-top testing. This allows the system to be operated in open air and at various power levels and antenna positions – demonstrating the unique flexibility of WiBotic's technology.



Note: WiBotic does not recommend immediate installation of the wireless power system on robots for testing purposes. Many robots consist of metallic frames or body parts that can temporarily de-tune wireless power antennas, resulting in degraded performance. To avoid de-tuning problems and to maximize system performance, WiBotic offers integration services for customers who are ready to test the system on their robot. Contact WiBotic for details.



For bench-top testing WiBotic provides an antenna stand to new customers. Shown at left, the stand includes mounts for both the transmit and receive antenna coils as well as a “T” shaped base. Sliding the antennas on the base allows power transfer to be tested at different distances (both face-to-face and side-to-side) and angularities.

To install the transmit antenna coil on its mount, turn the coil so the antenna connector is facing sideways. Then remove the bottom two enclosure screws and the captive nuts from the back side of the antenna enclosure. Place the nuts into the pockets on the back of the antenna mount.

Insert longer M5 screws (shown at right and supplied in the Dev Kit box) through the antenna enclosure, through the front side of the mount, and into the nut on the back side of the mount. Hand tighten both screws to secure the antenna coil to the mount.



When finished, the vented back side of the transmit coil enclosure should be against the plastic mount and the thumb screw should be on the back side of the mount as shown.

Repeat the process for the receive antenna using the longer M3 screws. Note that the receive antenna enclosure does not have a vented back side, but the enclosure screw heads should be on the opposite side from the mount and thumb screw.

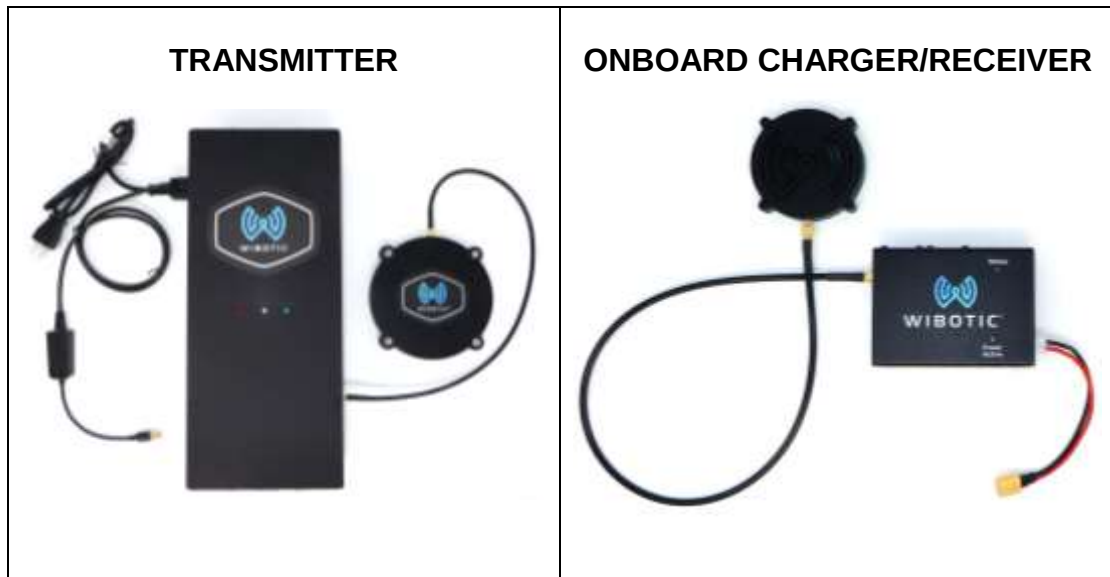
Now loosen the thumb screws and insert the head of each thumb screw bolt into the slots on the T base. It does not matter which antenna is inserted into each slot since both options allow antennas to move freely relative to one another. Just be sure the heads of the enclosure screws on both coil enclosures are facing each other.

Once the antenna stand is set up, proceed with the rest of the installation as follows.

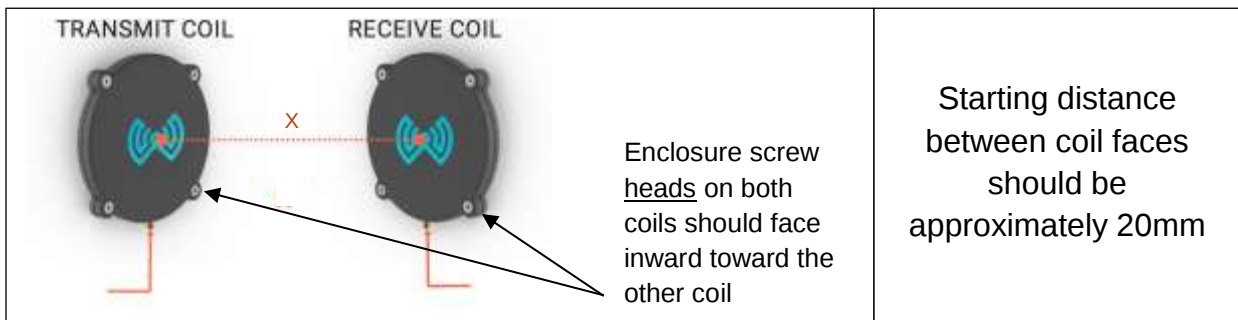


Connecting the Components

1. Connect one coaxial SMA cable between the Power Transmitter and the Transmit Coil.
 - a. For TR-110 systems, make sure the connection is tightened to 8 inch-lbs on both ends.
 - b. For TR-300 based systems, hand tighten the larger type “N” connector on the transmitter itself, and tighten the SMA connector on the transmit antenna end to 8 inch-lbs.
2. Connect the second coaxial SMA cable between the Onboard Charger and the Receive Coil. Make sure the connection is tightened to 8 inch-lbs on both ends.
3. Connect the Onboard Charger to the target device to be powered (e.g. the battery) using the supplied battery connector cable. **NOTE:** *The development kit requires a direct connection to the battery for operation. The “heartbeat mode” algorithm is disabled if the Onboard Charger does not receive power from the battery before wireless power is enabled.*



- With the antenna coils mounted on the coil stand, loosen the thumb screws and adjust the coil enclosures so they are concentric and approximately 2cm apart. This will be the starting position for testing, but you may then move the coils in and out and side to side during testing. You may also twist one or both coils to experience power transfer when the coils are not parallel. **NOTE:** *It may be possible for the coils to be placed too close together. Please reference your system's Certificate of Conformance (CoC) for optimal coil spacing (x). The CoC provides the distance between plastic enclosure faces only. If you intend to remove the antenna coils from their enclosures, please contact WiBotic for the correct spacing.*





CAUTION: Do *not* place the Transmit Coil on a metallic surface, and make sure that all metal objects are at least 30cm away from the Transmit and Receive Coils during testing.



CAUTION: Removing the antenna coils from their enclosures is not recommended without guidance from WiBotic. High voltage exists on antenna coil components during charging. **DO NOT** touch any portion of a bare antenna coil during charging.

5. Connect the Power Cable to the Power Transmitter (before plugging in the Power Cable to an electrical outlet or other power source).
6. Plug the other end into a standard electrical outlet. The Green “READY” LED on the Power Transmitter will only turn on after the system has been activated via the software GUI (see later steps) but the LCD screen will become active.
7. Connect the Ethernet cable from the Power Transmitter to a computer. Either plug directly into the Ethernet port on the computer (if available) or use the provided Ethernet-to-USB Adapter. There are additional steps required to configure the IPv4 settings on Windows. Refer to the following section on Configuring Ethernet Connectivity to properly configure the Ethernet connection before using the system.

