
IoTaWatt Documentation

Release 02_03_20

Bob Lemaire

Mar 12, 2021

Contents:

1	Quickstart	1
2	Installation	3
2.1	Software vs. Hardware	3
2.2	Components	3
2.3	Voltage and Frequency	3
2.4	Connections	4
3	Connecting to WiFi	5
3.1	Purpose	5
3.2	New connection	5
3.3	Resetting WiFi to Defaults	7
4	Device Configuration	9
4.1	Device name	10
4.2	TimeZone	10
4.3	Auto-update Class	10
5	Voltage Transformer Configuration	11
5.1	VT Model Selection	12
5.2	Voltage Calibration	13
6	Configuring Power Channels (CTs)	15
6.1	What is a power channel?	15
6.2	Connecting the CTs	16
6.3	Configuring the Input Channels	16
6.4	Generic CT	19
6.5	Enable derived three-phase	20
7	Device Status Display	21
7.1	Overview	21
7.2	Inputs/Outputs	22
7.3	Statistics	22
7.4	Web Servers	23
7.5	Data Logs	23
8	Outputs	25

8.1	Adding a new Output	25
9	Web Servers	31
9.1	Sending Data to a Web Service	31
9.2	PVoutput	32
9.3	influxDB	32
9.4	Emoncms	34
10	PVoutput	35
10.1	Create a PVoutput Account	36
10.2	Add Your System	36
10.3	Configure IoTaWatt	36
10.4	Reload History	38
11	Emoncms	39
11.1	Setup Emoncms	39
11.2	Configure IoTaWatt	39
11.3	Customizing Input data	41
12	influxDB	43
12.1	Configure IoTaWatt	43
12.2	tag-set	45
12.3	measurements	46
12.4	Variables	47
13	Three-phase Power	49
13.1	Configuring Direct Reference	49
13.2	Configuring Derived Reference	51
13.3	Reporting Power	53
14	CT Basics	55
14.1	What is a Current Transformer?	55
14.2	Types of CTs	55
14.3	Installation	57
14.4	Polarity	57
14.5	Single and three-phase systems	57
14.6	Split-phase systems	57
14.7	240V Split-phase circuits	62
15	Split-Phase Installation	65
15.1	What is split-phase?	65
15.2	Split-phase load centers	65
15.3	Monitoring split phase	65
15.4	Voltage Reference	66
15.5	Mains CT orientation	66
15.6	Load CT orientation	67
16	Data Visualization	69
16.1	Graph+	69
16.2	Original Graph	69
17	Graph+	71
17.1	Unit/Source selector	72
17.2	Time period selector	74
17.3	Graph window	76

17.4	Trace tables and options	77
17.5	Saving Graphs	81
17.6	Running Directly with URL	82
17.7	Reset	82
18	Original Graph	83
19	File Manager and Editor	87
19.1	IoTaWatt file systems	87
19.2	File Manager	87
19.3	Downloading Files	88
19.4	Uploading Files	88
19.5	SPIFFS	88
19.6	ACE Editor	89
20	Query API	91
20.1	Overview	91
20.2	Query types	91
20.3	query?show	91
20.4	query?select	92
20.5	time specifiers	94
20.6	Responses	96
21	Message Log	99
22	Troubleshooting	101
22.1	Led Indicator	101
22.2	Dull Green Glow	101
22.3	Dull Red Glow	101
22.4	Not Illuminated	102
22.5	Continuous Red-Green-Red-Green.	102
22.6	Led Sequences	102
23	iotawatt.local	105
23.1	How does it work?	105
23.2	How does that <i>not</i> work?	105
23.3	How to make it work better.	106
23.4	What else?	106
23.5	The last word	106

CHAPTER 1

Quickstart

So now you have your new IoTaWatt and can't wait to hook it up and start monitoring - but you couldn't find the quickstart guide in the box. That's because there is no quickstart guide per-se.

However... The first few topics that follow contain all you need to know about getting up and running, and will get you going pretty quickly. Each step produces a result and at the end you will know how to do all of the basics.

So just press "Next" below and get started on the first topic.

2.1 Software vs. Hardware

Strictly speaking, IoTaWatt began as a software project with no specific hardware platform in mind. That objective was soon supplanted by an open hardware design based on the ESP8266. Typical IoTaWatt installations use a commercially manufactured unit that is certified to UL and CE safety standards and complies with FCC requirements.

The IoTaWatt firmware is factory installed and can be automatically and securely updated over WiFi. This section of the documentation deals with the physical installation of the commercial IoTaWatt unit with factory installed firmware.

2.2 Components

IoTaWatt has 15 input channels. One is reserved for monitoring voltage, and the remaining 14 can be used to monitor power *or* voltage. The minimum requirements for operation are:

- IoTaWatt
- USB power supply
- AC voltage reference transformer
- Current Transformer (one or more)

2.3 Voltage and Frequency

IoTaWatt can work with all common line voltages at 50Hz/60Hz. Additionally, it can use multiple voltage references for polyphase measurement. The most common environments are 120/240V 60Hz and 230V 50Hz. The USB power supply and AC transformer must be appropriate for the power system.

2.4 Connections

Setup is simple:

1. Connect the USB power supply to the 5V DC USB
2. Connect the AC transformer to the 9V AC REF
3. Plug the power supply and AC transformer into a wall socket

That's it.

Next step: [connecting to WiFi](#)

Connecting to WiFi

3.1 Purpose

When IoTaWatt powers up, it attempts to connect to the last WiFi network used. A connection must be established to an internet connected network if the real time clock is not set, or to run the configuration utility, or if logging to an external server like Emoncms is desired. *It is strongly recommended to connect the IoTaWatt to a WiFi network with a reliable internet connection.*

If the power-up connection is successful, the LED will begin to glow dull green. This indicates that the IoTaWatt has connected and is in normal operating mode. At this point you can skip ahead to the next section [Device Configuration](#)

3.2 New connection

If the IoTaWatt has never been connected to a WiFi network (new), or the last used network is not available, the ESP8266 will enter AP (Access Point) mode when powered-up (not on a software restart). This state is indicated when the LED flashes the three color sequence RED-GREEN-GREEN. It will broadcast an SSID recognizable by the prefix `iota` followed by a unique number. Connect to it using a smartphone or tablet. Use the password `IotaWatt` (case sensitive). If the device was previously configured and the device name was changed, that new name will be the password required in this step.

After connecting, a page will be rendered with several choices. Select `Configure WiFi`.

192.168.4.1
iota427289

Log In

iota427289

WiFiManager

Configure WiFi

Configure WiFi (No Scan)

Info

Reset

A few seconds may elapse while the IoTaWatt scans for the local networks, then another page will be rendered allowing you to select one of the listed networks, or specify another network not listed.

Note that the IotaWatt only supports 2.4 GHz wireless networks. If you have a 5 GHz only network you can either enable 2.4 GHz on it, or create a separate (optionally hidden) network on 2.4 GHz for the IotaWatt to use.

192.168.4.1
iota427289

Log In

iotalwifi	 100%
flyaway-guest	76%
flyaway	 76%
flyaway_EXT	 14%

save

[Scan](#)

Select your network and enter the password, then save. Once connected, the new WiFi network credentials will be saved and the device will continue its startup procedure. If you see another LED sequence, refer to the [troubleshooting](#) section.

When the LED glows dull green, proceed to the next step [Device Configuration](#)

3.3 Resetting WiFi to Defaults

If you wish to change the WiFi SSID that the IoTaWatt is connected to (you might have done some testing in the lab and want to deploy 'live' somewhere where the SSID is different) BEFORE moving the device you need to reset the WiFi.

In the URL bar of your browser type:

```
> IoTaWatt.local/command?disconnect=yes
```

You could replace IoTaWatt.local with the IP address of your device.

After the command to disconnect the existing WiFi connection has been sent, IoTaWatt will respond “ok” and the LED will change from a dull green to a dull red as the IoTaWatt disconnects, indicating that the WiFi link has been severed.

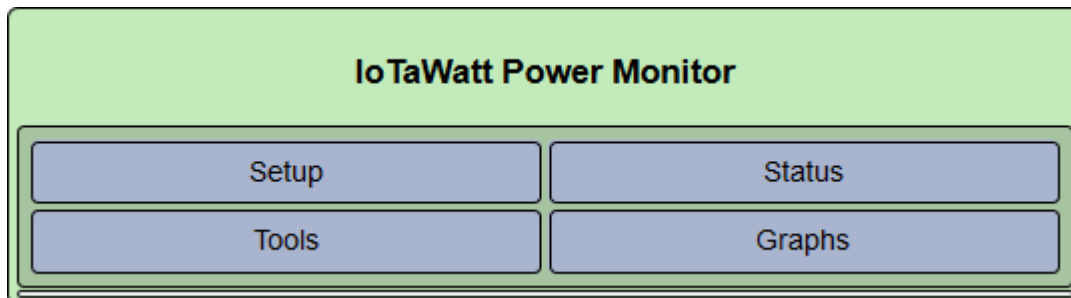
Then power cycle the IoTaWatt. It should restart with the RED-GREEN-GREEN led sequence like a new IoTaWatt ready to connect to the new SSID

CHAPTER 4

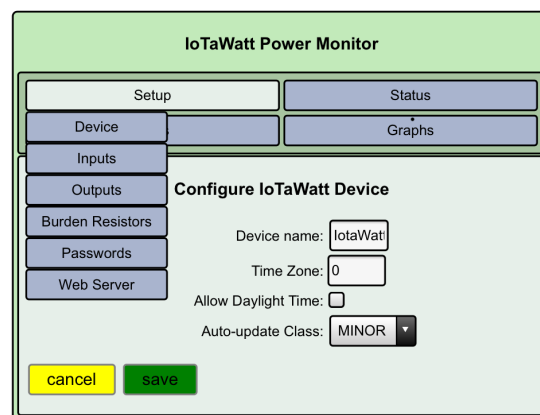
Device Configuration

Successful startup will be indicated by a dull green glow on the LED. If the LED is off, or blinking a sequence, see the [troubleshooting](#) section.

After successful startup, you can connect to the device with your web browser to access the configuration app. Use the url: `http://iotawatt.local` or, if your device has been renamed, use `http://<newname>.local`. The configuration app starts with a row of buttons:



Hover over  and click  in the dropdown menu.



4.1 Device name

You can change the `Device name` to another 8 character name if you wish. If you have more than one IoTaWatt you will need to do this so that they each have unique names. When you press save, the IoTaWatt will restart using the new name. From then on, `http://<newname>.local` will be the url that you will use to access the device from your browser, and the new `Device name` will be the password that you must use to connect to the AP if configuring for a different WiFi network.

4.2 TimeZone

Set your local Time Zone relative to UTC time. All of the measurements are time stamped using UTC, but `log messages` and various reporting apps will use this offset to show the data in local time.

If your time zone is subject to “Daylight Saving Time”, check the `Allow Daylight Time` box. IoTaWatt has DST rules for most of North America, Europe, Australia and New Zealand.

4.3 Auto-update Class

Auto-update Class tells IotaWatt if you want to receive automatic updates of the firmware and what type of updates you are interested in. The choices are:

NONE Do not Auto-update this device.

MAJOR Only update major releases of the software.


MINOR Update with minor releases. More frequent but somewhat tested firmware. This is recommended.

BETA Latest production firmware.

ALPHA Recently released firmware with the latest features - and the latest bugs.

IotaWatt checks the IotaWatt.com site for new software regularly. The update process takes less than a minute. New firmware is authenticated with a digital signature from IotaWatt and installed automatically.

4.3.1 Save

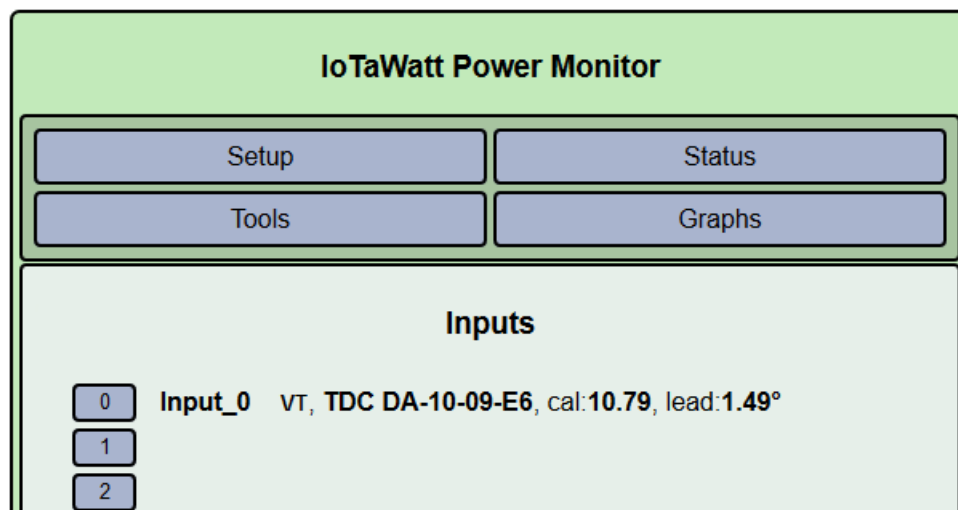
Click . Your changes will be saved. If you changed the name of your device, it will restart when you press save and you will need to restart the configuration application from `http://<newname>.local`.

The next step is [VT Configuration](#)

Voltage Transformer Configuration

A prime component of electrical power is voltage. The AC line frequency is the heartbeat of the IoTaWatt. A reliable and accurate AC voltage reference is very important. You should have installed the device with a 9 Volt AC voltage reference transformer (VT) plugged into the channel zero power jack. If your initial configuration has this channel pre-configured, your LED will be glowing green because it's rhythmically sampling that voltage.

Various transformer models produce different voltages, and it's important to insure that the VT specified matches the model that you have installed. To do this, select the inputs button in the Setup dropdown menu.



A list of all of the inputs will be displayed. The first entry will be input 0 and a default VT should be configured. Check to see if it's the same as your VT model. It's OK to unplug the VT to check the model number printed on it.

If your VT model doesn't match the model that is configured, you can easily change it. Click on the input channel 0 button on the left.

Configure Input 0

Burden: none configured.

Name:

Type:

Model:

☐ Reverse ⓘ

As you can see, the display changes to reveal the details of the input_0 configuration.

Configure Input 0

Burden: none configured.

Name:

Type:

Model:

- generic
- generic120V
- generic240V
- DCSS AC910(Aus)
- Powertech MP-3027(Aus)
- Ideal 77DB-06-09(UK)
- Ideal 77DE-06-09(EU)
- Ideal 77DA-10-09-MI(USA)
- TDC DA-10-09
- TDC DA-10-09-E6
- TDC DE-10-09(EU)**
- Jameco/100061
- Jameco/102234
- Jameco/2227444
- CUI 41A-9-1000

5.1 VT Model Selection

If your make and model is listed, select it from the list. At this point, you can just click and the standard calibration for your VT will be used. That calibration should be good for all but the most discerning users. If you have access to a good voltmeter or other reliable high accuracy voltage reference, you can fine tune with the calibration procedure below, but for average users, you should be good to go on to the next step Adding Power Channel CTs

If your VT wasn't listed in the dropdown above, the generic entry is a reasonable starting point that will get you in the ball park for your 9-12Vac adapter. If your country is 230V or 240V select "generic240V". Now you must perform the *Voltage Calibration* procedure below.

5.1.1 TDC DA-10-09 model ambiguity

There are two different voltage transformers available with the model designation TDC DA-10-09. These models are quite different and need to be properly configured.



Fig. 1: use model: TDC DA-10-09

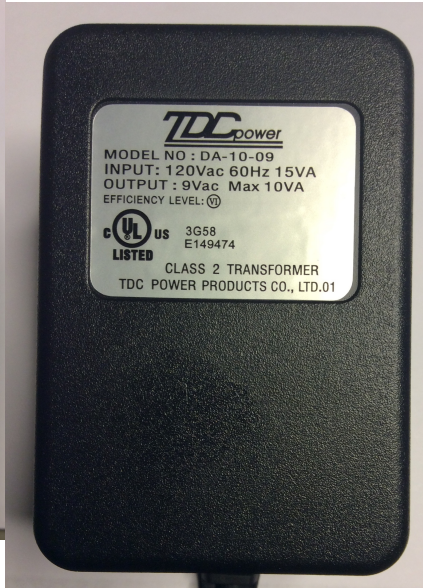


Fig. 2: use model: TDC DA-10-09-E6

5.2 Voltage Calibration

Again, if you are using one of the standard voltage transformers from the tables, this step is optional. Repeated random tests on the standard US and Euro transformers yield excellent calibration right out of the box.