Configuring Ethernet Connectivity

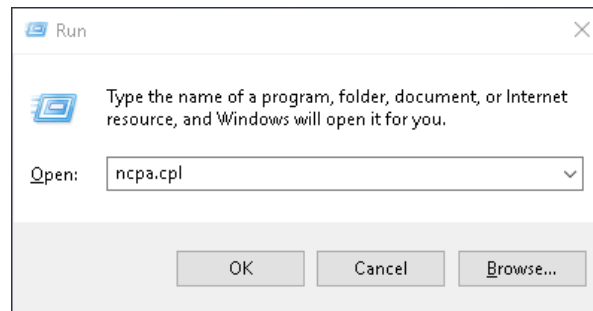
The WiBotic Web GUI allows users to control the WiBotic system, view its status, and make changes to settings without having to install separate software. The interface is accessible via modern versions of Firefox, Chrome, Edge, and Safari. A direct point-to-point Ethernet connection is recommended for initial setup. The Power Transmitter serves the WiBotic Web Graphical User Interface (“Web GUI”) on a specific IP address configured on HTTP port 80.

Configuring Direct Ethernet Connectivity

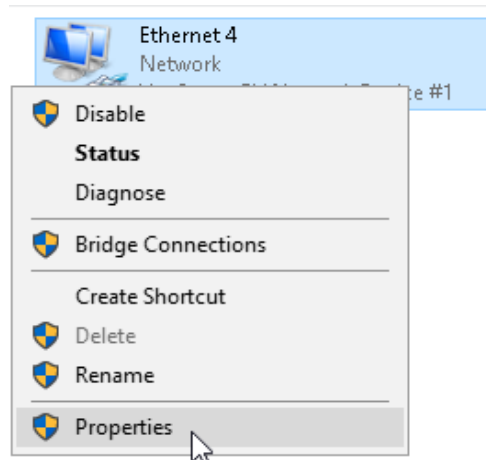
The following instructions create a direct Ethernet connection between your computer and the WiBotic Transmitter. Please note that WiBotic currently supports only direct Ethernet connections and users should not attempt to connect to the Power Transmitter over an Ethernet network/LAN. While a configuration screen for “Network Settings” is available via the Settings menu in the Web GUI, we do not support changes to any of the network parameters in that menu at this time. Full support for LAN integration will be available in a future system upgrade.

Windows 7/8/10

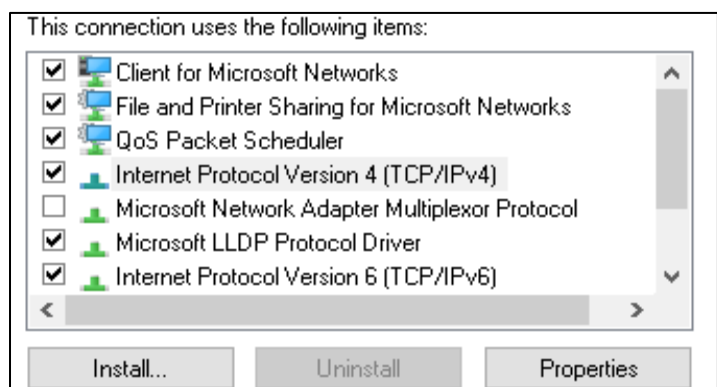
1. With the Power Transmitter powered on, confirm that the Ethernet Cable is connected to the computer either directly, or via the provided USB Adapter.
2. Open “Network Connections” by pressing **Windows Key + R**. Type in “ncpa.cpl”, and click OK:



3. Right click on the network adapter that is connected to the WiBotic Power Transmitter and select **Properties**:

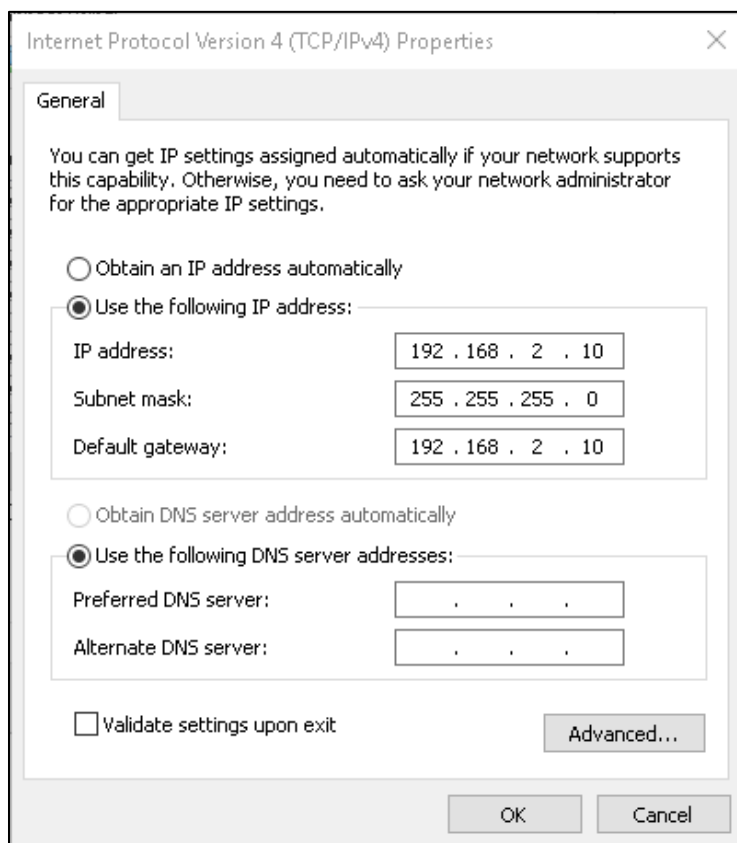


4. Select **Internet Protocol Version 4 (TCP/IPv4)** and click the **Properties** button below it:



- Click the **Use the following IP** address button and specify the following settings. The DNS section may be left blank. Click OK:

IP address:	192.168.2.10
Subnet mask:	255.255.255.0
Default gateway:	192.168.2.10



- After the network settings have been configured, **open a Web Browser** (Firefox, Chrome, or Edge) and go to the following URL:

<http://192.168.2.20>

7. Verify that the web application loaded and that the Transmitter status at the top of the screen shows “Idle”.



Linux (Temporary Connection)

1. Open a terminal window (many desktop environments have a keyboard shortcut CTRL+ALT+T to do this)
2. Type “ip address” to list all network interfaces on your computer and their respective IP addresses.
3. Locate the name of the network interface connected to the WiBotic system. This interface is usually near the bottom if using a USB dongle. It should be something similar to *enx0012345678*, *eno1*, or *eth0*.
4. Type “sudo ip addr add 192.168.2.10/24 dev *enx0012345678*” assuming you’re running a system that uses sudo to elevate permissions. Replace *enx0012345678* with the name of the network interface identified above.
5. Visit <http://192.168.2.20> in a web browser (Firefox, Chrome)
6. Verify that the web application (shown above) loaded and that the Transmitter status at the top of the screen shows “Idle”.

Linux (Permanent Connection) or Other OS

Please refer to the network setup guide for your operating system. You will need the following information:

Static IP (for the computer you are setting up):	192.168.2.10
Static IP (for the WiBotic transmitter):	192.168.2.20
Subnet Mask:	255.255.255.224 (/27) or larger

Additionally, if your OS has an option similar to “Use network only for resources on its own network”, this option should be checked for the network connection used for a direct Ethernet connection to the WiBotic hardware.



Note: All Ethernet connectivity instructions are intended for an ad-hoc point-to-point connection directly to the Development Kit. Changing the IP address, default gateway, and subnet mask of your default adapter may cause a loss of connectivity between your computer and your local network.



Note: Only connect the Power Transmitter to a local machine’s internal Ethernet port or by using the USB-to-Ethernet adapter provided by WiBotic. The Power Transmitter should not be plugged into any network switchable interface.

Ethernet Configuration Reset

If the WiBotic system does not seem to be responding on the default IP address, you can attempt to reset the Ethernet configuration to the default factory settings.

1. Locate a thin rigid object such as a small hex key or an unfolded paperclip.
2. Gently insert the object into the hole located on the Transmitter just below the Ethernet jack.



3. Upon inserting the thin rigid object, you should feel a button that depresses about 1mm. Hold the object on that button for more than 5 seconds.
4. Upon releasing the button, the LCD screen (if equipped) on the front of the transmitter will turn magenta and display “Ethernet Config Reset”.



Note: If the LCD only turns white and displays network configuration settings after being released, the button did not detect a continuous 5 second press and the Ethernet settings have not been reset. Try again.

5. Power cycle the transmitter in order for the new network settings to take effect.

Using the System – Bench Testing

Once the network connection has been established, all system cables have been connected, and the coils are within the ideal operating range, you’re ready to experience wireless power!



Important Note: Once again, testing of the system on a benchtop is highly recommended as a first step. This will ensure the system is operating correctly and you'll experience the range and power level it can provide. You may then mount the system on your robot for further testing. If you experience different performance after robot installation, however, it is very likely the system has been temporarily de-tuned by the robot's chassis or some other conductive object. WiBotic offers integration services to help re-tune the system for your specific robot, so please contact us if this occurs!

At this time open the Web-GUI using the following IP address typed directly into your web browser: <http://192.168.2.20>

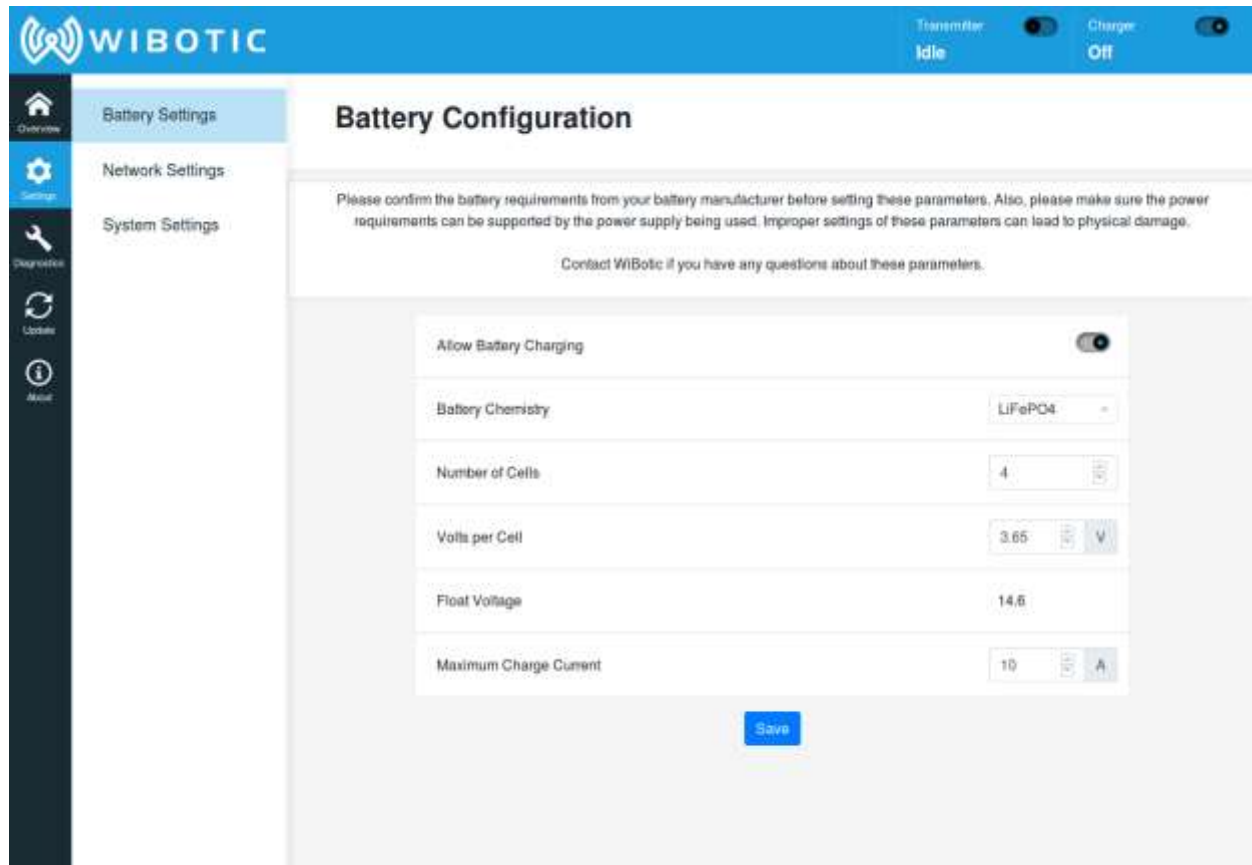
As the default operating mode, wireless power will be disabled (off) when the Transmitter is initially powered on. Therefore, the GUI must be opened and the system must be manually turned on to begin charging. At a later time, the default operating mode can be changed to have the system automatically turn on whenever a robot is in range. See the "System Settings" sub-menu in the main "Settings" menu for this function.

Before starting, please review the rest of this document to familiarize yourself with system operation and the Web GUI application.

Default Charge Settings

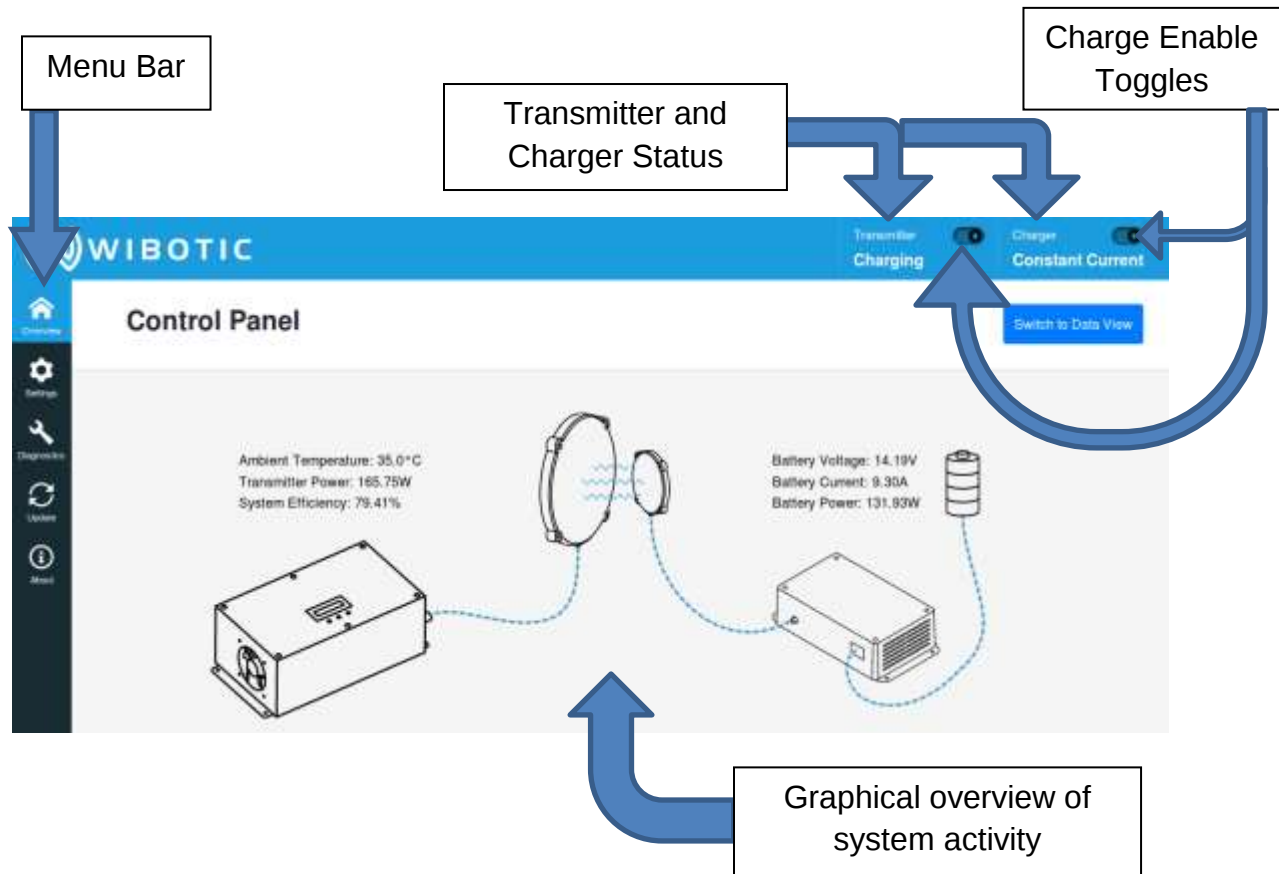
Prior to shipping the system, WiBotic will pre-configure the battery chemistry, number of battery cells, per-cell charging voltage (which begins as a default based upon battery chemistry) and maximum charge current based upon the information you provide. Please begin by opening the "Battery Settings" sub-menu within the main "Settings" menu to confirm these values.