You will need a halfway decent voltage reference for this step. If you don't have a decent true RMS voltmeter and can't borrow one, go out and get a Kill-a-Watt. They cost less than \$20 (some libraries lend them out) and I've found their voltage readings are usually accurate.

click [calibrate](#)

Calibrate Voltage Channel 0

118.7

calibration factor: 10.79

cancel

save

Using a voltmeter to display the AC line voltage, adjust the calibration factor until the displayed voltage reasonably matches the voltmeter reading. Click save to update the channel with the new calibration factor.

Follow the instructions on the page. Increase or decrease the “cal” factor until the voltage shown settles down and is a pretty good match with your reference meter. It’s not possible to match exactly. 0.2V in a 120V installation is 0.2% variation. A good meter accuracy is 1% at best. Just try to get the two to dwell around the same set of fractional digits.

As instructed on the page, click save to record the calibration factor. The new calibration factor will take effect immediately. Click the Status menu button to display the voltage:

Inputs/Outputs Status

Inputs	Outputs
voltage: 118.6 Volts	

IoTaWatt Statistics

Firmware version: 02_03_20	644 samples per AC cycle
Running time: 5h 33m 48s	39.7 AC cycles sampled/second
free Heap: 26808	60.0 Hz

InfluxDB

Data Logs

Wait a few seconds then check that the voltage displayed is still in the ball park. If not, repeat the calibration procedure.

Once calibration is complete and verified, you will not need to do it again unless you change your VT transformer. The IoTaWatt has a very accurate internal calibration reference and will maintain its accuracy indefinitely. You should have no further need for the voltmeter.

Now the device is ready for the next step [Configuring Power Channel CTs](#)

Configuring Power Channels (CTs)

6.1 What is a power channel?

Power channels measure the current flow through a circuit and combine that with the reference voltage to determine power, expressed in watts, and to accumulate energy used, expressed in watt-hours. Current through a circuit is measured indirectly by installing a passive sensor, called a *current transformer* (CT), around one of the conductors in the circuit. CTs come in a various capacities, physical connection type, and electrical output.

The good news is that IoTaWatt supports a wide variety of readily available CTs, and many can be configured simply by selecting the model from a list. CTs are connected with 3.5mm stereo jacks (headphone jacks). The CTs that are sold at the [IoTaWatt stuff store](#) are manufactured with 3.5mm jacks.



Fig. 1: Echun ECS16-100 CT

6.2 Connecting the CTs

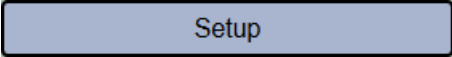
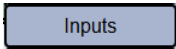
This tutorial does not cover physical installation of the CTs to your electrical circuits. That should be done by someone familiar with electrical wiring. Your qualified installer will know how to do this. The 3.5mm connectors plug into any of the 14 input channels on the IotaWatt.

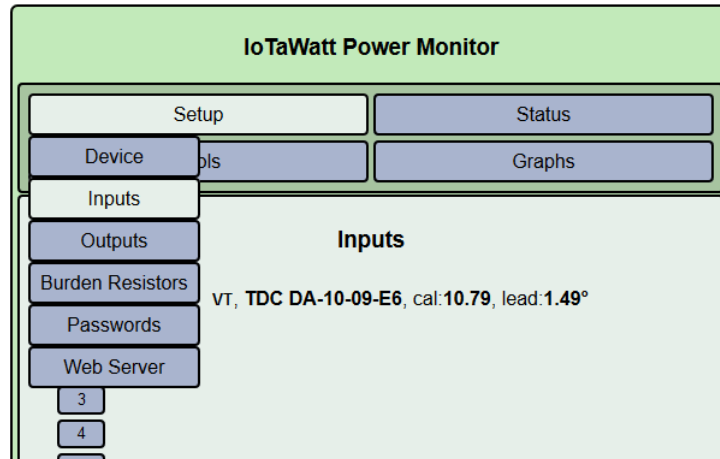
The only additional recommendation is that all of the CTs be oriented the same way with respect to current flow. Most CTs have an arrow or other marking to aid in consistent orientation. Not to worry, in the event some end up backward, IoTaWatt will still work, will tell you which ones appear to be backward, and provides a way to correct virtually.


You can find more detailed information about physical installation of CTs in the [CT Basics](#) section.

6.3 Configuring the Input Channels

At this point, you should have the IoTaWatt up and running with the voltage channel connected, configured, and calibrated if necessary. You are using the config app in a browser connected to your WiFi network. Hover over

 and click  in the dropdown buttons.



This screen should look familiar. We came here to configure the voltage transformer (VT). Now we will configure current transformers (CTs) to other inputs. To add or edit the CT specification for an input channel, click the channel's number button. Let's add a CT to channel .

Configure Input 1

Burden: 24 ohms

Name:

Type:

Model:

Turns:
Range is 120 to 21600

Phase lead:

☐ Allow negative power value ⓘ

☐ Double ⓘ

☐ Reverse ⓘ

The app enters channel edit mode. Here you specify the model and other details about the CT connected to this particular channel. But first, it is helpful to name the channel by typing a name in the name box. You can just use the default “Input_1” or something more meaningful like “Kitchen” or “Living Room” or “Main”. We will configure one of our US 120/240V split-phase mains as **main_1**.

The default type is **CT** and that’s correct.

The next drop-down box is the model of the CT. Initially it will be *generic*, Click the drop-down list and select the **ECS24200** CT.

Configure Input 1

Burden: 20 ohms

Name:

Type:

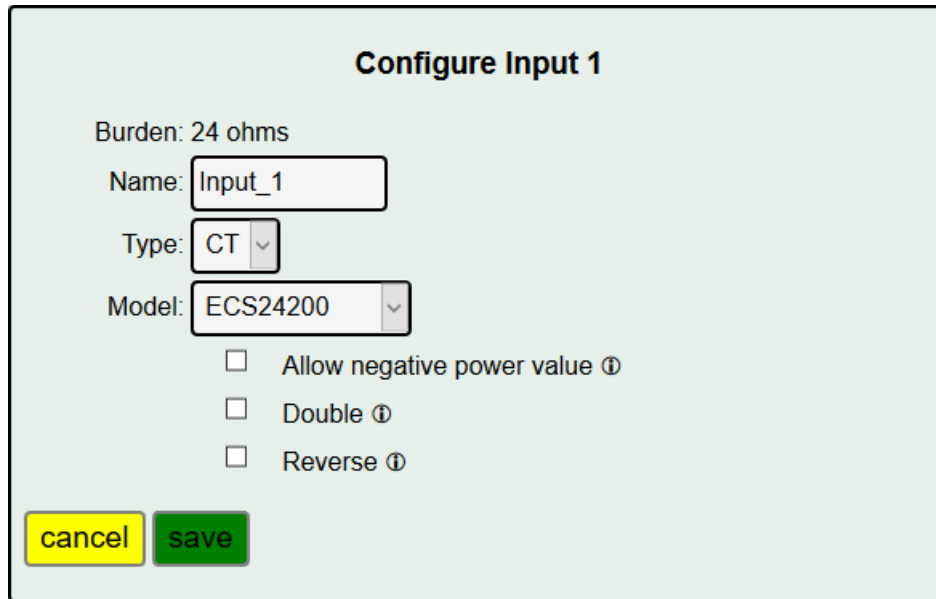
Model:

Turns:

Phase lead:

☐ Allow negative power value ⓘ

- generic
- CR3110-3000
- SCT006-000
- SCT013-000
- SCT019-000
- SCT023R
- SCT027H
- ECS25200-C2
- ECS24200**
- ECS24200
- ECS24200
- ECS24200
- ECS16100
- ECOL09
- ECS1050
- WC-1
- HWCT-004



Configure Input 1

Burden: 24 ohms

Name:

Type:

Model:

☐ Allow negative power value ⓘ

☐ Double ⓘ


☐ Reverse ⓘ

Notice that after selecting a specific device from the table, the input fields for “turns” and “phase” disappear. That’s because those values are known for the listed CTs. If you have a CT that is not found in the list, you will need specify the “generic” entry and provide the turns-ratio and phase-lead for that CT. see [generic CT](#) below.

There are check-boxes to further configure the CT. Most of the time, these will not be used, but there are circumstances where you would check one or more of them. If you hover on the ! next to each, a brief description will appear. They are explained below:

Allow negative power value This is typically checked only for mains in an installation with grid-tied solar (net-metering). Checking this box tells IoTaWatt that it is normal for current to flow backward through this circuit, as when a PV system creates more power than you are using locally and the balance is “exported” to the grid. When you check this box you are affirming that the CT is installed correctly and that negative power should not be automatically “corrected” to positive.

Double In North American split-phase power systems (120V/240V), all circuits are assumed to be 120V. When this option is selected, power will be computed using double the value of the reference voltage, or nominally 240V. Use this for 240V circuits where one CT has been applied to one of the conductors and there is no neutral (white) wire used by the appliance. Typical circuits would be Water Heater, Water Pump, Mini-Split Heat-Pump. There are other ways to monitor 240V circuits as well.

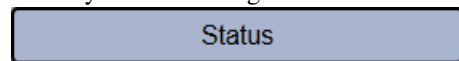
Reverse Sometimes a CT is installed backwards with respect to normal current flow. IoTaWatt will sense this and correct automatically in single-phase power systems. It will correct the negative value automatically and indicate so in the status display with a little  symbol. Selecting this option will virtually reverse the CT as if it were oriented correctly, obviating the need to physically reverse it. Doing so can be safer and/or easier, especially with solid core CTs. While merely convenient for single-phase systems, correct orientation is a necessity in three-phase installations because the IoTaWatt cannot automatically sense a reversed CT and correct for it.

Press  to finish.

Inputs	
0	voltage VT, TDC DA-10-09-E6, cal:10.79, lead:1.49°
1	main_1 CT, ECS24200, cal:200.00, lead:0.35°
2	
3	

That's it. The screen returns to the complete list of inputs where you can add more channels or change the configuration of existing inputs. Each time you press save, the new configuration is sent to IoTaWatt and the changes take effect immediately. If the CTs are installed and connected, you will can see the power displayed in the Input Channel [Status screen](#).

When you have configured all of the CTs connected to the IoTaWatt, basic configuration is complete. Click the



button to see the IoTaWatt in action.

The following additional information may provide guidance for more advanced installations.

6.4 Generic CT

We just configured a *Current Type* CT that was of a model known to IoTaWatt. If your particular CT is not one of the dropdown models, you will need to describe the **generic** parameters. You will recall that this is the initial model designation for a CT when a new channel is added. Its also a drop-down choice when editing a CT channel. With this model selected, you must specify additional information depending on the type of CT:

6.4.1 Current Type CT

Current type CTs are the most common type of CT used with IoTaWatt and all of the CTs available in the IoTaWatt *Stuff Store* are of this type. They are typically described by the ratio of the maximum primary current that they can measure and the corresponding secondary current that will be produced, as in 200A:50mA. For these CTs, you will be asked to specify the “Turns:”. This is the ratio of primary current/secondary current. So that 100A:50mA described above would be $100/.050 = 2000$ turns.

6.4.2 Voltage Type CT

Voltage type CTs are typically described with an output in volts (V) and have an internal burden resistor that causes them to produce an output voltage rather than current. They are connected to a modified IoTaWatt input that has had the internal burden resistor removed and specified as zero in the device configuration burden menu. IoTaWatt will ask for a **Cal** factor. This is the primary current in amps that corresponds to 1 volt of output from the CT. An example of this is the SCT013-050 from YHDC. It is marked 50A/1V, so the **Cal** is 50. Simple enough.

6.4.3 Phase

Both of the generic CT types above will also provide a place to specify **Phase**. Representative samples of the CTs in the model list have been tested to determine a phase correction value to compensate for phase shift of the transformer. If you have a generic CT a rule of thumb would be to use 2.0 for a split core CT (one that snaps onto a wire), and 0.2 for solid core CTs (Basically a solid doughnut that you pass the conductor through).

6.5 Enable derived three-phase

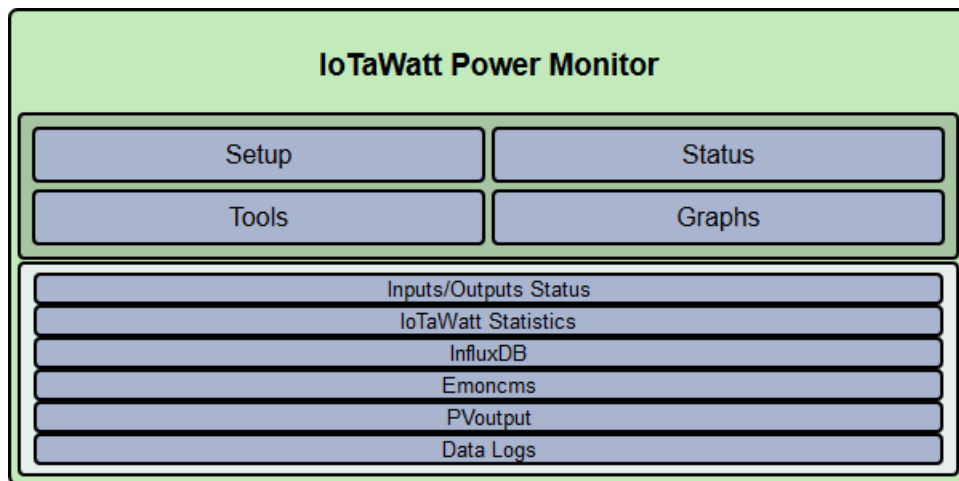
This checkbox enables advanced features used to configure inputs in a three-phase power system. Refer to the section [Three Phase Power](#) for more information.

Device Status Display

7.1 Overview

The configuration app can provide a continuously updating display of many aspects of IoTaWatt operation. The output is organized into expandable tabs containing various categories of information. Simply click the

Status button.



The initial display will always have the Inputs/Outputs Status tab expanded. Simply clicking on that tab, or any of the other tabs, will toggle them between expanded and collapsed. The various tabs as of this writing are detailed as follows:

7.2 Inputs/Outputs

Inputs/Outputs Status	
Inputs	Outputs
Voltage: 119.9 Volts	amps_main1: 5.72 Amps
Main_1: 596 Watts, pf: 0.87	amps_main2: 8.30 Amps
Main_2: 926 Watts, pf: 0.93	maindiff: 330.1 Watts
Heat_Pump: 976 Watts, pf: 0.84	misc: 96.6 Watts
Sub_Panel: 418 Watts, pf: 0.86	total_power: 1522.6 Watts
Hot_Water: 0 Watts	
Dryer: 0 Watts	
Oven_Stove: 3 Watts	
Fridge: 4 Watts	
Basement: 20 Watts	
Kitchen: 6 Watts	
HP2: 0 Watts	
Kitchen_HW: 0 Watts	
Dishwasher: 0 Watts	
IoTaWatt Statistics	
InfluxDB	
Emoncms	
PVoutput	
Data Logs	

This tab lists all of the configured input channels on the left, and all of the defined outputs on the right. Both columns show the current measurement value. Input channels values are in Watts or Volts for CT and VT channels respectively. CT channels also will display the power-factor (PF) when the power is sufficient to develop that reliably. Output channel values are in the units configured. They may be Volts, Watts, Amps, VA, Hz, or PF. All values are damped, which is to say they are averaged with an exponential decay algorithm so that they will not jump around excessively. That said, the algorithm does respond to large changes quickly and settles in on small changes within a few seconds.

7.3 Statistics

Inputs/Outputs Status	
IoTaWatt Statistics	
Firmware version: 02_03_20	642 samples per AC cycle
Running time: 18d 8h 58m 44s	38.8 AC cycles sampled/second
free Heap: 23024	60.0 Hz
InfluxDB	
Emoncms	
PVoutput	
Data Logs	

The statistics tab provides insight into the current operation of the IoTaWatt with the following information:

- **Firmware version:** Release of IoTaWatt firmware.
- **Running Time:** Since last restart.

- **free Heap:** An indication of the working memory available to the firmware.
- **Samples per AC cycle:** Average samples IoTaWatt is achieving when sampling a channel.
- **AC cycles sampled/second:** Average number of channels that are measured per second.
- **Hz:** Current AC frequency.

7.4 Web Servers

Inputs/Outputs Status
IoTaWatt Statistics
InfluxDB
<input type="button" value="Stop"/> Running, Last update 1/7/2019 10:33:10 PM
Emoncms
<input type="button" value="Stop"/> Running, Last update 1/7/2019 10:33:10 PM
PVoutput
<input type="button" value="Stop"/> Running, Last update 1/7/2019 10:30:00 PM
Data Logs

These tabs appear when [data upload](#) to PVoutput, Emoncms, influxDB has been configured. They indicate the state of the services that are responsible for sending the data to the respective server.

Start/Stop button: Use to pause and resume uploading data. This button works asynchronously, and may take awhile to perform the action. Pressing multiple time may cause it to go out of sync. Use it carefully.

Running/Stopped: The current state of the service.

Last Update: Data/time of the last data sent to the server.

7.5 Data Logs

Inputs/Outputs Status
IoTaWatt Statistics
InfluxDB
Emoncms
PVoutput
Data Logs
Current Log from: 12/27/2018 2:12:20 AM to: 1/7/2019 10:35:25 PM History Log from: 1/17/2017 1:37:00 PM to: 1/7/2019 10:35:00 PM

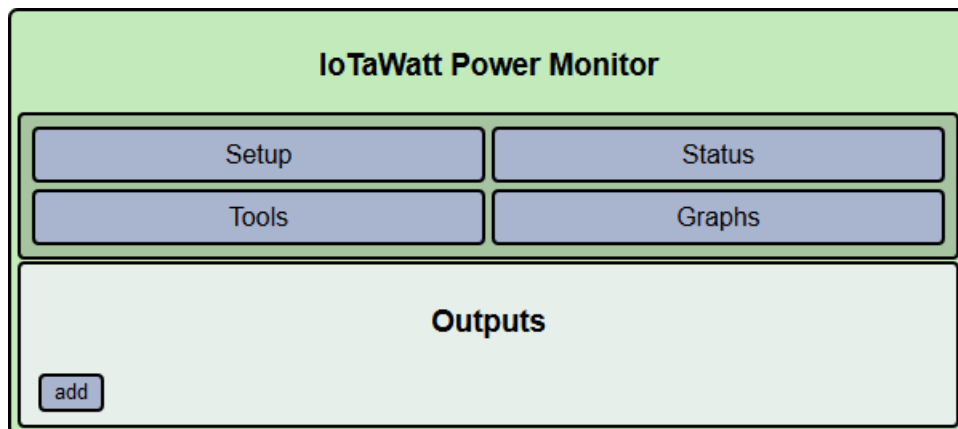
This tab displays the date/time of the first and last entry in both the Current Log and the History Log. These dates may take a few seconds to become accurate as the IoTaWatt starts. The History Log should begin from the data/time when the IoTaWatt was first installed, or when the History log was last created after being deleted. It should increment every minute while the unit is running.


The Current Log contains up to a year's worth of data, and maintains that data at 5 second resolution. When the approximate year capacity is reached, it will "wrap" back around and begin writing over the oldest entries. Newer logs will show the same start date as the History Log. As the log ages and reaches capacity, the start date/time will advance at 5 second intervals along with the ending date/time.


Outputs

Outputs provide useful values that are computed from input channel values using a calculator like interface. For instance, in a typical US installation, there are two MAIN circuits, the sum of the two is the total power into a panel. Its nice to know at a glance what that total is, but the two mains are measured separately using two input channels. We need a way to add them together to display the total usage.

Hover over  and click  in the dropdown buttons.

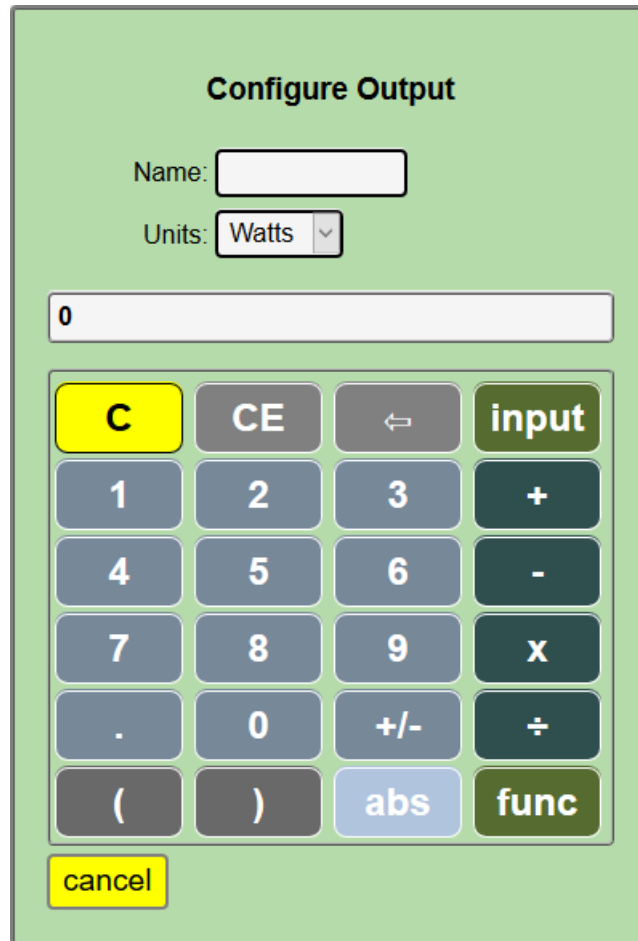


This screen will list any outputs that you have already configured, and allow you to click  to create new ones.

You can click  on existing outputs to change or delete them. There is no practical limit to the number of outputs that you may create. The only requirement is that they be uniquely named.

8.1 Adding a new Output

So lets click .



The image shows a 'Configure Output' dialog box with a light green background. At the top, the title 'Configure Output' is centered. Below it, there is a 'Name:' label followed by a text input field. Underneath, there is a 'Units:' label followed by a dropdown menu currently showing 'Watts'. Below the units, there is a large text input field containing the number '0'. The main part of the dialog is a calculator interface. It features a grid of buttons: a yellow 'C' button, a grey 'CE' button, a grey button with a left arrow, and a green 'input' button. Below these are rows of numeric buttons (1-9, 0, and a decimal point), arithmetic operators (+, -, x, ÷), and function buttons (parentheses, 'abs', and 'func'). At the bottom left of the calculator area is a yellow 'cancel' button.

This is the *calculator* interface that IoTaWatt uses to specify how to calculate an output using input channel values. A script is created that IoTaWatt uses to compute the value when needed. It works just like the simple four function

calculators we are all used to, and using the **input** key, you can select input channel values to be used in the formula that you are creating. The resulting expression is evaluated left to right, with calculations within parenthesis evaluated before being used.

So lets make an output channel that combines two main inputs called *main_1* and *main_2*. We enter the name *total_power* in the **Name:** box and hover over the **input** button of the calculator to see a list of the inputs.

Configure Output

Name:

Units:

C

CE

←

input

1

2

3

+

4

5

6

-

7

8

9

x

.

0

+/-

÷

(

)

abs

func

delete

cancel

save

Voltage

main_1

main_2

Heat_Pump

Sub_Panel

Hot_Water

Dryer

Oven_Stove

Fridge


Basement

Kitchen

HP2

Kitchen_HW

Dishwasher

Select main_1 from the list and it will appear in the calculator formula display. Next click on the , then repeat the input process selecting Main_2.

8.1. Adding a new Output

27

Configure Output

Name:

Units:

CCE←input

123+

456-

789x

0+/-÷

()absfunc

cancelsave

Easy as that. Now press **save** to return to the outputs list. Your new output should appear within a second or two.

Outputs

edit

total_power Watts = main_1 + main_2

add

Now go back to the Channels Status screen and see that the new output channel is listed and indeed has a value that is the sum of the two inputs *main_1* and *main_2*.

Inputs/Outputs Status	
Inputs	Outputs
Voltage: 119.7 Volts	misc: 94.4 Watts
main_1: 672 Watts, pf: 0.89	total_power: 1723.9 Watts
main_2: 1052 Watts, pf: 0.94	
Heat_Pump: 1131 Watts, pf: 0.87	
Sub_Panel: 460 Watts, pf: 0.88	
Hot_Water: 0 Watts	
Dryer: 0 Watts	
Oven_Stove: 3 Watts	
Fridge: 4 Watts	
Basement: 20 Watts	
Kitchen: 11 Watts	
HP2: 0 Watts	
Kitchen_HW: 0 Watts	
Dishwasher: 0 Watts	

Some other useful outputs would be:

- Power used in a solar PV system, calculated by adding the Solar inverter input to the (signed) Main input. If for instance the inverter were putting out 4500 watts and your Main(s) indicated an outflow represented as -3100 watts, local usage would be 1400 watts with 3100 watts exported.
- Where the Main(s) are monitored and selected circuits within the panel are also measured, you can create an output that shows the aggregate unmeasured usage by subtracting the measured inputs from the Mains as the *misc* output in the status display above. That output is defined:

Configure Output

Name:

Units:

$$\text{main_1} + \text{main_2} - (\text{Heat_Pump} + \text{Sub_Panel} + \text{Hot_Water} + \text{Dryer} + \text{Oven_Stove} + \text{Fridge} + \text{Basement} + \text{Kitchen} + \text{Kitchen_HW} + \text{Dishwasher})$$

C
CE
←
input

9.1 Sending Data to a Web Service

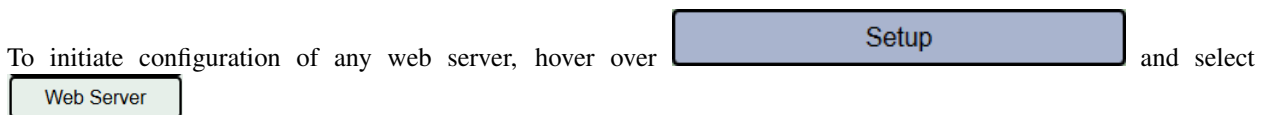
As a standalone unit, IoTaWatt is a very capable data logger with an integrated web server/API interface that can provide real-time as well as historical data, using the provided graph interface, or otherwise to any web connected client. That said, there are limitations in that queries can take several seconds and the web server currently works with locally connected clients on the same WiFi network and requires port forwarding and dynamic DNS to access from outside. There is also a chance that the local datalogs could be lost along with all of the accumulated history and there is no local backup capability yet.

There are cloud based services available that will store uploaded data logger information and present that data to client applications across the internet. Some of those databases can also be installed on a private server hosted by anything from a Raspberry Pi to a commercial hosting service. IoTaWatt has the capability to upload selected data while still maintaining all of its own local logging and reporting capabilities.

When one of these services becomes unavailable for any reason, IoTaWatt will upload the backlog when service is restored, whether that backlog is a minute or a month.

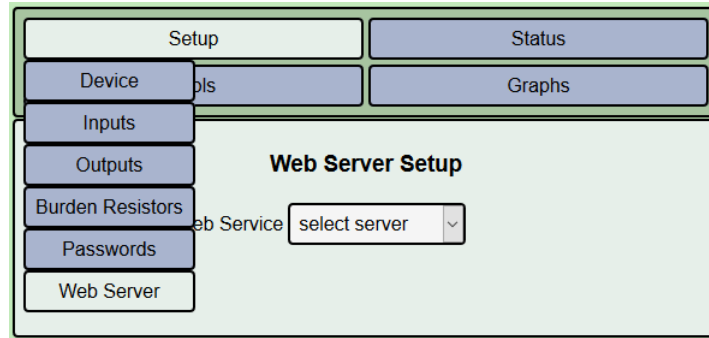
IoTaWatt can support multiple upload services simultaneously, so it's not necessary to sacrifice uploading to an influxDB database in order to participate in the PVoutput project.

To initiate configuration of any web server, hover over



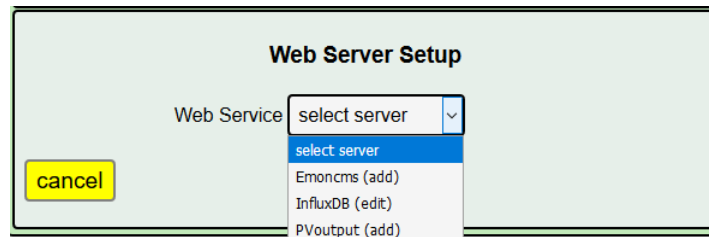
Web Server

Setup



click the **Web Service:** dropdown and select a server you would like to use from the dropdown list. Each service is designated as

- (add) - New service specification
- (edit) - modify existing service configuration (including delete)



When you select a server, the configuration menu for that particular server will appear. For details of configuring each unique server type, click on the heading of the corresponding section below.

9.2 PVOutput

PVOutput is a free online service for sharing and comparing photovoltaic solar panel output data. It provides both manual and automatic data uploading facilities.

Output data can be graphed, analysed and compared with other PVOutput contributors over various time periods. The ability to compare with similar systems within close proximity allows both short and longer term performance issues to be identified.

While pvoutput is primarily focused on monitoring energy generation, it also provides equally capable facilities to upload and monitor energy consumption data from various energy monitoring devices.

Both solar generation and energy consumption data can be combined to provide a live 'Net' view of energy being generated and consumed.

9.3 influxDB

influxDB is a free industrial strength, open-source, schema-less time series database. It is fast, efficient and scalable. You can install on a Raspberry Pi, a home server, or a commercial web host site.

There are several excellent visualization packages that can be used to visualize and report the data.



Welcome, PVOutput is a free service for sharing and comparing PV output data.

If you own a solar system please contribute your power output readings.

[Home](#) | [Latest Outputs](#) | [PV Ladder](#) | [PV Donut](#) | [Daily Outputs](#) | [Live Outputs](#) | [Teams](#) | [About](#) | [Register](#)

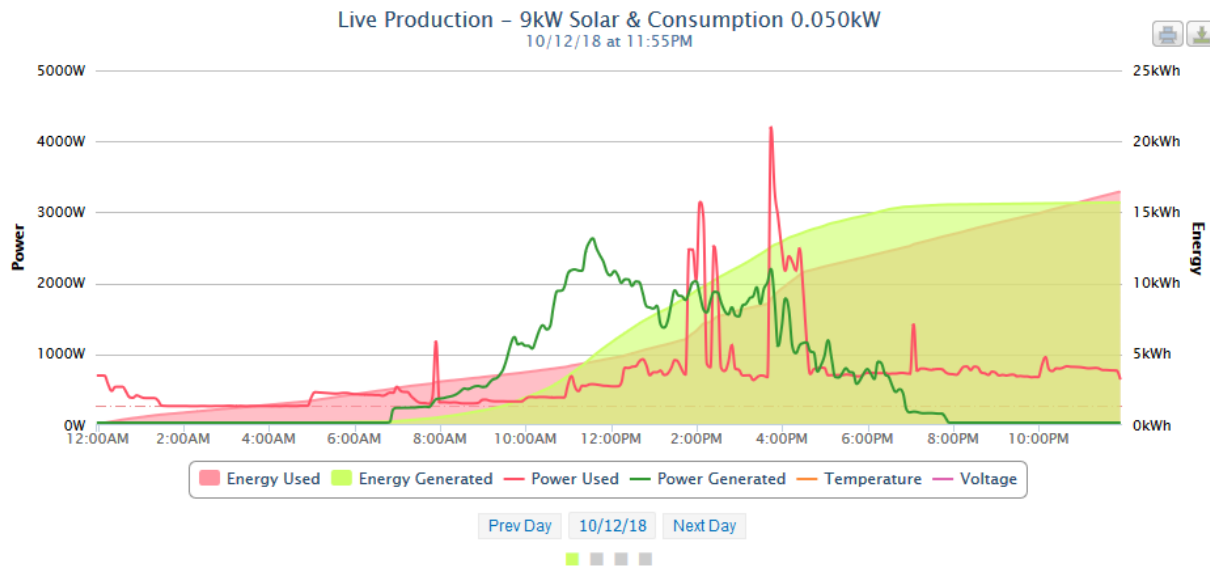


Fig. 1: PVoutput Graphic Display

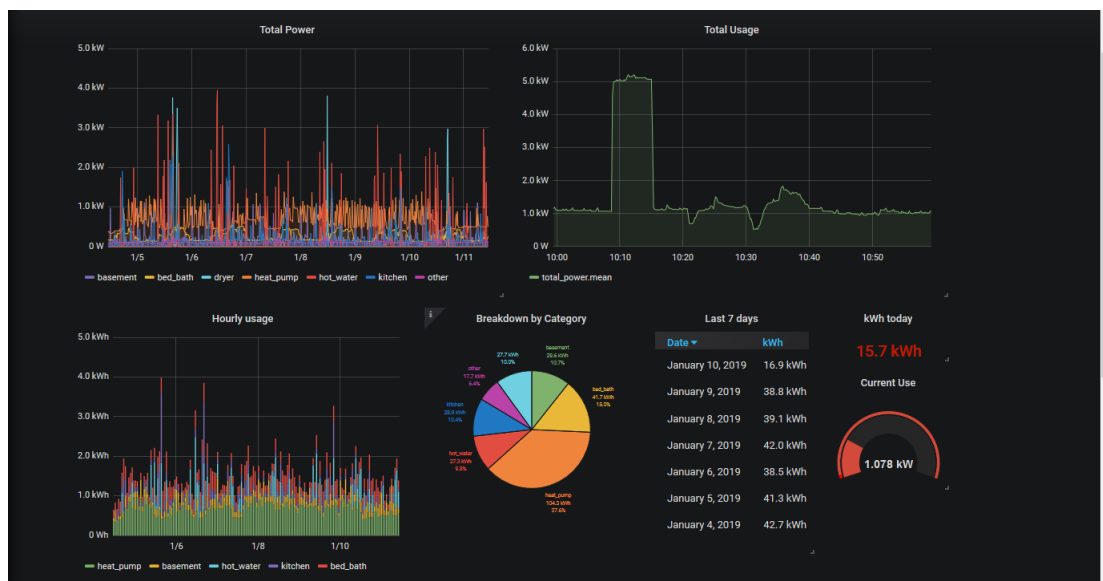


Fig. 2: grafana dashboard with influxDB

9.4 Emoncms

Emoncms is another open-source time series database that was specifically designed to handle energy monitoring data with robust set of visualization tools.