If you have switched batteries or the WiBotic system is installed on a different robot, these values can be adjusted. Simply use the Battery Configuration screen, shown below, to adjust the settings for your battery. Note that a "custom" option is also provided in case the default values provided by WiBotic do not match your particular battery.



WARNING: Please confirm the battery requirements to ensure the charge current and voltage are within permissible limits for your battery. Damage to the battery, charger or connected equipment may occur if incorrectly configured. A damaged battery could cause fire or personal injury.

WiBotic Control Panel GUI Overview



Starting Wireless Power Transmission

To begin testing, confirm that the System Status “Charger” toggle switch in the upper right corner is in the default “On” position and the “Transmitter” toggle switch is in the default “Off” position.

Next, turn the Transmitter on by clicking the Transmitter toggle switch. If there is an Onboard Charger ready to accept power, and it is within range of the transmit coil, the system will now enter its charging sequence. In a few seconds, charging will begin and the graphical indicators in the center of the screen will show active charging.



Once wireless power transmission begins, the system will stop only if the Transmitter toggle is clicked again to turn it off, or if a shutdown event occurs. (See the “Power Down Events” section for details). WiBotic recommends disconnecting power to the development kit when not in use for extended periods of time.

Battery Charging Process

WiBotic Onboard Chargers are configured by default to act as Constant Current/Constant Voltage (CC/CV) battery chargers.

If the Power Transmitter is turned on and an Onboard Charger is in range, the system will begin charging if the battery is not already considered fully charged. The charger regulates output voltage to the battery at the pre-defined voltage limit in the “Battery Settings” menu. The initial voltage limit is set at the WiBotic factory based upon the customer’s reported battery specs and is listed on the Certificate of Conformance. The voltage limit can be changed and will be stored in memory even if the Onboard Charger is powered off and/or disconnected from the battery.

During startup, the charging current will ramp up to the maximum constant-current charge rate as configured in the WiBotic software. The Onboard Charger will always try to ramp-up to the maximum charge current. However, in some cases, depending on either sub-optimal coil position, environmental conditions or high temperatures of the electronics, the Onboard Charger may not be able to achieve the maximum charge current. If unable to reach the maximum charge current, the Onboard Charger will not increase beyond a charge current that the system can reliably and safely sustain.

After the constant current portion of the charge cycle is complete, the Onboard Charger will enter constant-voltage charging mode. During constant-voltage charging mode, the charge current decreases gradually as the voltage of the battery approaches the charge termination voltage.

There are two ways for the system to terminate charging in constant-voltage mode:

- 1. CV Mode Timeout:** Once charging has started, a 3-hour timeout period begins. Even if the battery is not fully charged after this period, the charge cycle will terminate and the Power Transmitter will momentarily return to standby. If the system senses battery voltage is below the fully charged level, the system will resume charging for another 3 hours.
- 2. CV Mode C/10 “Charge Complete”:** If the charge current drops below one-tenth of the maximum charge current rate (C/10), then charging is considered complete and the Power Transmitter will return to standby.

The GUI will indicate that the battery is fully charged by displaying “Battery Full” as the charger’s status.



Once the battery has been fully charged, and to avoid repeatedly topping-off the battery, there is a voltage threshold within WiBotic software that must be met before charging will automatically resume.

The threshold value depends on the battery chemistry and float voltage and is set to maintain the battery state-of-charge at greater than 95% capacity. If the Power Transmitter is left on, and the battery discharges below the threshold, the system will automatically resume charging to keep the battery full.

As an example, if a robot finishes a task and returns to the Power Transmitter for charging, the battery will automatically begin to charge. Once the battery is fully charged, and the transmitter stops charging there may still be a load on the battery from other robot systems (the CPU, sensors, etc.). This load will slowly drain the battery until it drops below the threshold value. At that point, the WiBotic system will resume charging to bring the battery back to the “Battery Full” state.

Power-Down Events

There are several mechanisms that can cause the Power Transmitter to power-down during charging:

COMMUNICATION ERROR / RX TIMEOUT: The Power Transmitter and Onboard Charger communicate over a low-power 2.4GHz radio link. If the Onboard Charger is out of range, obstructed by metal, or is in a particularly electrically noisy environment, the GUI may display “RX Disconnected”. The Power Transmitter will power-down until the radio link is restored.

GUI INFORMATIONAL MESSAGES: There are informational messages that appear in the log of the GUI. These messages indicate either a status update or an alert – some of which may cause a precautionary shut down. If an alert has been triggered and power transfer stops, the message will provide a recommendation to ideally solve the problem. If the GUI application itself stops responding, it may be necessary to try refreshing the web page. If these problems persist, contact WiBotic for support. See the section on GUI Alert Messages for more details.

Transmitter/Charger Status Indicators

The table below enumerates the transmitter and/or charger status indicator name and descriptions.

Table 1: GUI Status Indicators

STATUS NAME	STATUS DESCRIPTION
Off	The Device is off.
Idle	The Power Transmitter will periodically check for Onboard Chargers when they are in range of the 2.4GHz low power radio.
Stabilizing / Ramp Up	Wireless power is enabled and power is ramping up to target levels.
Charging	The battery is charging.
Constant Current	The battery is being charged in constant-current mode.
Constant Voltage	The battery is being charged in constant-voltage mode.
Battery Full / Done	The battery has completed charging.
Alarm	An unexpected event occurred. See log message for details. The system should restart once conditions return to normal.

Alert Messages and Troubleshooting

The following events will cause the Transmitter to shut down temporarily, and then automatically re-enter heartbeat mode.