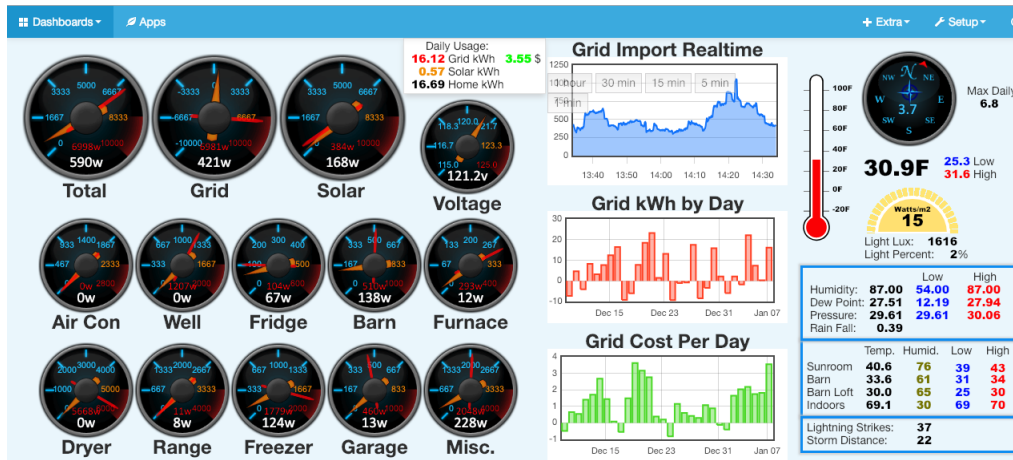


Fig. 3: An Emoncms Dashboard

Like influxDB, it can be local hosted on almost any machine, including Raspberry Pi, but also is available as a relatively inexpensive pay-as-you-go cloud service at emoncms.org.

CHAPTER 10

PVoutput

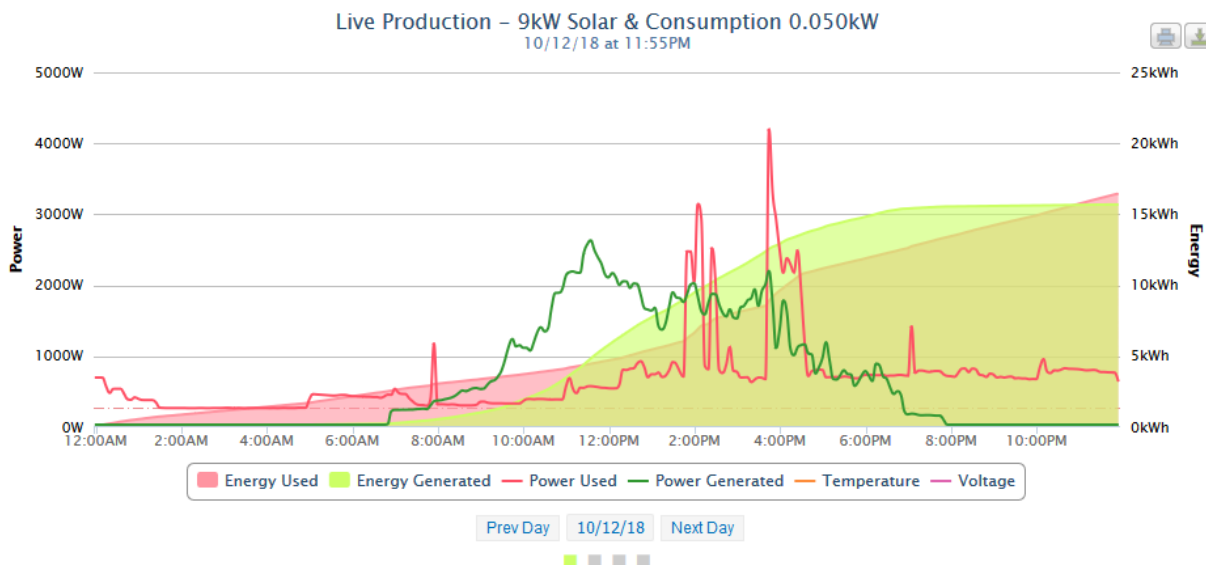
PVOutput is a free service for sharing, comparing and monitoring live solar photovoltaic (PV) and energy consumption data. It is a worldwide catalog of installed PV sites containing details of location, output, capacity, and efficiency. Individually, you can maintain data about the production of one or more sites along with voltage and consumption data. There is also an ability to use the service to capture and report up to six additional data items with the ability to generate real-time alerts based on simple rules. IoTaWatt supports uploading in both the basic free mode and the extended donator mode.



Welcome, PVOutput is a free service for sharing and comparing PV output data.

If you own a solar system please contribute your power output readings.

[Home](#) | [Latest Outputs](#) | [PV Ladder](#) | [PV Donut](#) | [Daily Outputs](#) | [Live Outputs](#) | [Teams](#) | [About](#) | [Register](#)



But you don't need to have solar generation to use the service. You can use it to upload consumption and voltage, as well as six more data sets in donator mode. So let's look at how to set this up with IoTaWatt.

First, you should review the local time offset specified in your IoTaWatt. IoTaWatt synchronizes with PVoutput using local time, including daylight time where applicable. In the Setup/Device display of IoTaWatt, set the time offset for your local standard time, then if your locale observes daylight time (or summer time) check the Allow Daylight Time box.

10.1 Create a PVoutput Account

From your browser, follow the register link on the PVoutput.org login page or just click [here](#). Enter a login ID, password, your email to setup an account, then go to the settings page. Go through the items and select what is appropriate for your locale in terms of date/time format, decimal characters etc. Pay particular attention to the timezone setting. Your account must match the local time setting of your IoTaWatt. Click Save at the bottom and your account is setup. Note that this page is where you will find your API Key, which is one of the few things you will need to specify in the IoTaWatt setup.

10.2 Add Your System

Now click on the Add System link under Registered Systems at the bottom of the page. You really don't need to enter much here to get started. The System Name should be something meaningful to you, but if you want to allow your PV data to be public, you might consider using a something that is not personally identifying. If you have no solar, you can check the Energy Consumption Only box. If you have solar and you fill in the details about your setup, PVoutput will report your efficiency and be able to compare it to similar systems.

Under Live Settings, set the interval to 5 minutes and double check that the timezone matches what you have set in IoTaWatt. Click Save. Note the System ID. IoTaWatt will need to know that.

10.3 Configure IoTaWatt

It's time to configure IoTaWatt to upload to the new account. From within the IoTaWatt configuration app, click Setup/Web Server and then select PVoutput from the dropdown menu.

Add PVOutput Service

API Key:

See <https://pvoutput.org/account.jsp>

System ID:

Please match the requested format: System ID number - all digits.
See <https://pvoutput.org/addsystem.jsp>

upload history from:

Reload History ☐ ⓘ

Status Outputs

Copy and paste the API Key and System Id from the PVoutput system page. For now, set *upload history from* to a recent date like yesterday. Do not check the *Reload History* box.

All that's left is to specify the data to upload. Under **Status Outputs**, click .

Configure Output

Name:

Units:

C

1	4	5	6	+
7	8	9	-	x
.	0	+/-	÷	
()	abs		

As you can see, it's the standard calculator/script interface. The illustration shows the dropdown menu associated with the name. There are nine data items that can be uploaded. Of the nine, generation, consumption, and voltage are

standard. The six *extended_1(v7)* through *extended_6(v12)* entries are additional data that can be uploaded in donation mode. At a minimum, you will need to configure either generation, consumption or both.

generation Specify this if you have a solar PV system. Select the IoTaWatt input that measures your inverter output.

consumption This is the amount of power you use. There are two general cases for this depending on where the solar power is introduced into your system.

- If the inverter feeds in before the mains breaker, then consumption is simply the value of your mains:


```
(main_1 + main_2)
```

- If the inverter feeds in after the mains breaker, i.e. into a breaker inside your panel, then your consumption is the sum of the mains and the solar:


```
(main_1 + main_2 + solar) max 0
```

voltage PVoutput will record and plot your voltage. Most users will simply use the channel_0 voltage input for this.

extended_1(v7) - extended_6(v12) These are the extended values that you can record when you make a donation to PVoutput. Some of the PVoutput documentation refers to them as extended_1 through extended_6, other places they are called v7 through v12. They are the same.

When all of the outputs are specified, click . The PVoutput service will start and uploading will begin. You can monitor the progress in the PVoutput tab of the Status display.

10.4 Reload History

PVoutput allows reloading of historical data subject to lookback limits and maximum transaction rates. Once you are confident that your configuration is correct and uploading what you want, you can upload whatever historical data may be in your data logs. To do this, select the date that you want to begin from, and click the *Reload History* box. When you press , the reload will begin.

If necessary, the starting date will be adjusted to coincide with the contents of the data log. Up to 14 days of history can be uploaded in free mode while donator mode allows 90 days.

Large history uploads may pause due to hourly transaction limits imposed by PVoutput. The message log will indicate these pauses and when to expect resumption.

When the reload is complete, you must reset the reload history checkbox manually, or the data will reload after every restart.

11.1 Setup Emoncms

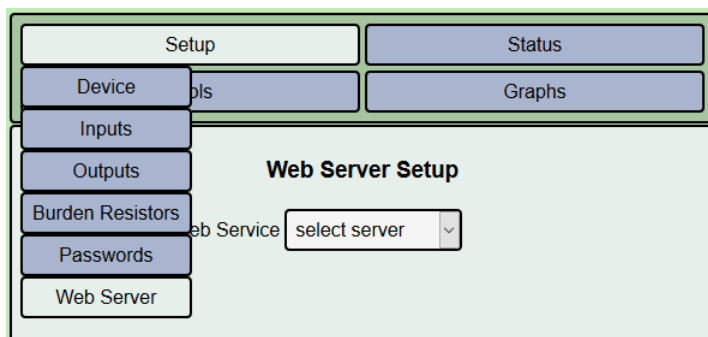
Emoncms is an open-source system that stores time-series data and has multiple reporting apps developed by the authors as well as apps contributed by users in the open community. The graph application used locally by IoTaWatt is a derivative of that effort.

For a nominal fee you can use the managed enterprise version of Emoncms at Emoncms.org or you can install the software on your own machine, including a RPi, and run a local instance for free.

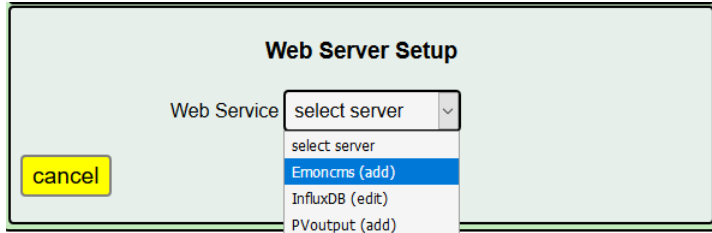
Configuring an IoTaWatt to upload to Emoncms is easy. First, go to Emoncms.org and establish an account, or install the software on your server and setup an account. There is a nominal fee to use Emoncms.org that should amount to a few dollars per month for a typical IoTaWatt user.

11.2 Configure IoTaWatt

After establishing an account, run the IoTaWatt configuration application, hover over **Setup** and click **Web Server** from the dropdown menu.



Choose Emoncms.



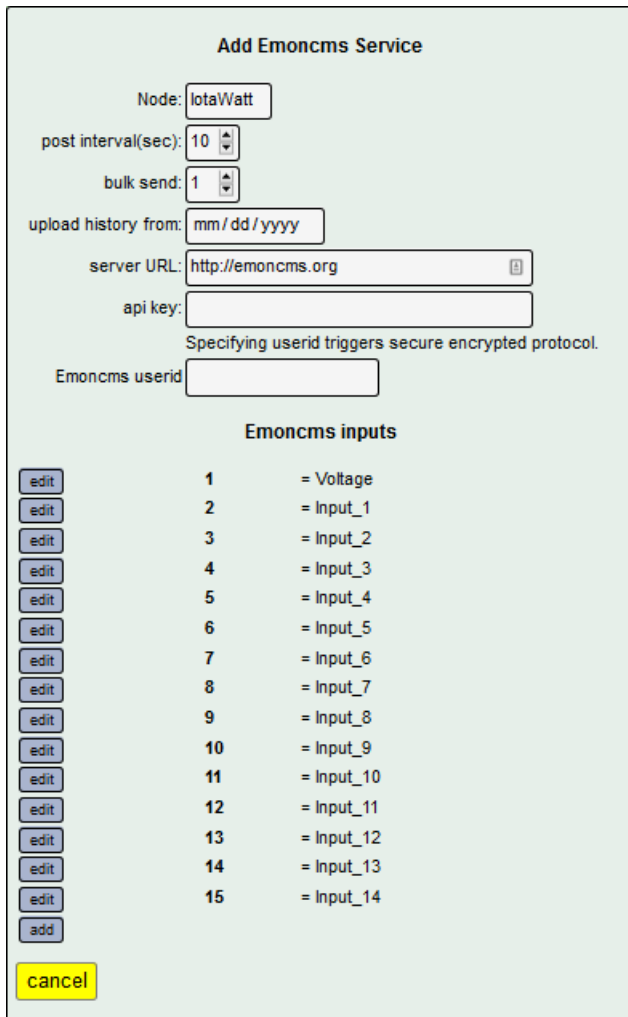
Web Server Setup

Web Service: select server

cancel

select server
Emoncms (add)
InfluxDB (edit)
PVoutput (add)

Here you will specify how IoTaWatt is to upload its data to Emoncms.



Add Emoncms Service

Node:

post interval(sec):

bulk send:

upload history from:

server URL:

api key:

Specifying userid triggers secure encrypted protocol.

Emoncms userid:

Emoncms inputs

<input type="button" value="edit"/>	1	= Voltage
<input type="button" value="edit"/>	2	= Input_1
<input type="button" value="edit"/>	3	= Input_2
<input type="button" value="edit"/>	4	= Input_3
<input type="button" value="edit"/>	5	= Input_4
<input type="button" value="edit"/>	6	= Input_5
<input type="button" value="edit"/>	7	= Input_6
<input type="button" value="edit"/>	8	= Input_7
<input type="button" value="edit"/>	9	= Input_8
<input type="button" value="edit"/>	10	= Input_9
<input type="button" value="edit"/>	11	= Input_10
<input type="button" value="edit"/>	12	= Input_11
<input type="button" value="edit"/>	13	= Input_12
<input type="button" value="edit"/>	14	= Input_13
<input type="button" value="edit"/>	15	= Input_14
<input type="button" value="add"/>		

cancel

Node Grouping that you want to assign to all data upload from this IoTaWatt to distinguish it from data uploaded from any other devices that you may have. It is somewhat arbitrary and defaults to the name of your IoTaWatt device.

post interval Number of seconds that each data point will represent. The tradeoff is between higher resolution (small interval) and minimizing the storage requirement of the data over time (larger interval). IoTaWatt accepts any value from 5 seconds to 3600 seconds (1 hour), in 5 second increments. Because of the way Emoncms reports, it's best to use a number that is an even factor or multiple of one minute: 5, 10, 15, 20, 30, 60, 120, etc.

bulk send Number of interval postings to aggregate into a single posting request. Specifying 1 will send a data packet to Emoncms at each interval. If your interval is 5 or 10, it will send a packet every 5 or 10 seconds. That's fairly inefficient and can be problematic when there are internet connectivity issues. By specifying a larger bulk send, IoTaWatt will aggregate the posting data for that many intervals and send the data in one packet. For instance


if your interval is 10 seconds, specifying bulk send = 3 will cause data to be sent every 30 seconds. Large bulk send values will cause any real-time dashboards in Emoncms to update less frequently, so you should strike a balance. IoTaWatt keeps all of the data in local storage, so there is no risk of losing data by using this feature. Should there be any failure to deliver, IoTaWatt will pick up with the last successful posting when the problem resolves itself.

upload history from When starting a new Emoncms node, you can specify a date from which to upload historical data. To the extent that the data is available in the local datalog, IoTaWatt will bulk upload the historical data to Emoncms. The bulk upload can take anywhere from several minutes to a day or more depending on the time-frame and the upload speed. Regardless, the historical upload is done in the background and will not significantly affect normal operation.

server URL URL of the Emoncms server. If using the Emoncms.org server, the default to the Emoncms.org site, but the software is open, so you may maintain your own server (software on GitHub) or you may buy one of the OpenEnergyMonitor.org products that run a version of the software on a raspberry-pi. When using a local instance of Emoncms, you can use the IP format with optional port number. An example might be 192.168.0.112:80/emoncms.

api key A 32 character hexadecimal key to authorize posting to an account. Once you establish your account, locate the read-write api key and copy/paste into this field.

Emoncms userid By specifying this optional value, you instruct IoTaWatt to use a secure encrypted protocol to send data to Emoncms. There is no downside to doing this, and it is recommended. Your userid is a four or five digit number located in the “My Account” section of the Emoncms.org site.

Now click  and IoTaWatt should begin sending data to your Emoncms account. You can see the status of the Emoncms service in the [Status Display](#)

IoTaWatt uploads the current voltage or power corresponding to all of the input channels in each post. You can configure Emoncms (follow their instructions on the website) to save only what you want to keep in “feeds”. Setting up and configuring the Emoncms account is documented on the Emoncms.org site.

11.3 Customizing Input data

You can customize the data that is sent to Emoncms using the Emoncms Inputs list at the bottom of the configuration screen. This list is very similar to the Configure Outputs section, except rather than calculate named values that can be viewed in the status screen or displayed in the graph application, you specify how to calculate the individual inputs to Emoncms using the same calculator interface.

Initially, these fields correspond to each of the IoTaWatt inputs. By editing this list, you can change the values that are sent, delete specific values, or add additional computed values to be sent.

One distinctive feature of this list vs the outputs list is that the names must be numeric values between 1 and 99. The name of an entry corresponds to the Emoncms input “key” value. When editing the various fields, if you add a new entry with the same number as an existing entry, or change an entry to the same number as an existing entry, it will replace the existing entry. Each time you save an entry, the list will be reordered.

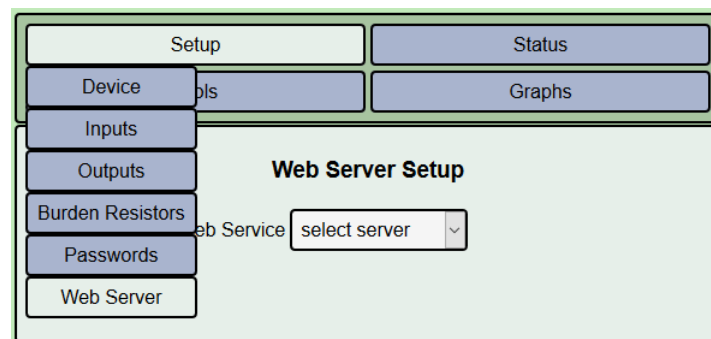
influxDB is a fast open-source time-series database package. It can run on a variety of platforms and is also available as a managed, fully hosted service, from several vendors. As part of their “TICK” stack, InfluxData provides tools to facilitate collecting data from a variety of sources, as well as tools for infrastructure monitoring, alert management, data visualization, and database management. The popular grafana visualization tools also work well with influxDB.

IoTaWatt fully supports the influxDB HTTP API for sending data to influxDB at a specific interval of 5 seconds to one hour. Like other similar IoTaWatt services, continuity of updates is maintained despite outages that may interrupt the communications.

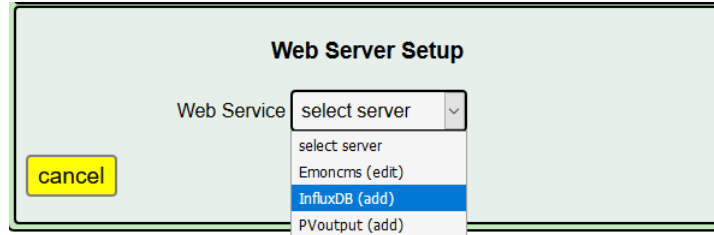
This tutorial assumes you have established your own instance of influxDB or subscribed to a hosted service. It does not attempt to explain how to install or use influx or any of the related visualization tools. There’s a whole universe of enthusiastic users at the influxData forum, where you can get help with anything and everything.

12.1 Configure IoTaWatt

To configure the influxDB upload service in IoTaWatt, Hover over **Setup** and click **Web Server** from the dropdown menu.

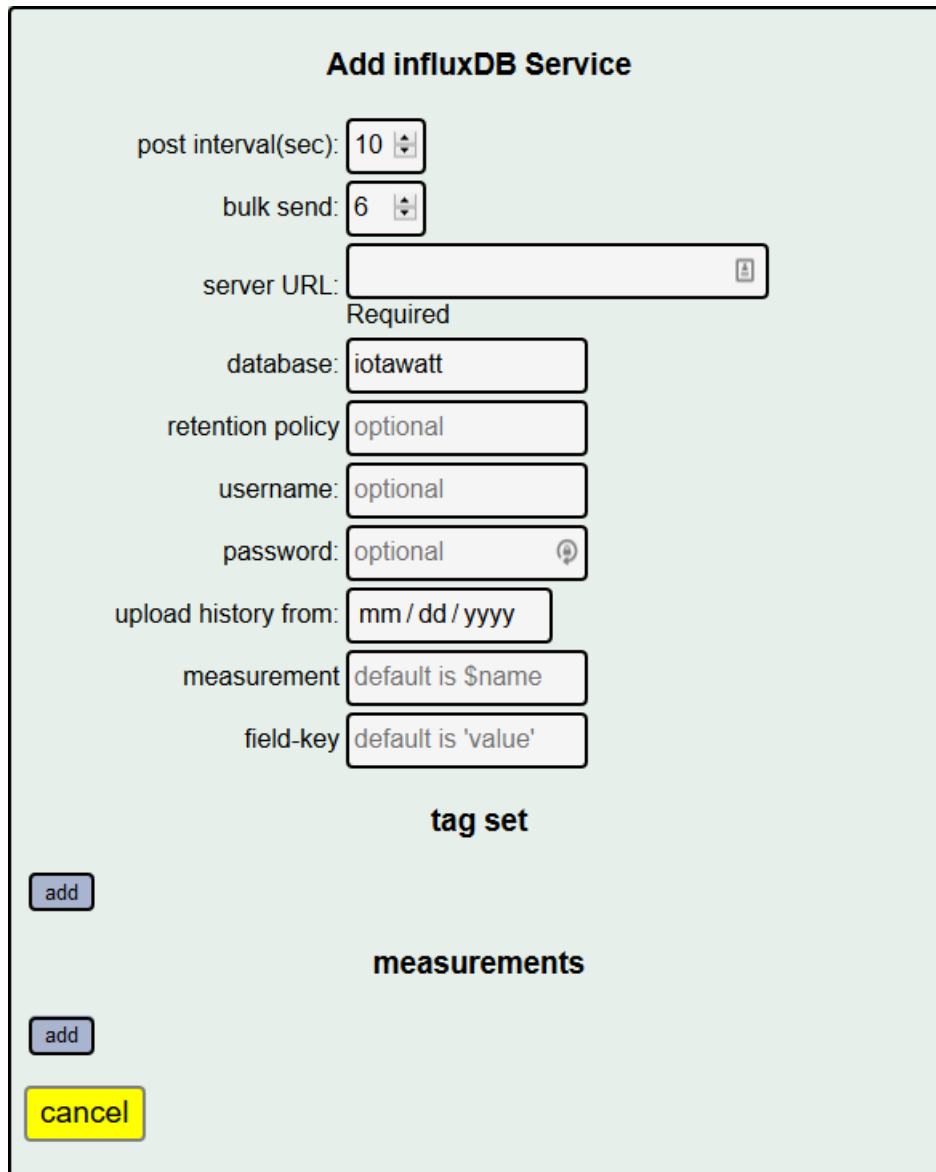


Choose influxDB.



The 'Web Server Setup' dialog box contains a 'Web Service' dropdown menu. The dropdown is open, showing options: 'select server', 'select server', 'Emoncms (edit)', 'InfluxDB (add)' (highlighted in blue), and 'PVoutput (add)'. A yellow 'cancel' button is located to the left of the dropdown.

Here you specify what you want IoTaWatt to upload.



The 'Add influxDB Service' dialog box contains the following fields and controls:

- post interval(sec):** A numeric input field with a value of 10 and up/down arrows.
- bulk send:** A numeric input field with a value of 6 and up/down arrows.
- server URL:** A text input field with a placeholder icon. Below it is the text 'Required'.
- database:** A text input field with the value 'iotawatt'.
- retention policy:** A text input field with the value 'optional'.
- username:** A text input field with the value 'optional'.
- password:** A text input field with the value 'optional' and a password icon.
- upload history from:** A text input field with the value 'mm / dd / yyyy'.
- measurement:** A text input field with the value 'default is \$name'.
- field-key:** A text input field with the value 'default is 'value''.

Below these fields is a section titled 'tag set' with an 'add' button. Below that is a section titled 'measurements' with an 'add' button. At the bottom right is a yellow 'cancel' button.

post interval Number of seconds that each data point will represent. The trade-off is between higher resolution (small interval) and minimizing the storage requirement of the data over time (larger interval). IoTaWatt accepts any value from 5 seconds to 3600 seconds (1 hour), in 5 second increments. Although not a strict requirement, it's best to use a number that is an even factor or multiple of one minute: 5, 10, 15, 20, 30, 60, 120, etc.

bulk send Number of interval postings to aggregate into a single HTTP request. Specifying 1 will send a data packet to influxDB at each interval. If your interval is 5 or 10, it will send a packet every 5 or 10 seconds. That's fairly

inefficient and can be problematic when there are internet connectivity issues. By specifying a larger bulk send, IoTaWatt will aggregate the posting data for the specified number of intervals and send the data in one packet. For instance if your interval is 10 seconds, specifying bulk send = 6 will cause data to be sent every 60 seconds. Large bulk send values will cause any real-time dashboards to update less frequently, so you should strike a balance. IoTaWatt keeps all of the data in the SD data log, so there is no risk of losing data by using this feature. Should there be any failure to deliver, IoTaWatt will pick up with the last successful posting when the problem is resolved.

server URL URL of the influxDB server. The URL must begin with <http://> (not <https://>). The url may contain a domain name or an IP address. The *:port number* is optional and defaults to the influxDB default of :8086.

database Name of the influxDB database that you have created to be the repository for the IoTaWatt data.

retention policy Optional name of the influxDB retention policy that you want to associate with the measurements that are written to influxDB. If not specified influx will use the default policy. If you specify a retention policy, it must be defined to influxDB before data can be written.

username/password Optional security credentials. If specified, IoTaWatt will use standard authorization headers with these credentials.

upload history from Specify the starting date that IoTaWatt will use to upload history to a new set of measurements. When the influxDBService starts, a query is made to determine the date/time of the last data written, qualified by the first tag-set. Data upload is initiated as follows:

Condition	begin date specified	begin date not specified
new measurement set	begin date 00:00	current date/time
existing measurement set	greater of last entry date/time or begin date	last entry date/time

measurement Name that you assign to the measurements that IoTaWatt will be posting. The specification can be a constant string, or can include variables as explained below under variables. Note that if not specified, the variable \$name will be used.

12.2 tag-set

tag-set A collection of optional user specified tag-key/tag-value pairs that will be included as part of each measurement. The influx documentation somewhat explains them here. Basically, these tags each produce a table index that can be helpful in increasing the performance of data retrieval. The first tag-set is a special case for IoTaWatt, and if specified, is used to uniquely identify the measurement subset from this device so that upload can resume seamlessly. If this is not the only device that will be posting to the database, a unique identifier for this device should be included as the first entry in a tag-set. tag-values can be a constant string or can include variables as explained below under variables

edit an existing tag by clicking on it's associated edit button, or add a new one with the add button. add tag-set

Edit influx tag

tag key

=

tag value

Tags are optional key-value pairs of metadata that can be associated with the measurement. Because they are indexed, queries on tags perform better than queries on fields.

When tags are specified, IoTaWatt assumes the first tag is unique to this measurement and is referenced to determine the time of last measurement during restart recovery. In that way, an IoTaWatt can maintain context and continuity where multiple data sources are writing to a single database.

cancel

field-key Each measurement contains a field key and field value. The field value is always the value specified in the “calculator” function. The field key to be used is specified here. It can be a constant string or can include variables as explained below under variables. If not specified, the default field key in each measurement is the string ‘value’.

12.3 measurements

measurements The set of *measurement*s that will be sent at each interval. The editor uses the “calculator” interface to create scripts to generate data using the various IoTaWatt inputs. Each entry will generate a measurement using the influx line protocol:

```
<measurement>[,tag-key1=tag-value1[,tag-key2=tag-value2...]] field-key=field-
↪value time
```

Note that **measurement**, **tag-value**, and **field-key** are specified above and can be fixed strings or can be or contain variables. Some examples of the various ways this can be used to create different types of measurement specifications are given below.

As each measurement is written the *\$name* and *\$units* variables are assigned the value specified in the individual measurement specification.

The units of the field set defaults to watts. While Watts is the typical unit reported, the following additional units are available.

- watts
- volts
- VA
- pf
- Hz
- kW
- wH

- kWh

Once configured, a new influx service will be created. The current state of the service and the date/time of last update will be displayed under the influxDB tab in the status display.

influx status bar

The service can be started and stopped using the start/stop button. When a running influx service is changed, the service is automatically stopped and restarted.

12.4 Variables

Variables provide a way to further customize the way data is organized in your influxDB database. This tutorial will not get into the implications of different conventions, except to say that future generic visualization templates may be based on using the default specifications for measurement and field key.

There are three variables defined:

- **\$device** - The name assigned to this IoTaWatt device in the device configuration section.
- **\$name** - The name specified for the current measurement
- **\$units** - The units specified for the current measurement

When these variables appear as all or part of the string specified for measurement field key or a tag value, the instance of the variable name is replaced by it's value. Evaluation proceeds left to right in a string.