Table 2: GUI Alert Messages

ALERT NAME	ALERT DESCRIPTION
TX Alarm: Power Supply Over Current	If the power supply exceeds the maximum permitted current, this alarm will occur. The system will restart and enter heartbeat mode. If the problem persists, disconnect the transmitter power cable and contact WiBotic before continuing.
TX Alarm: Power Supply Over Power	If the power supply exceeds the maximum power limit, this alarm will occur. The system will restart and enter heartbeat mode. If the problem persists, disconnect the transmitter power cable and contact WiBotic before continuing.
TX Info: Radio Communication Lost	If the radio link is not maintained between the transmitter and Onboard Charger, the transmitter will turn off. Try moving the transmitter and/or Onboard Charger closer to each other. The system will shut down and enter heartbeat mode.
TX Info: Received power low	This usually occurs when the coils are positioned too far apart or in a non-optimal orientation. Transmitter will immediately shut down. Try repositioning the coils if this alert persists.
TX Info: Coil position not optimal	The transmitter cannot transition out of heartbeat mode, and will retry in 10 seconds. Make sure the coils are placed within the Ideal Operating Range. If the message persists, the transmitter will remain in heartbeat mode.
TX Info: Internal communication error	An internal error might occur that will cause the system to restart. The system will restart and enter heartbeat mode. Contact WiBotic if this problem persists.
TX Alarm: RX is in alarm state	The Onboard Charger has encountered a problem. The system will attempt to restart and resume operation.
TX Alarm: Power transfer too inefficient. Try moving coils apart.	Wireless power transfer is presently too inefficient. Try moving coils farther apart to resolve issue.
RX Alarm: Battery voltage incorrect	The battery parameters (number of cells, chemistry, volts per cell) are inconsistent with the current battery voltage detected by the charger. Correct the battery parameters or change batteries and try again.
RX Alarm: Rectified voltage crashed	The input voltage seen by the Onboard Charger has changed abruptly. This may be due to poor coil position or other hardware error. If this problem is not resolved by repositioning coils, contact WiBotic.
RX Alarm: Charger shorted	The battery charger on the Onboard Charger has detected a short circuit on the battery output. Try disconnecting the battery to power cycle the Onboard Charger and see if the problem persists. If this problem is not resolved, contact WiBotic as there may be a hardware issue with the Onboard Charger.
RX Alarm: Internal event alarm	The Onboard Charger has had an internal error and has attempted recovery. Try disconnecting the battery to power cycle the Onboard Charger and see if the problem persists. If this problem is not resolved, contact WiBotic.
RX Alarm: Low battery current	Something is limiting the amount of current that can be used to charge the battery. This caused charging to stop. Allow the system to resume, but if the problem persists across multiple charging attempts, contact WiBotic.

Troubleshooting GUI Connectivity

If the Power Transmitter is powered on and connected to the computer via Ethernet, but the web app is inaccessible, proceed through the following steps:

1. Make sure you have setup the IPv4 settings as outlined in the Configuring Ethernet Connectivity Section as in previous section.
2. Try resetting the Ethernet Connection settings as outlined in the Ethernet Configuration Reset section as in previous section.
3. If you still cannot connect to the GUI, contact WiBotic at info@wibotic.com for assistance. It is helpful if you include the results of pinging the device with the command “ping 192.168.2.20” in a terminal or command prompt window. If able, running a packet capture tool such as Wireshark on the interface where the WiBotic system is plugged in, and sending the capture to WiBotic, may also help us assist you in getting the system up and running.

Updating Firmware

Process Description

WiBotic will occasionally release new firmware for the Transmitter or Onboard Charger to improve system performance. A new version of the Web GUI interface may also be made available with enhanced features. In most cases these three updated files will be released together. WiBotic will send a message to customers making them aware of the new firmware’s availability, including download instructions.

When updating firmware, the first step is to download the three files to the computer connected to the WiBotic transmitter via the Ethernet cable. The filenames will be similar to the following:

- **WEB.RFS** – the Web GUI update that is loaded onto the Transmitter
- **rx-m4-ota.bin** – the “rx” indicates this is the firmware file for the Onboard Charger/Receiver
- **tx-m7-ota.bin** – the “tx” indicates this is the firmware file for the Transmitter

Once the files reside on the computer, they can be uploaded to the Transmitter. From there, a process is run via the Web GUI to upgrade the GUI itself, and then to install the new firmware on the Transmitter and Onboard Charger. The recommended order of installation is 1) WEB.RFS, 2) Onboard Charger Firmware, and 3) Transmitter Firmware.

Uploading the Firmware to the Transmitter

Ensure that the WiBotic Transmitter is connected to your computer via an Ethernet connection before proceeding.

Next, type `http://192.168.2.20/update` (substituting 192.168.2.20 for any other address you may have configured your system to use) into the connected web browser.



Note: It's also possible to get to this page from the WiBotic Control Panel web app by clicking the “Update” section on the menu on the left-hand side of the screen, then selecting the first “Upload” button.

The following page should appear:

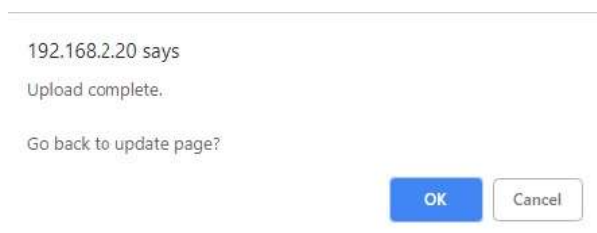


On this page, upload **WEB.RFS**, **rx-m4-ota.bin** and **tx-m7-ota.bin** by first clicking on “Browse...” and then selecting the first file. Click “Upload” and wait for the Upload Complete message to appear.

The message asks if you'd like to go to the main update page in the Web GUI, but since you have two more files to upload, click "Cancel" for now. This will allow you to stay on the upload page and you'll see a green check box indicating successful upload of the first file.



Repeat the steps for the other two files until all three have been successfully upload.



After the last file is uploaded, when prompted to return to the update page, select "OK". At this point you'll be redirected to the main update page in the Web GUI.

Once the page redirects to the Web GUI update page (shown below), check to make sure that the uploaded firmware versions match with the version of the Web Interface.



Onboard Charger

First, ensure that the onboard charger hardware that you want to update is connected to the transmitter. This can be verified by the presence of a firmware version under “Current Firmware Version” for the onboard charger.

If the Current Firmware Version under the Charger section of the update page is different than the Uploaded Firmware Version, then proceed with the update by pressing the “Update” button to the right of the uploaded firmware version section.

The update will proceed. After the update is complete, the Current Firmware Version shown in the page under the Charger section should match the firmware version under Uploaded Firmware Version.

If the system takes more than 5 minutes to update, try restarting both the onboard charger and the transmitter and try again.

Starting Update...

This make take several minutes. Please do not power off or disconnect your device

Updating Charger...

This make take several minutes. Please do not power off or disconnect your device

Finishing up...

This make take several minutes. Please do not power off or disconnect your device

Update Complete!

This make take several minutes. Please do not power off or disconnect your device

Transmitter

If the Current Firmware Version under the Transmitter section of the update page is different than the Uploaded Firmware Version, then proceed with the update by pressing the “Update” button to the right of the uploaded firmware version hash.

The update will proceed. After the update is complete, the Current Firmware Version shown in the page under the Transmitter section should match the firmware version under Uploaded Firmware Version.

If the system takes more than 5 minutes to update, first try refreshing the web page to see if the firmware may have updated, but just not refreshed the page. If it appears that the transmitter still hasn’t been updated, try power cycling the transmitter and try again.

Starting Update...

This make take several minutes. Please do not power off or disconnect your device

Updating Transmitter...

This make take several minutes. Please do not power off or disconnect your device

Update Complete!

This make take several minutes. Please do not power off or disconnect your device

At this point your system’s firmware has been updated and you can begin to enjoy the new features and/or performance afforded by the update.

APPENDICES:

A: Network API

B: Python Helper Library

C: Onboard API

APPENDIX A: NETWORK API

The WiBotic Network API allows users to programmatically access the WiBotic Transmitter over an Ethernet network for system monitoring and control. All functionality available via the WiBotic Web GUI is also programmable via this interface. The following pages provide an overview of the API functionality. WiBotic also provides a Python library and sample code to help users get started. If you have not received the sample files from WiBotic, please contact us at sales@wibotic.com for access.

Getting Started

The WiBotic Network API is accessible over a WebSocket at the address `ws://192.168.2.20/ws` (where 192.168.2.20 is the IP address of the WiBotic Transmitter) using the “*wibotic*” WebSocket subprotocol. You can use any language or environment that can communicate over WebSockets to send and receive API data to the WiBotic system.

A helper library and demo application are available in Python to assist in development (see Appendix B).

NOTE: The WebSocket server does not incorporate any authentication at this time. Do not connect the WiBotic system to a public network without considering the security implications.

NOTE: The WiBotic WebSocket server could be susceptible to an attack where a malicious webpage opens a socket and sends data to the WiBotic system in another isolated network segment. This will be addressed in release versions of the system.