So as an example, when generating a measurement configured as:

with a device name of IotaHome and the current value of the input solar of 2944.6, the following different measurements could be generated:

measurement	tag-value	field-key	measurement sent to influxDB
\$name		value	solar value=2944.6 1523810195 (This is the default)
\$units	\$device	\$name	kWh,tag1=IotaHome solar=2944.6 1523810195
\$device	\$name	\$units	IotaHome,tag1=solar kWh=2944.6 1523810195
\$name.\$units		value	solar.kWh value=2944.6 1523810195
power	\$device.01	\$name	power,tag1=IotaHome.01 solar=2944.6 1523810195

Three-phase Power

IoTaWatt has the capability to measure power in polyphase systems. In the interest of keeping the user interface simple for the majority of users with single phase power, the details are disguised and/or hidden until needed. There has been a lot of interest in using IoTaWatt for three-phase monitoring, particularly in three phase countries like Australia and Germany. For home energy monitoring, the *derived reference* method is very popular.

This chapter explains two methods for measuring a so called “four wire” or “wye” system, by far the most common implementation of three-phase. IoTaWatt can also be used with “three wire” or “delta” systems using different methods.

As explained in the introductory single phase section, a voltage reference is needed to measure real power. The challenge with three-phase power is to obtain three different voltage/phase reference signals. IoTaWatt can do this by two different methods:

Direct Reference uses three discrete voltage transformers or VTs, each one plugged into a circuit on a different phase. The primary advantage is voltage accuracy, while the disadvantages are that the extra VTs add cost, require using two of the inputs, and require plugs on each phase in proximity to the IoTaWatt.

Derived Reference uses a single voltage transformer and *derives* a reference voltage/phase for the other two phases by numerically shifting the phase of the single reference by 120° or 240°. The primary advantage is low cost and convenience of installation. The disadvantage is that large variations in voltage between phases may result in decreased accuracy.

13.1 Configuring Direct Reference

13.1.1 Connecting additional VTs

To use Direct Reference three-phase power measurement, it's necessary to install two additional VTs (total of three), and to plug each of them into a receptacle that is supplied by a unique phase.

Version 5 of IoTaWatt, available second quarter 2019, will have native plugs to connect the additional VTs. This tutorial will assume you have the new version 5 IoTaWatt.

The additional VTs will plug into two sockets at the rear of the unit. They are labelled VT-13 and VT-14. When these are used, the standard channel 13 and 14 jacks should not be used.



13.1.2 Configuring the voltage inputs

Now the additional VTs can be configured and calibrated. Do this in the same way that the first VT was configured. Click the channel number, click “VT” then specify the model. Click Calibrate and calibrate the voltage to match your reference. It’s not necessary that the VTs be plugged into their eventual phase for this step. If you have two outlets on any of the phases, use those to plug in each VT in turn along with a voltage reference while you calibrate. Once calibrated, the VTs can be moved to the appropriate phase/socket.

Inputs

0	phase_a	VT, TDC DE-10-09(EU), cal:18.66, lead:1.55°
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13	phase_b	VT, TDC DE-10-09(EU), cal:18.66, lead:1.55°
14	phase_c	VT, TDC DE-10-09(EU), cal:18.66, lead:1.55°

☐ Enable derived three-phase

Name each of the phases to uniquely identify each reference. You can use phase_A, phase_B etc., or maybe use the color coding of your system to be more descriptive - voltage_red, voltage_black, voltage_blue (US).

13.1.3 Configuring the CTs

Now start adding your CTs. The twist here is that because more than one VT is configured, an additional selection box is displayed to specify which VT is associated with the phase of that particular CT.

Configure Input 3

Burden: 20 ohms

Name:

Type:

Model:

VRef:

negative power value ☐

le ☐

le ☐

☐ Reverse ☐

If your service is consistently color coded, you should know the phase by the color of the conductor that you clamp the CT onto.

13.2 Configuring Derived Reference

Another way to approach the voltage reference problem in three-phase is to use the voltage/phase reference of one phase to derive a reference signal for the other two. While not as exact, this method can produce good results. The IoTaWatt numerically shifts the single voltage/phase reference by 120° or 240° to measure power on the additional two legs. Using this method, a three-phase system can be monitored with a single VT, just as with single-phase systems.

This chapter explains how to configure IoTaWatt to use “Derived Reference”. It’s pretty straightforward.

13.2.1 Configure the VT

Set up your IoTaWatt with the voltage reference VT on whatever phase of the three-phase is convenient. We will call that phase A. This is important:

However you identify your physical phases, IoTaWatt always identifies the phase that the VT is connected to as phase A.

The other two virtual phases will be phase B and phase C. It doesn’t matter where which of the physical phases your VT is connected to. Your starting point is always A.

There are color coding schemes for the phases, but they vary so widely that I’m not going to try to reconcile this scheme with any of them. That exercise is left to the reader. The good news is that you really don’t have to know what any of the phases are to complete this setup.

13.2.2 Configure the CTs

Connect CTs to each of the circuits that you want to measure, and configure them as described here.

Be sure to orient all of the CTs the same way with respect to source and load.

The following instructions will not work as described if any CTs on the derived phases are reversed.

If you've done everything correctly, your IoTaWatt status display should be displaying the correct power for all of the circuits on phase A, and roughly half power for all of the circuits on phases B and C.

Now in the input configuration menu, click the box for "Enable derived three-phase" at the bottom.

Inputs

0	voltage	VT, Ideal 77DE-06-09(EU), cal:19.26, lead:1.78°, phase:A
1	main_A	CT, SCT013-000, cal:83.33, lead:3.00°, phase:A
2	main_B	CT, SCT013-000, cal:83.33, lead:3.00°, phase:A
3	main_C	CT, SCT013-000, cal:83.33, lead:3.00°, phase:A
4	circuit_1	CT, CR3110-3000, cal:125.70, lead:1.85°, phase:A
5	circuit_2	CT, HWCT-004, cal:41.83, lead:0.35°, phase:A
6	circuit_3	CT, SCT006-000, cal:33.33, lead:3.80°, phase:A
7		
8		
9		
10		
11		
12		
13		
14		

☒ Enable derived three-phase

The configured input channels should now have "phase:A" appended to their descriptions. Go to the status display and evaluate which of the inputs appear to be showing power that is half what is expected. Note them and go back to the input configuration screen. If you know the relative phase of your circuits, you can just specify them now and fast-forward over this "trial and error" approach that follows.

Edit each of the incorrect inputs in turn, changing the "Mains Phase" to phase B.

Now go back to the status display and see which inputs still appear to be about half of the expected value, go back to the input menu and change those to phase C.

The status display should now indicate the correct power for all of the phase.

This procedure works best when the loads are substantial and have high power factors.

One additional point. Once you configure inputs to indicate mains phase B or C, the “Enable derived three-phase” checkbox will remain set and cannot be turned off until all of the inputs are reconfigured back to phase A.

13.3 Reporting Power

Once all of the VTs and CTs are configured, there are several ways to view the power used. For circuits and/or loads that use only one phase, the power value displayed for that channel should be correct as is. If there are devices that use two or three of the phases, you must add the power from each of the phases to get total power. For each such device, define an output channel and use the calculator to specify adding the component channels. If you are reporting the data to a server, the data can be tailored with the calculator to send the single combined aggregate power for those devices.

14.1 What is a Current Transformer?

From Wikipedia:

A current transformer (CT) is a type of transformer that is used to measure alternating current (AC). It produces a current in its secondary which is proportional to the current in its primary.

In our case, the primary is *one* of the conductors of the circuit that we want to measure, and the secondary is the output jack that plugs into the IoTaWatt.

So without connecting anything directly to any high-voltage wiring, it's possible to get a scaled down measure of the primary current that can be used to passively measure power (Watts) of a circuit.

14.2 Types of CTs

CTs come in various types, sizes, and capacities, and are made for a variety of end uses. This tutorial doesn't try to address all aspects. That's what Wikipedia does well. Here we'll try to focus on the CTs that are suited to use with the IoTaWatt in typical scenarios.

Physically, a CT needs to have an iron core through which one or more primary conductors pass. The most basic type is a solid-core CT, where an iron doughnut is wrapped with turns of wire. This type of CT is relatively inexpensive, typically very accurate, but requires that the primary conductor be disconnected and reconnected to install, thus exposing the installer to high-voltage and disrupting the primary circuit.

Split-core CTs also require an iron core around the primary, but do so using two hinged halves that mate to form the continuous iron loop. This type of CT can be installed by simply snapping the two halves over an active primary conductor.

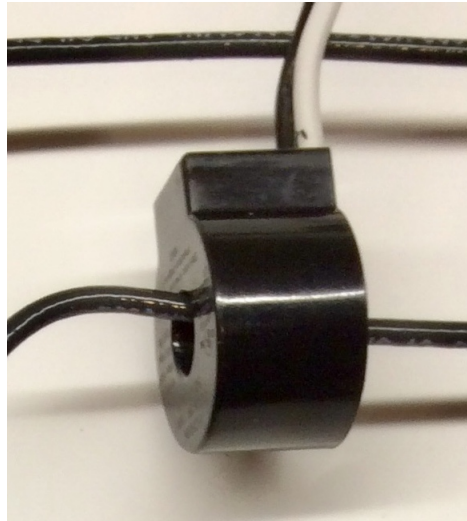


Fig. 1: Solid Core CT

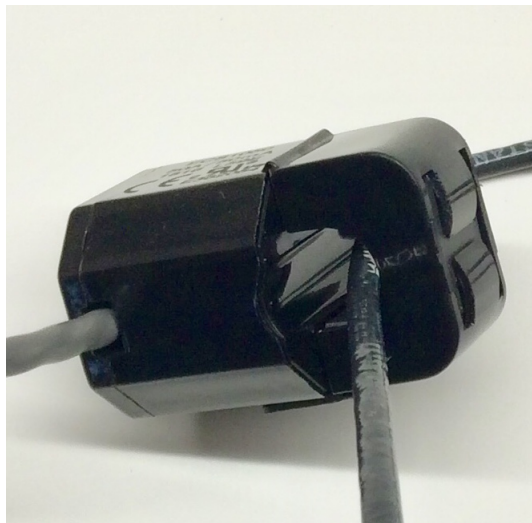


Fig. 2: Split Core CT

14.3 Installation



The installation of CTs can be dangerous and/or cause hazardous situations resulting serious injury or death. World-wide, there are a variety of electrical conventions, regulations and standards. It is the user's responsibility to insure that the installer is qualified and all local codes and regulations are followed.

To measure the current in a circuit, a CT is installed on one, and only one, of the conductors in a circuit. Either the conductor is passed through the solid-core, or the split-core is clamped over it.

CTs must have a load. Without a load, they will develop very high voltages that can damage the core windings and/or create a safety hazard. When plugged into the IoTaWatt, the secondary windings are loaded by a *burden resistor*.

Some CTs have protective diodes, called TVS diodes, that will protect against damage when unplugged for short periods. Even if a diode protected CT is to be unplugged for an extended length of time where the primary is energized, the CT should be removed or shorted. Shorting will not damage the CT.

14.4 Polarity

CTs are manufactured to produce a secondary current that is in phase with the primary current when installed with a particular orientation. In single and split-phase installations it is important to observe polarity in certain situations. In three-phase installations, it is imperative that a polarity convention be observed.

IoTaWatt will accept many different CTs from different manufacturers. While most have some type of markings that can be used as a reference for polarity, there is no universal standard. Typically, CTs from the same manufacturer will be consistent with respect to source and load indicators.

And so it is for the Echun CTs that IoTaWatt, Inc makes available. When installing, we use the notion of a *source* and a *load*. The source can be conceptualized as *where the power comes from* and the load as *where the power goes*.

So for the mains, or incoming power to a service, the *source* would be the meter side, or incoming power feed. The *load* would be the main circuit breaker or fuse side.

For branch circuits, it would be just the opposite. The *source* would be the circuit-breaker side, and the *load* would be the appliance side.

For a solar inverter connection, the *source* would be the inverter side, and the *load* would be the circuit-breaker or other point of interconnect.

14.5 Single and three-phase systems

All of the CTs in single or three-phase systems should be installed identically with respect to load and source. This is especially important when configuring three-phase systems using the [Derived Three-phase](#) method.

14.6 Split-phase systems

Most of North America and some Asian countries use a split-phase power system with dual voltage, typically 120/240V. With this power system, there are two mains with exact opposite phase. The voltage between either main



Fig. 3: Here you can plainly see “This side toward source”



Fig. 4: Here the arrow points source(K)->load(L)



Fig. 5: This is an Echun ECS25200 clamp type CT used for 200A mains. Both sides are shown. Note the arrows just under the opening. The arrow pointing up to the opening indicates the source side, and the down arrow indicates the load side.

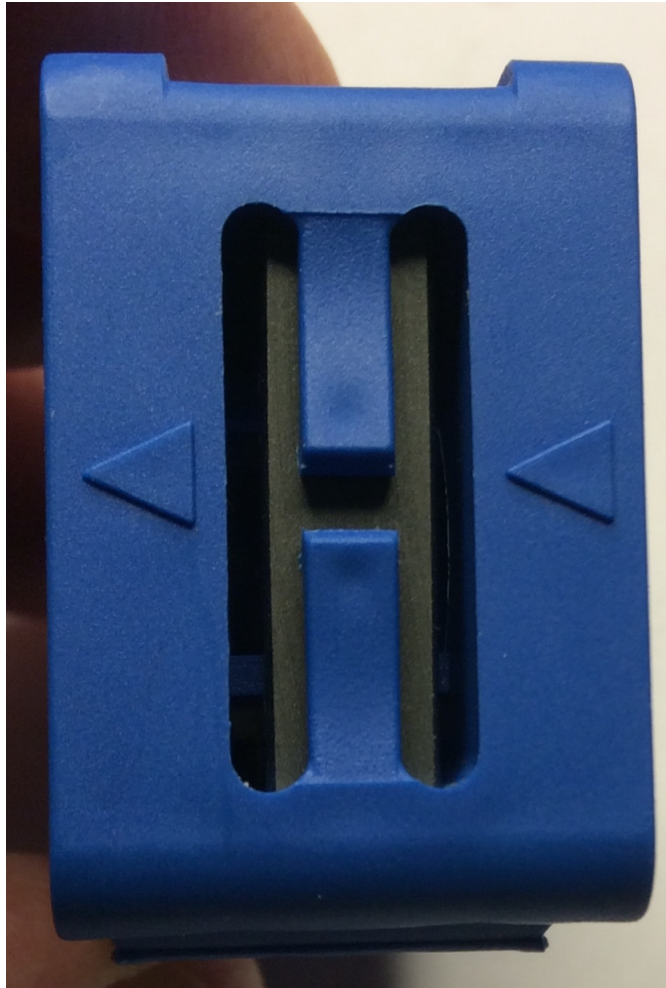


Fig. 6: This is the common SCT013 CT. If you are using them exclusively, the arrow can be aligned consistently as source to load. But note that if using with the Echun CTs, they must be installed with the arrow pointing from load to source. This isn't a fault of either manufacturer. It just reflects the lack of a standard for how to connect the CT secondary to the 3.5mm jack used to connect.

and neutral is 120V, while the voltage between the two mains is 240V. This provides an advantage of the relative safety of lower voltage in small appliance outlets, while still providing high voltage for workhorse appliances like water-heaters, ranges, and clothes dryers.

In these systems, while possible to use two voltage references, typical IoTaWatt installations use a single reference that reflects the phase and voltage of one of the sides, or *legs* as they are commonly called. The result is that CTs on the other leg must be oriented the opposite way to be in phase with the opposite voltage reference. This can be accomplished by physically installing them reversed, or by installing all of the CTs the same way and checking the *reverse* box when configuring.

There is more to installing CTs on 240V circuits in split-phase systems in the next chapter.

14.7 240V Split-phase circuits

As explained above, split-phase systems can provide high-voltage for large appliances. These circuits are connected to two adjacent CTs that are on different legs. The usual convention is to use RED and BLACK wires or, as explained below, BLACK and WHITE for *pure* 240V circuits.

14.7.1 240V only

When I say *pure* 240V circuits, I mean circuits that are usually a single load, and do not have a third neutral wire to use either leg independently for 120V. Examples of *pure* 240V circuits would be a resistive water-heater, well-pump, and baseboard electric heater. A common giveaway for these circuits is that they don't have a neutral wire, and usually use two conductor with ground BLACK and WHITE leads.

With these circuits, you can place the CT on just one of the conductors, and check the *double* box in input configuration, directing IoTaWatt to double the voltage value to report correct power and amperage.

14.7.2 120/240V circuits

Like the *pure* 240V circuits above, these circuits use two adjacent circuit-breakers, but also have a neutral conductor. They usually have RED and BLACK conductors on the circuit-breaker and a white neutral conductor that connects to the neutral bussbar. Typical appliances are ranges, ovens, and clothes-dryers. Circuits feeding sub-panels are usually of this type as well.

For these circuits, the two legs must be measured individually because the current in each is not always the same. There are a couple of ways to do this.

The easiest way is to pass both the RED and BLACK conductors through the CT. A CT will measure the total current of all of the conductors that pass through the primary. But there is a twist. The phase of the current in each is exactly opposite the other, so they they will cancel each other out and rather than get the sum of the two, you can get the difference between the two.

The solution is to pass one conductor through in the opposite direction to the other. There is a common trick for this. In most panels, the conductors are brought past the CT in a U shape so that there is some excess wire in case the circuit needs to be moved within the panel. You can use this U configuration to easily reverse one of the conductors. In this case, the CT needs to handle the combined capacity of the two circuit breakers when added together. An ECS1050 can probably be used up to about a 2x30A breaker.

An alternate method, and recommended with high amperage sub-panel circuits, is to put a separate CT on each leg. The CTs can be connected to two individual IoTaWatt inputs and added together later for the total. With this method, each of the two CTs only need match the capacity of one of the circuit breakers.

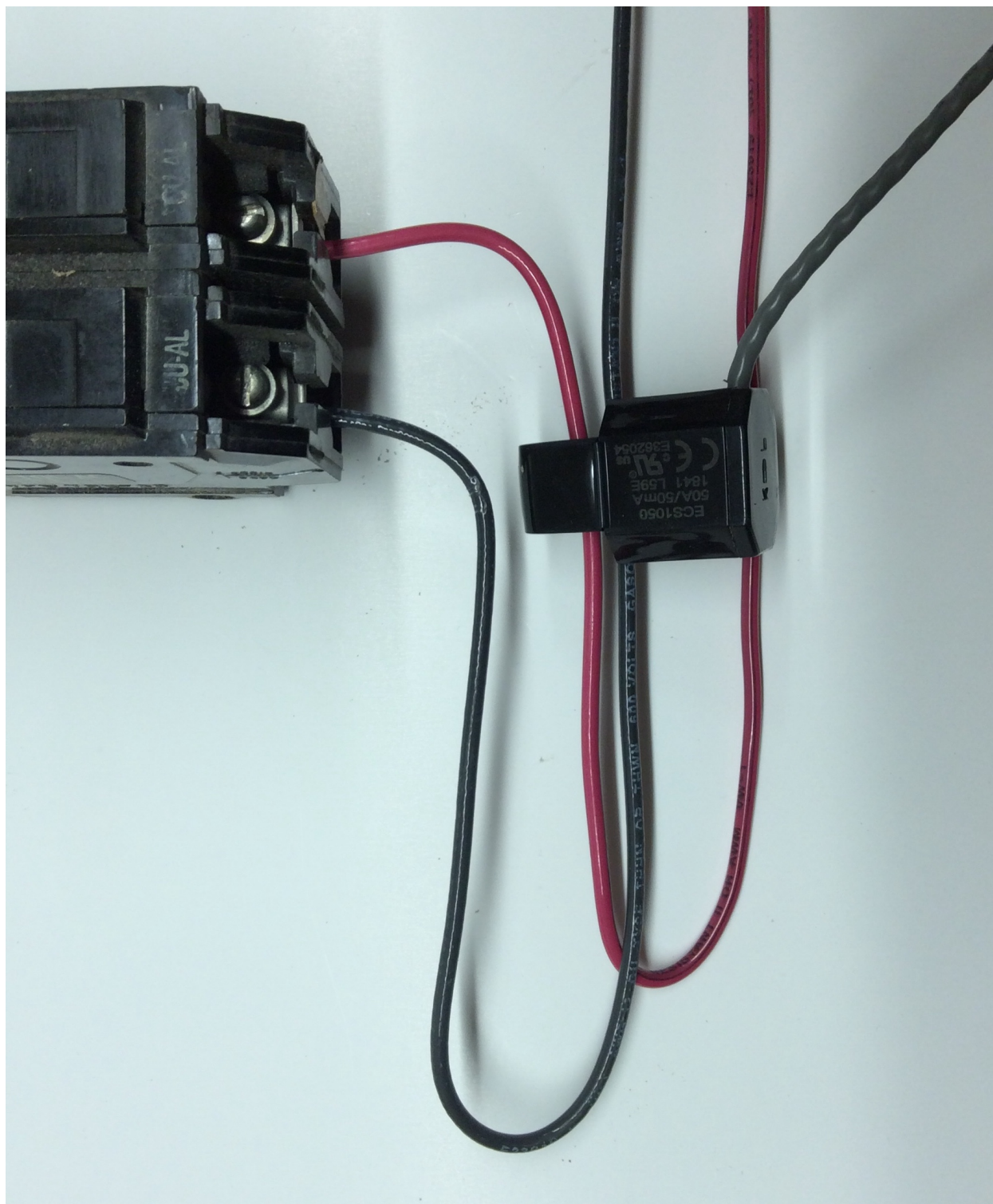


Fig. 7: The CT is clamped around the RED wire going down and the BLACK wire going up.

Two individual CTs can also be combined with a common headphone splitter and fed into a single IotaWatt input. When combining this way, both CTs must be the same model with an individual capacity sufficient to measure the combined capacity of the two circuit breakers.

CHAPTER 15

Split-Phase Installation

This chapter assumes the reader has read the [CT Basics](#) section.

15.1 What is split-phase?

Worldwide, virtually all of the residential power delivered to residential homes is 230V-240V single phase. In North America, the single-phase 240V supply is split so that there are two 120V legs that are typically used for lighting and light-duty appliances. To learn more see the [Split-Phase Wiki](#).

15.2 Split-phase load centers

A typical split-phase load center has two Main circuit breakers, one for each of the two “legs” coming from the service entrance. There is also a third neutral wire that is directly connected to the neutral bus - a long bar with holes and screws to connect conductors. Another ground bus is provided which is connected to a reliable local earth connection like a ground stake. In most entrance panels, but not sub-panels, the neutral and ground busses are “bonded” (connected together).

The service is typically described in terms of the amperage rating of the Mains circuit breakers or fuses. Most common are “100A service” and “200A service”.

See the North America section of this Load-Center [Wiki](#). to see how the split-phases correspond to alternating breaker rows.

15.3 Monitoring split phase

When monitoring a circuit in a split phase service, it’s helpful to recognize exactly which of the mains circuits it is utilizing. There are three possibilities:



Fig. 1: Maybe neater than normal load center

15.3.1 120V phase A

These circuits are two conductor (plus ground) and typically use a black conductor connected to the circuit breaker and a white conductor connected to the neutral bus. As described in the [Wiki](#), circuit breakers in odd rows will be on phase A. Odd rows include breakers numbered [1,2] [5,6] [9,10]...etc.

15.3.2 120V phase B

Same as above, except circuit breakers will be in the even rows with breakers numbered [3,4] [7,8] [11,12]... etc.

15.3.3 240V

These circuits are typically larger appliances like Hot-Water, Range, Pumps, Electric Heat, Dryers, Heat-pumps, sub-panels, and Air-conditioners. They use two circuit breakers in adjacent rows, so one of the breakers is on phase A and the other is on phase B.

As explained in the [CT Basics](#) section, 240V loads can be two-wire or three-wire.

Two-wire loads typically use two conductor cable with the white and black conductors connected to the two circuit-breakers. For these circuits, a single CT can be placed on either one of the two conductors and the “double” box checked to indicate to IoTaWatt that the voltage is doubled to 240V. Orientation of the CT is dependent on the row that the CT is associated with.

Three-wire loads typically use three conductor cable with black, red, and white conductors. The red and black are connected to the two breakers and the white is connected to neutral. These loads must be measured with two CTs, one on the red and one on the black, or by passing the two conductors through one CT in opposite directions so that the resulting orientation of the CT is correct for the row of the breaker that each conductor is connected to. An illustration can be found in the [CT Basics](#) section.

15.4 Voltage Reference

The voltage between the two mains conductors is nominally 240V. That is broken down into two 120V potentials between each of the Mains and the neutral, but they unless they are in perfect balance, there will always be a difference in voltage between the two phases. That said, the difference in most situations is practically negligible and the sum of the two will always equal the voltage between the two Mains.

This is an important point, because in order to measure real-power, what your meter measures and you pay for, it's necessary to also have a reference voltage on that circuit.

IoTaWatt supports multiple voltage references, but as a practical matter, you only need one to measure any of the three circuits described above.


The simplest and most effective reference is a 120V wall transformer connected to an ordinary 120V plug as close to the load center as possible. That will provide a reference for the phase it is plugged into, *and a reference that is the exact opposite of the other phase*, or to put it another way, the *reverse* of the other phase.

15.5 Mains CT orientation

The first CTs to be installed should be the Mains.

As mentioned in multiple discussions, this is a job for someone familiar with the working with live, partially exposed electrical wiring and familiar with all of the risk factors. An electrician is recommended.



Because each of the Mains is on a different phase, and we are using the same voltage reference for both phases, we need to reverse one of them so that the current it measures aligns with the reversed voltage reference. So which one? The answer is that because the wall transformer plug is not polarized as well as other uncontrolled factors, we don't know yet.


So what we do is install just install the CTs with opposite orientation, [configure them](#), and look at the status display of the IoTaWatt. If reversed, they will show a  symbol. To correct this, you can do one of two things.


- Reverse the wall transformer in it's socket.
- Click the “reverse” box in the VT configuration menu.

Now the two Mains inputs should show the Watts for each Main and no reverse .

15.6 Load CT orientation

First, it's important to note that the only consequence of installing a load CT backward is that it will show a  symbol next to the input in the status display. This indicates that IoTaWatt has recognized that the voltage and current are opposite and is producing the correct measurement by reversing the output numerically. There is no error attributable to this correction. If the  symbol doesn't bother you, you can place the CTs without regard for phase.

Another approach is to simply install the CTs without regard for phase orientation and then simply check the “reverse” box for any inputs that show the  symbol in the status display or physically reverse those CTs in the load center.

To install with correct orientation initially, the easiest method is to install one CT on an active circuit and note if the  symbol appears in the status display. If so, reverse that CT. Now note which way the correctly oriented CT is installed and whether it's row is even or odd.

If it's an even row, all of the CTs that you install on even row circuit-breakers should be installed with the same orientation and the odd row circuit breakers with the opposite orientation. And visa-versa.

CHAPTER 16

Data Visualization

IoTaWatt is capable of storing ten years or more of high resolution data. An integrated web-server provides access to that data using a versatile query/graph application called Graph+ and also provides a RESTful data query facility capable of producing JSON or CSV formatted downloads.

You can also upload your data to InfluxDB, PVoutput or Emoncms.org providing a variety of alternative ways to organize and view your voltage, power and energy use. The servers can be fast and accessible from any place where there is internet connectivity. But there are some circumstances under which you may prefer to access your data directly from the IoTaWatt:

- You prefer the simplicity and power of the local graphic viewer.
- You are not using Emoncms or influxDB to save your data.
- You require finer resolution than is used in your Emoncms feeds.
- You need to see data that is collected by IoTaWatt but is not being uploaded to your cloud service.

There are two graphing packages available with the current release:

16.1 Graph+

This latest graphic viewer unlocks virtually all of the data in the datalog. You can graph all of the metrics associated with each input or output, including Volts, Watts, Wh, Amps, VA, PF and Hz.

There are selectable predefined time-periods like “Today”, “Yesterday”, “Last Week” and “Last Month”, and a calendar interface that can be used to select custom date/time bounds. Data can be grouped by hour, day, week or month. The IoTaWatt query recognizes week and month boundaries and daylight-time changes.

16.2 Original Graph

The graph program provided in the initial releases through 02_03_02 is still supported and available for those who may have become comfortable with it and whose capabilities are adequate for your needs. It remains as a menu pick

under Data tab.

The new Graphic Viewer is a superset of the functions of the Original Graph, so this feature should be considered deprecated and may be removed in a future release.

CHAPTER 17

Graph+

Available with release 02_05_00.

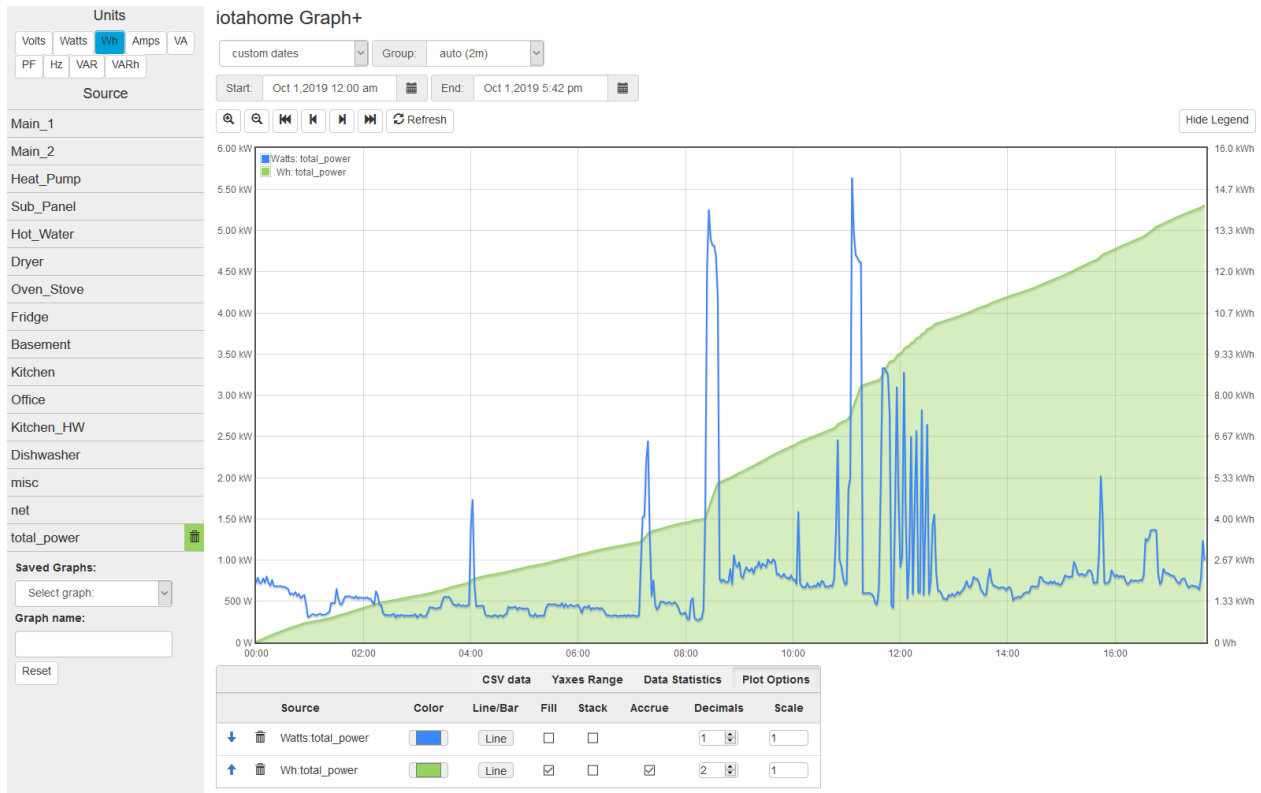
This data visualization application runs in any browser to graphically present any of the data stored in the IoTaWatt. It features relative and absolute time period selection and handles all units of measure supported by IoTaWatt. You can produce line and/or bar charts with optional stacking and fill. Real-time graphs can be set to automatically refresh. Once created, any graph specification can be saved for quick future use.

The application is fully adaptive so it will work well on a full range of devices from mobile to desktop.

The window can be broken down into four parts, each serving a different function:

- *Unit/Source selector*
- *Time period selector*
- *Graph window*
- *Trace tables and options*

Below is a complete window with all of the parts. We're plotting power for the current day (midnight to 5:42pm). Along with that we've plotted the accrued Watt-hours for the day, 14.2 kWh as of 5:42.