General Packet Format

The WiBotic Network API uses binary WebSocket frames. The frames begin with a byte that indicates the frame type (see Response Packet Types and Request Packet Types below). This byte determines how the rest of the frame should be interpreted.

Request Packet Types

Name	Code	Description
Read Parameter	0x01	Request a parameter to be read
Write Parameter	0x03	Request a parameter to be written

Response Packet Types

Name	Code	Description
Parameter Update	0x80	A parameter was updated
Parameter Response	0x81	Response to a request for data from a parameter
ADC Update	0x82	New real-time ADC packet

API Requests

The following diagrams illustrate the position of the bytes that should make up the binary WebSocket frames that contain the API requests and responses. Multiple successive bytes with the same name indicate that the value is to be split over those bytes. The bytes illustrated continue on to the next line if there are too many bytes to show on one line.

Building API Requests

Read Parameter

0x01	Device ID	Parameter ID	Parameter ID	Parameter ID	Parameter ID
------	-----------	--------------	--------------	--------------	--------------

Write Parameter

0x03	Device ID	Parameter ID	Parameter ID	Parameter ID	Parameter ID
New Data	New Data	New Data	New Data		

Parsing API Responses

Parameter Update

0x80	Device ID	Parameter ID	Parameter ID	Parameter ID	Parameter ID
Status					

Parameter Response

0x81	Device ID	Parameter ID	Parameter ID	Parameter ID	Parameter ID
Param Data	Param Data	Param Data	Param Data		

ADC Update

0x82	Device ID	ADC ID 1	ADC ID 1	ADC Data 1	ADC Data 1
ADC ID 2	ADC ID 2	ADC Data 2	ADC Data 2	...	ADC ID <i>n</i>
ADC ID <i>n</i>	ADC Data <i>n</i>	ADC Data <i>n</i>			

The number of ADC values in a packet can be determined by the following equation:

$$\frac{(Number\ of\ Bytes) - 2}{4}$$

Parameters

Name	ID	Description	Read	Write	TR/OC Availability
Address	3	Address of the device on the internal point to point wireless link	Yes	No	Both
RadioChannel	4	Current device radio channel	Yes	No	Both
DigitalBoardVersion	26	Version of the digital board that is running the system	Yes	No	Both
BatteryCurrentMax	34	Maximum current that should be delivered to the battery in milliamps	Yes	Yes	Both
ChargerCurrentLimit	35	Current limit that the system has decided can be delivered to the battery	Yes	No	OC

MobileRxVoltageLimit	36	Calculated maximum battery voltage based on chemistry, number of cells, and voltage per cell	Yes	No	Both
RxBatteryVoltageMin	37	Calculated minimum battery voltage based on chemistry, number of cells, and voltage per cell	Yes	No	OC
BuildHash	38	Version hash of the firmware that is loaded on the device	Yes	No	Both
TargetFirmwareId	39	Type of firmware image that is running on the device	Yes	No	Both
RxBatteryVoltage	42	Last read ADC value of the battery's voltage	Yes	No	OC
RxBatteryCurrent	43	Last read ADC value of the battery's current	Yes	No	OC
RxTemperature	44	Last read ADC value of the power board's temperature	Yes	No	OC
EthIPAddr	45	Network Static IP Address if no DHCP	Yes	Yes	TR
EthNetMask	46	Network Subnet Mask if no DHCP	Yes	Yes	TR
EthGateway	47	Network Gateway if no DHCP	Yes	Yes	TR
EthDNS	48	Network DNS Server	Yes	Yes	TR
EthUseDHCP	49	Device should use DHCP on the network	Yes	Yes	TR
EthUseLLA	50	Device should fallback to Link-Local Addressing if DHCP fails	Yes	Yes	TR
DevMACOUI	51	MAC Address Organizationally Unique Identifier	Yes	No	Both
DevMACSpecific	52	MAC Address Specific Identifier	Yes	No	Both
EthMTU	54	Ethernet MTU	Yes	Yes	TR
EthICMPReply	55	Reply to Pings	Yes	Yes	TR
EthTCP TTL	56	TCP Time to Live	Yes	Yes	TR
EthUDP TTL	57	UDP Time to Live	Yes	Yes	TR
EthUseDNS	58	Use DNS	Yes	Yes	TR
EthTCPKeepAlive	59	Keep TCP Connections Alive	Yes	Yes	TR

ChargeEnable	60	Enable or disable the ability for this device to transfer power or charge a battery	Yes	Yes	Both
I2cAddress	61	Address of this device on a I2C bus	Yes	Yes	OC
RxBatteryNumCells	62	Number of cells in the battery this charger is configured to charge	Yes	Yes	OC
RxBatterymVPerCell	63	Voltage of a battery cell (in millivolts) that this charger is configured to charge	Yes	Yes	OC
LogEnable	67	Enable logging battery charge data	Yes	Yes	TR
RxBatteryChemistry	68	Chemistry of the attached battery	Yes	Yes	OC
IgnoreBatteryCondition	70	Ignore battery condition when charging. Potentially Dangerous.	Yes	Yes	OC
PowerBoardVersion	71	Version of the power board that is running the system	Yes	No	Both
AccessLevel	78	Current level of access that external entities have to the system.	Yes	Yes	Both

Parameter Status Codes

Name	Code	Description
Failure	0	The parameter was not set due to a general failure
Hardware Failure	1	Some hardware did not respond as expected. The parameter was not set.
Invalid Input	2	The data that was to be written to the parameter was not valid for the parameter.
Non-critical Fail	3	The data was not written to the parameter, but this should be anticipated for the given parameter
Read only	4	The parameter is currently in a read only state and should only be read
Success	5	The data to be written to the parameter was written successfully
Not Authorized	6	The current session was not authorized to change the selected parameter

Real-time ADC Packets

Name	ID	Data Type	Description
PacketCount	0	uint32_t	Tick timebase in milliseconds
ChargeState	1	uint8_t	Current state of system
Flags	2	uint16_t	Flags

PowerLevel	3	uint16_t	Output power level
VMon3v3	4	float	3.3v voltage
VMon5v	5	float	5v voltage
IMon5v	6	float	5v current
VMon12v	7	float	12v voltage
IMon12v	8	float	12v current
VMonGateDriver	9	float	Gate Driver voltage
IMonGateDriver	10	float	Gate Driver current
VMonPA	11	float	Power Amplifier voltage
IMonPa	12	float	Power Amplifier current
TMonPa	13	float	Power Amplifier temperature
VMonBatt	14	float	Battery Voltage
VMonBattProg	15	float	Charger Voltage
VRect	16	float	Rectified Voltage
TBoard	17	float	Board Temperature
ICharger	18	float	Charger current
IBattery	19	float	Battery current
TargetIBatt	20	float	Target battery current
IMaster	21	float	Charger 1 current
ISlave1	22	float	Charger 2 current
ISlave2	23	float	Charger 3 current

Device ID

Device	Address
Transmitter	1
Charger	2

APPENDIX B: PYTHON HELPER LIBRARY

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.

Setup

The libraries and sample code require Python 3.6 (or newer) to be installed and the sample code was developed 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!

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 wiboticssample.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

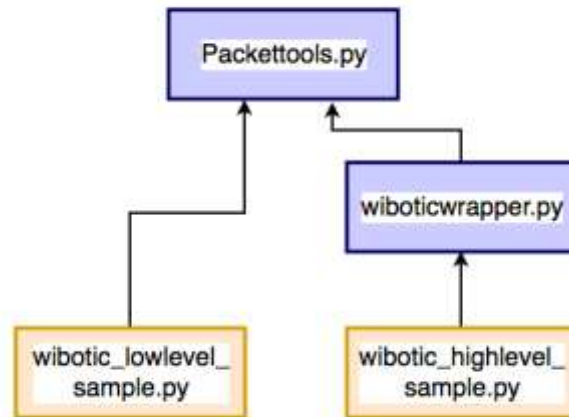
Receiver Battery Charge Current: 0.00A

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.

Python Development

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



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 Onboard Charger/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 Onboard Charger/Receiver build versions and MAC addresses. Finally, it enters a loop to show the most recently received Transmitter and Onboard Charger/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.

APPENDIX C: ONBOARD API

Similar in functionality to the Network API, the Onboard API allows for direct communication between your robot/drone and the WiBotic Onboard Charger. Rather than using Ethernet, however, the Onboard API uses a Control Area Network (CAN) interface as a means of communication - specifically “UAVCAN” a lightweight protocol used for communication in robotic and aerospace applications (<https://uavcan.org/>). With this this interface, users have complete access to the WiBotic Onboard Charger for monitoring and configuration of charge settings directly from CAN enabled flight or robot controllers.

CAN Interface



Figure 2: WiBotic On-Board Charger Connections



Figure 3: WiBotic On-Board Charger Connections with J4 (J3 on OC300) JST CAN Connector

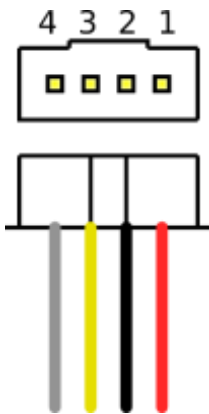


Figure 1: J4 (J3 on OC300) Pinout

Pin	Function
1	NC
2	CAN Low
3	CAN High
4	GND

USB to CAN

Pin	Function
1	NC
2	CAN Low
3	NC
4	NC
5	CAN High
6	NC
7	GND
8	NC
9	NC

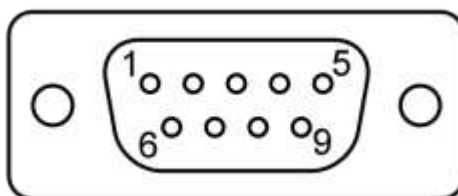


Figure 5: USB2CAN Pinout



NOTE: A 120 Ω terminator resistor must be placed between CAN High and CAN Low as per ISO-11898 standard.

It is possible to communicate with an Onboard Charger via CAN through any device that has a USB port by using a USB2CAN (<https://www.8devices.com/products/usb2can/>) converter (or similar) and adapter cable. The adapter cable must be plugged into the USB2CAN device with the wires connecting the JST connector for the CAN bus on the on-board charger.



CAN API

WiBotic's CAN API uses the UAVCAN protocol (<https://uavcan.org/>). UAVCAN handles the packet serialization and communication when using the API through the CAN bus. How information is sent through UAVCAN is determined by its DSDL (data structure description language) definition.

Adjustments to the data sent out from the on-board charger and the ID of the CAN bus can be changed by modifying the *CANMessageConfig* and *CANID* NVM parameters on the on-board charger. For access to all data accessible via the CAN API use custom WiBotic provided DSDL definitions.

Basic Setup

Python is a programming language that allows for quick setup and execution of commands.

Detailed documentation of the programming language itself can be found at:

<https://www.python.org/doc/>. When using the CAN API via Python it is strongly recommended that Python 3 is used.

Windows

1. Navigate to the official Python website and download the latest version of Python 3: <https://www.python.org/downloads>. For this guide we chose to download the executable installer.
2. Run the downloaded executable.
3. Make sure the “Add Python 3.x to PATH” box is checked and click “Install Now.”
4. Install a CAN driver for Windows (See vendors best practice). Use the vendor specified CAN tools to set the CAN interfaces bitrate to 500000.
5. Open a Command Prompt
6. Run: `pip install uavcan`

Now you can either select the Python executable or type `python` into the command prompt to use Python.

Linux (Debian based)

1. Make sure the systems package information is up to date by running: `sudo apt-get update`.
2. Install Python 3 by running: `sudo apt-get install python3`.
3. Install PIP by running: `sudo apt-get install python3-pip`.
4. Install the UAVCAN Python library by running: `sudo -H pip3 install uavcan`.
5. Install a native CAN interface package (e.g. SocketCAN or pyserial). For this example, we install SocketCAN by running: `sudo apt-get install can-utils`.
6. If necessary for your CAN hardware, load the CAN kernel modules. Most CAN adapters do not need these modules; however, built in CAN interfaces like on a BeagleBone generally do:
`sudo modprobe can`
`sudo modprobe can-dev`
`sudo modprobe can-raw`
7. Bring up the CAN interface by running: `sudo ip link set up can0 type can bitrate 500000`.
8. Confirm CAN interface is running by checking: `ip addr`.
9. (Optional) For debugging purposes when using SocketCAN running the command `candump can0` will display the raw communications happening on the CAN bus.

`candump` should result in a stream of data that looks similar to the image.

```
can0 012 [8] DA EB B1 68 03 4A 00 80
can0 012 [8] 00 00 00 00 00 00 00 20
can0 012 [8] 00 00 00 00 00 00 00 00
can0 012 [8] 00 00 00 00 57 69 42 20
can0 012 [6] 6F 74 69 63 00 40
can0 012 [8] D8 03 03 4A 00 00 C4 81
can0 012 [8] 2C 00 00 B1 68 00 00 21
can0 012 [7] 00 00 00 00 00 00 41
```

Talking over CAN

In order to talk to an Onboard Charger across CAN, the UAVCAN protocol must be setup. For ease of use it is recommended that `pyuavcan` (which was installed as a part of setup) is used. More descriptions of how to use `pyuavcan` can be found here:

<https://uavcan.org/Implementations/Pyuavcan/Tutorials/2. Basic usage/>.

Setting up a UAVCAN Node in Python

1. Open a Python 3 terminal.
2. Import UAVCAN by typing: `import uavcan`
3. Setup node information:

```
# Creates a Node Info object
node_info = uavcan.protocol.GetNodeInfo.Response()

# Set the name of the object
node_info.name = 'org.uavcan.pyuavcan_demo'

# Set the version of the node
node_info.software_version.major = 1

# Gives the node a unique ID
node_info.hardware_version.unique_id = b'12345'
```

4. Initialize a node to your device (make sure to replace `can0` with your device name if it differs): `node = uavcan.make_node('can0', node_id=123, node_info=node_info)`

After this the node variable is all set and ready to use

Including WiBotic's DSDL definitions

The function `uavcan.load_dsd1(PATH)` can be used to load WiBotic's DSDL definitions as a third party definition. This will add the `thirdparty` path to the definition (e.g. `uavcan.thirdparty.wibotic.*`).

Broadcasting a Message Over the CAN Bus with a UAVCAN Node

For this example, we will send a WiBotic command to change the Onboard Charger's configured number of battery cells to 3 across the CAN bus.

1. Make sure to have already setup your node. We will be using the variable name "node" for this example.
2. Create a `WiBoticCommand` object (NOTE: lines starting with "#" are interpreted as comments in Python):

```
# Creates the WiBoticCommand object

message = uavcan.thirdparty.wibotic.WiBoticCommand()

# Number for RxBatteryNumCells (configured number of cells
# for a receiver's
# battery).

message.command = 62

# Sets the desired value for RxBatteryNumCells to 3

message.payload = 3

# Broadcasts the message through the CAN bus

node.broadcast(message)

# When executing outside of a Python terminal the node has
# to spin for # actions to be processed. Adding a timeout
# allows code to continue # running after the given
# timeout.

node.spin(timeout=1)
```

Receiving a Message Over the CAN Bus

In this example a `WiBoticRead` will be sent to the Onboard Charger and then the message will be deserialized. As a result, we will read how many battery cells the Onboard Charger is configured for.

1. Make sure to have a UAVCAN node setup.
2. Create a callback for incoming messages and register the callback with the UAVCAN node. It should be noted that in order to display messages while still using a terminal a separate thread must be created or this should be run in a different terminal:

```
# Defines a function that will be run when a message is
received (callback function)
def node_read_callback(event):
    print('Incoming message from:',
          event.transfer.source_node_id)
    print(uavcan.to_yaml(event))

# Creates a handler for receiving a WiBoticCommand (the
type that is      # replied with when asking to read a
certain parameter.
handle =
node.add_handler(uavcan.thirdparty.wibotic.WiBoticCommand,
                  node_read_callback)

# Node must be spinning in order for the handler to
recognize an      # incoming message and route it to the
right location. This may need to # be in a separate thread.
Put this into a separate thread if using the # Python
terminal.
node.spin()
```

3. Optionally, if a separate thread is needed (i.e. if you still want to use the python terminal) run to spin the node in a separate thread:

```
import threading
threading.Thread(target=node.spin, daemon=True).start()
```

4. Create and send a WiBoticRead message:

```
# Creates the WiBoticRead object

message = uavcan.thirdparty.wibotic.WiBoticRead()

# Sets the parameter to read from the receiver

message.parameter_id = 62

# Broadcasts the message through the CAN bus

node.broadcast(message)
```

```
# When executing outside of a Python terminal the node has
to spin for # actions to be processed. Adding a timeout
allows code to continue # running after the given
timeout.

node.spin(timeout=1)
```

Within a second or two you should see the message from the callback function appear with data from the Onboard Charger on the CAN bus.

Serializing Packets

General Concepts

DSDL (data structure description language) is a UAVCAN provided format to define data structures that can be read and written by the UAVCAN protocol. A DSDL definition is a description of how the data will be serialized and deserialized during communication over the CAN bus.

Just setting up UAVCAN on a system will enable communication with any of UAVCAN's default DSDL definitions. WiBotic's CAN API has support for the BatteryInfo definition; however, only the temperature, voltage, and current will have values filled in.

For more in-depth information WiBotic's custom DSDL definitions can be used: WiBoticCommand, WiBoticRead, and WiBoticInfo. Full details about each of WiBotic's custom DSDL definitions can be found under *WiBotic DSDL Definitions*.

Using DSDL Definitions

To use a DSDL definition an object for that definition has to be created and then the fields that need to be set must be filled out. For example, if using WiBoticCommand in order to commit all parameters:

```
# Creates the DSDL object

message = uavcan.thidparty.wibotic.WiBoticCommand()

# Sets relevant parameters for the DSDL object

message.command = 0xFFFFD
```

WiBotic DSDL Definitions

WiBoticCommand

WiBoticCommand is a way to send a specific action for the on-board charger to do. It works by setting the command equal to the NVM parameter ID that should be changed, or to a hex command for a specific action on the on-board charger. The payload should either be set to the value to overwrite for a given parameter, parameter ID to commit, or be left blank. After sending a WiBoticCommand to the on-board charger the charger will respond with another WiBoticCommand where the command is the same one that was just sent but the payload is equal to the response code (full list under the *Parameter Response Code* section).

Command	Payload	Description
ID (options listed under Parameter)	Value to write	Writes a parameter to be equal to the given value
0xFFFE	ID (options listed under Parameter)	Stages the given On-Board Charger NVMMable parameters
0xFFFD		Commits all staged On-Board Charger NVM parameters
0xFFFC		Returns the last returned parameter

WiBoticRead

WiBoticRead is a way to read the value of a given NVM parameter on the on-board charger. It works by setting the parameter ID (full list of compatible parameters listed under *Parameters*) to the ID of the parameter that should be read and sending that data to the connected on-board charger. The transmitter will then respond with a WiBoticCommand that has its command equal to the parameter ID it received and the payload equal to the requested read value.

Parameter_ID	Description
Paramter ID (options listed under Parameter)	Reads and returns the value of a given parameter

WiBoticInfo

WiBoticInfo sends the information available from the on-board charger's ADC packets across the CAN bus.

Parameter	Description
vmon_battery	Battery Voltage

imon_battery	Battery Current
vmon_rectified	Rectified Voltage
vmon_charger	Charger Voltage
tmon_board	Board Temperature
BatteryITarget	Battery Target Current
IMonCharger	Charger Current
ISenseBatt1	Charger 2 Current
ISenseBatt2	Charger 3 Current

Setting up WiBotic's DSDL Definitions as Non-Thirdparty

In order to fully use the CAN API WiBotic's DSDL definitions will have to be installed. In order to install the definitions:

1. Find the "site-packages" folder in your python installation. Typing the command:
`python -m site --user-site` into a terminal or command prompt should display the location of this folder.
 2. Navigate into `uavcan/dsdl_files`.
 3. Create a new folder named "wibotic"
 4. Copy and paste the WiBotic DSDL definitions into the folder you just created.
- You should now have access to the DSDL definitions just like any other definition (make sure to restart any python terminals you have running if the definitions do not immediately appear).

Parameters

Name	ID	Description	Read	Write
Address	3	Address of the device on the internal point to point wireless link	Yes	No
RadioChannel	4	Current device radio channel	Yes	No
DigitalBoardVersion	26	Version of the digital board that is running the system	Yes	No
BatteryCurrentMax	34	Maximum current that should be delivered to the battery in milliamps	Yes	Yes
ChargerCurrentLimit	35	Current limit that the system has decided can be delivered to the battery	Yes	No
MobileRxVoltageLimit	36	Calculated maximum battery voltage based on chemistry, number of cells, and voltage per cell	Yes	No
RxBatteryVoltageMin	37	Calculated minimum battery voltage based on chemistry, number of cells, and voltage per cell	Yes	No
BuildHash	38	Version hash of the firmware that is loaded on the device	Yes	No
TargetFirmwareId	39	Type of firmware image that is running on the device	Yes	No
RxBatteryVoltage	42	Last read ADC value of the battery's voltage	Yes	No
RxBatteryCurrent	43	Last read ADC value of the battery's current	Yes	No
RxTemperature	44	Last read ADC value of the power board's temperature	Yes	No
DevMACOUI	51	MAC Address Organizationally Unique Identifier	Yes	No
DevMACSpecific	52	MAC Address Specific Identifier	Yes	No
ChargeEnable	60	Enable or disable the ability for this device to transfer power or charge a battery	Yes	Yes
I2cAddress	61	Address of this device on a I2C bus	Yes	Yes

RxBatteryNumCells	62	Number of cells in the battery this charger is configured to charge	Yes	Yes
RxBatterymVPerCell	63	Voltage of a battery cell (in millivolts) that this charger is configured to charge	Yes	Yes
RxBatteryChemistry	68	Chemistry of the attached battery	Yes	Yes
IgnoreBatteryCondition	70	Ignore battery condition when charging. Potentially Dangerous.	Yes	Yes
PowerBoardVersion	71	Version of the power board that is running the system	Yes	No
AccessLevel	78	Current level of access that external entities have to the system.	Yes	Yes
CANMessageConfig	82	Bitset for which DSDL definitions should be used to send ADC packet information. 0: None 1: BatteryInfo 2: WiBoticInfo 3: Both	Yes	Yes
CANID	83	ID of the CAN node on the Onboard Charger	Yes	Yes

Parameter Status Codes

Name	Code	Description
Failure	0	The parameter was not set due to a general failure
Hardware Failure	1	Some hardware did not respond as expected. The parameter was not set.
Invalid Input	2	The data that was to be written to the parameter was not valid for the parameter.
Non-critical Fail	3	The data was not written to the parameter, but this should be anticipated for the given parameter
Read only	4	The parameter is currently in a read only state and should only be read
Success	5	The data to be written to the parameter was written successfully
Not Authorized	6	The current session was not authorized to change the selected parameter
Pending	7	Item is waiting to be processed
Value Clamped	8	The set value has been limited into a safer range for the given parameter

WiBotic Inc.

4545 Roosevelt Way NE Suite 400

Seattle, WA 98105

650-265-7987

www.wibotic.com

info@wibotic.com