17.1 Unit/Source selector

So now let's break it down. On the left is the “sidebar” where you select each of the measurements that you want to plot. Every input (or output) can produce a variety of measurements. VTs measure Voltage and frequency(Hz), and CTs measure Watts, Watt-hours, Amps, VA, and Power-Factor (PF). So the first thing is to select the measurement you are interested in. At the top are the unit selection buttons. Click on the particular unit you are interested in.

Units

Volts

Watts

Wh

Amps

VA

PF

Hz

VAR

VARh

Source

Main_1

Main_2

Heat_Pump

Sub_Panel

Hot_Water

Dryer

Oven_Stove

Fridge

Basement

Kitchen

Office

Kitchen_HW

Dishwasher

misc

net

total_power

The Source list that appears below the unit selection will list all of the sources that can produce a measurement with the selected until. Basically Volts and Hz pertain to VT inputs and all other units pertain to inputs (or outputs) that are configured with CTs. The list will change immediately when a new unit is selected.

On the right Watts are selected and this IoTaWatt lists 15 sources that are configured as power channels (CT inputs or CT based outputs). Note that the last entry shows a blue trash-can indicating that particular unit/source has been selected and is currently graphed.

Units

Volts

Watts

Wh

Amps

VA

PF

Hz

VAR

VARh

Source

Voltage

V2

To the left Volts are selected, so a different set of data sources corresponding to voltage inputs (VTs) are listed. Click on any of the individual sources in the list to add that data source to the graph, measured in the selected unit. This combination is called a trace. It will be assigned a distinguishing color and the list entry will now include a trash-can symbol with the assigned color as a background. The trace should also appear on the graph within several seconds.

While the list of sources may be the same for different units like Watts and Wh, they only show active if they are being graphed in the currently selected unit. In the case of the graph above, Total-Power is plotted as both Watts and Wh,

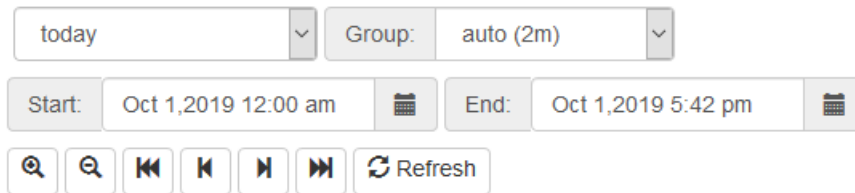
so it will appear as selected under both units, but note in the large picture above that the color is blue under the Watts unit, while it is green under the Wh units, indicating they are different traces.

One final note about the sidebar. It will disappear when the screen size gets to be too narrow to accomodate it. When that happens, it is replaced by a list icon at the top left of the main screen. Pressing that will overlay the main body with the selection sidebar. An X on the sidebar is used to hide it again.

17.2 Time period selector

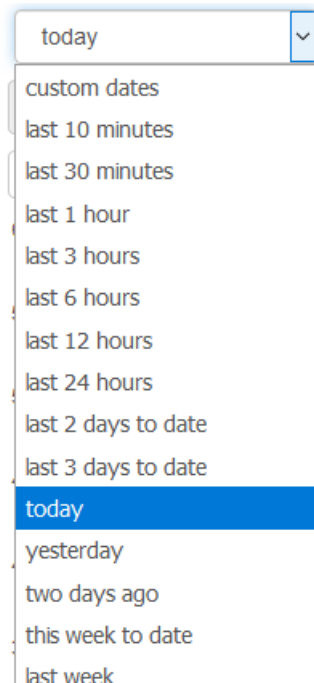
Graph+ plots all selected traces over a single time period that is defined and modified using the time period selection section. You can select one of the many pre-defined relative time periods or specify absolute dates and times. Once data has been plotted, the zoom/pan buttons can be used to modify the time-frame. Graphs with relative time-periods ending in the present can be set to automatically refresh.

iotahome Graph+



17.2.1 Period Selector

This is where you specify an initial time period, and where you go to change it. In the upper left is a dropdown selector that is used to specify common relative time periods.



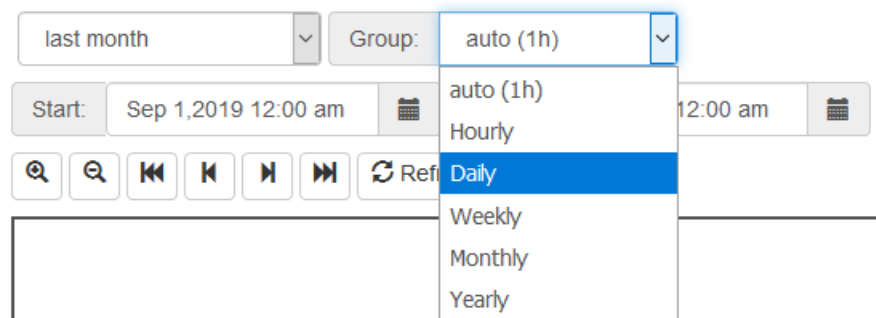
For most analysis of recent activity, these selections should make things quick and easy. All of these predefined periods are relative to the current date/time. So if you ask for today, you will get a plot of measurements from midnight to the

current time.

17.2.2 Group Selector

IoTaWatt records each measurement at 5 second intervals. That's 17,280 measurements per day. The graph size cannot possibly represent that, and it would take a long time to query that amount of data. Instead, IoTaWatt can deliver either the average value or net change of a measurement for any given interval. So when plotting a day's worth of data, we ask for automatic grouping which results in aggregating the data over 2 minutes in a 24 hour period. If you were looking at one hour or less, the grouping would be the highest resolution 5 seconds.

There are other choices when looking at longer time periods. Here we're plotting last month's usage and will group by day. This will return the average or change in value for each of the 28-31 days in the past month. All units are averaged except Wh, which return the total used in each grouping. So in this case, there would be 28-31 data points plotted.



It's important to note that when selecting grouping by day, week, month or year, you are not just getting the nominal grouping of 24 hours, 168 hours, etc. The selection process recognizes daylight-time in determining hours and days, and recognizes day of week, and month boundaries.

- Weeks begin and end at 12:00am on Sunday.
- Months begin and end at 12:00am on the 1st day of the month.
- Years begin and end at 12:00am on Jan 1.

17.2.3 Custom Date Selection

If the time period needed isn't covered in the selection list above, there are date pickers that can be used to choose specific start and end dates (and times). If you click on these dates, a calendar will appear to select a start and/or end date and time. When you change either of these dates, the period selector will automatically change to "custom dates" and the graph will be updated to span the new date specification.

This is only one of several ways to manipulate custom dates. There are two other ways to modify the dates bounding the current graph. The most obvious is the zoom/scroll bar.

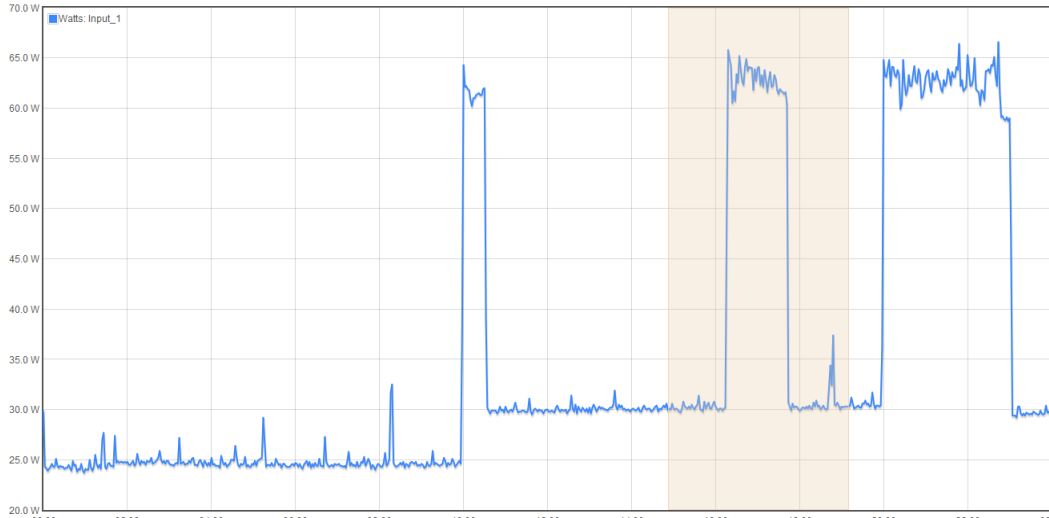


This bar works just as you think.

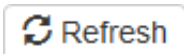
- Zoom (+) will zoom in 50% on the center of the graph.
- Zoom (-) will zoom out 100% on the center of the graph.
- Left Full (<<) Will shift the time into the past 100%, ending where it once began.


- Left Half (<) Will shift the time 50% into the past.
- Right Half (>) Will shift the time 50% into the future.
- Right Full (>>) Will shift the time 100% into the future, starting where it once ended.

There is one last way to modify graph period. You can simply select a subset of the graph window holding down the left mouse button. When you release it, the highlighted selection will become the new time period.




17.2.4 Refresh/Freeze



The  Refresh button will immediately refresh the current plot. If the time period of the current plot ends at the current time, as in the “Today”, “Last 10 minutes”, etc., the display will continue to refresh at the “interval” rate. For example if the auto interval is 2 minutes, it will continually refresh every 2 minutes. You will know it is auto refresh



mode because the Refresh button will change to  Freeze. If you click this button the auto refresh will stop and

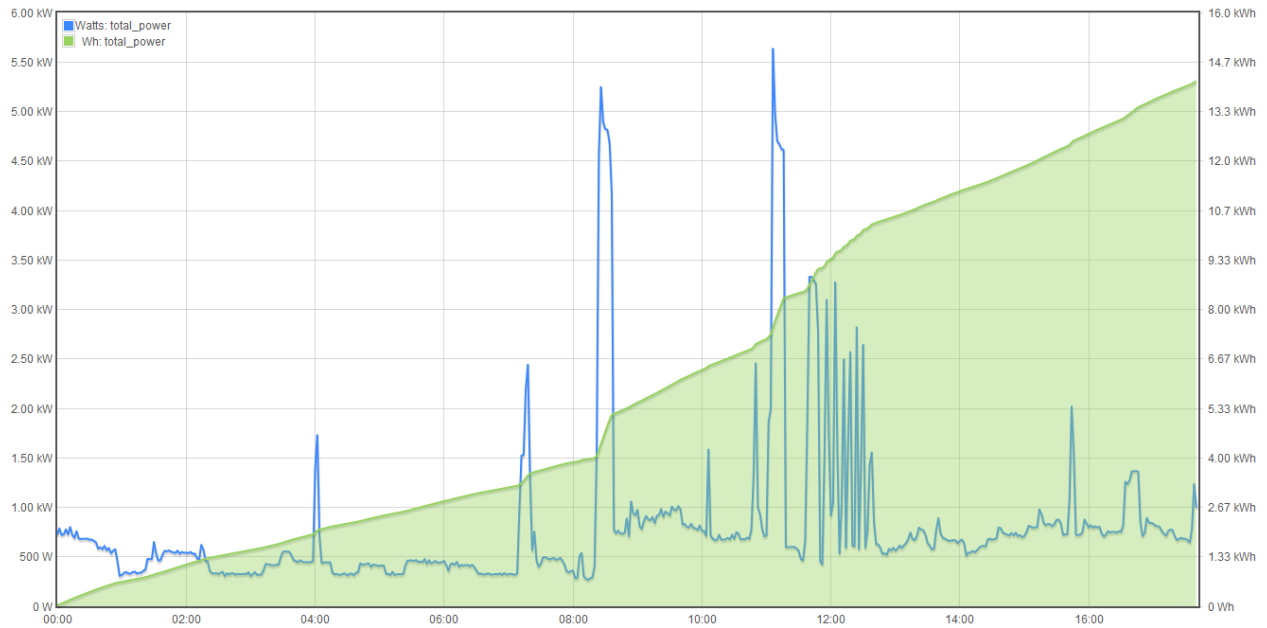


the button will revert to the manual  Refresh button.

The Refresh/Freeze state is retained when saving and subsequently restoring graphs. This feature is useful when loading a saved graph as an embedded window where there are no controls, allowing an auto refresh graph can be displayed in a frame.

17.3 Graph window

Once data sources are selected, the graph window comes to life.



In the upper left is the legend, a list of all of the unit/source combinations that are being plotted. The color of each trace matches the color used to designate the source selection and the color associated with the trace in the option table that will be described later.

There is no notion of a left or right Y-axis selection. Each unit that is included in the plot is alternately placed on the left and right side of the plot automatically. You know which scale pertains to each trace because the scales contain the unit designation.

17.4 Trace tables and options

The last major section of the window is the options and information table area. This is a multi-purpose area that displays different tables depending on the selection in the top row. There are four tables:

17.4.1 Options Table

CSV data								Yaxes Range	Data Statistics	Plot Options
Source	Color	Line/Bar	Fill	Stack	Accrue	Decimals	Scale			
↓ Watts:total_power		Line	<input type="checkbox"/>	<input type="checkbox"/>		1	1			
↑ Wh:total_power		Line	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	1	1			

This table lists all of the traces selected for the current graph, and allows modifying the default settings for each trace. Changing an option will have an immediate effect on the graph. There is no Save or Refresh required.

Arrows These sort arrows appear when there are two or more entries in the table. Use them to reorder the entries. Primarily helpful when using Stacked traces.

Trash Removes the trace from the graph and deselects from the source list.

Color Selects an override color.

Line/Bar Toggle between line or bar chart for this trace.

Fill Fill the area under the line or bar.

Stack Stack this trace above any other stacked traces appearing before it in this list. You can change the position of a trace (and so it's stack position) using the sort arrows appearing at the beginning of each entry when two or more traces are present.

Accrue This checkbox will appear on Wh traces and causes the Watt-hours to accrue in order to plot a running total. Wh are accrued in the sample graph to illustrate this feature.

Decimals The number of decimal places to request and plot. The default value is typically appropriate for the unit of measure, but sometimes increasing the precision provides a more detailed representation.

Scale The values returned in the query will be multiplied by this scale factor. This will affect the values in the CSV table as well.

17.4.2 Statistics Table

		CSV data	Yaxes Range		Data Statistics		Plot Options
	Source	Quality	Min	Max	Diff	Average	Sum
↓	Watts:total_power	100% (532/532)	267.7	5634.8	5367.1	800 W	14.2 kWh
↑	Wh:total_power	100% (532/532)	7.8	187.9	180.1	26.6 Wh	14.2 kWh

This is a list the traces with useful statistics.

Arrows These sort arrows appear when there are two or more entries in the table. Use them to reorder the entries. Primarily helpful when using Stacked traces.

Trash Removes the trace from the graph and deselects from the source list.

Quality This indicates the number and percent of groups for which data was available. It is typically 100%, but could be less because of power failures or malfunction during the period.

Min The smallest group value graphed for this trace.

Max The largest group value graphed for this trace.

Diff Difference between Min and Max.

Average Mean value of all of the groups graphed for this trace. Does not include null values.

Sum For Watt and Wh traces this is the total Wh for the period.

17.4.3 Yaxes Range Table

			CSV data	Yaxes Range	Data Statistics	Plot Options
Unit	Plot Range	Data Range	Min	Max		
Watts	0 W to 6.00 kW	267.7 to 5634.8	<input type="text" value="auto"/>	<input type="text" value="auto"/>		
Wh	0 Wh to 16.0 kWh	0.0 to 14152.8	<input type="text" value="auto"/>	<input type="text" value="auto"/>		

This list is used to modify the Yaxis range for each unit. The default is “auto”, which works well to represent the full range of the data, however the Min and/or Max can be specified here to override the auto default. Once changed, the new limit will remain in effect until one of these things happens:

- The override value is removed.
- A saved graph is loaded.
- The reset button is clicked.

Plot Range: The Yaxis range used in the current graph.

Data Range: The actual range of all of the traces using this unit.

Min: The lower bound to be used in subsequent graphs, or blank for auto assignment.

Max: The upper bound to be used in subsequent graphs, or blank for auto assignment.

17.4.4 CSV Data

CSV data		Yaxes Range	Data Statistics	Plot Options
Time Format:	Date-time string	Null values:	Show	Copy Download
2019-10-01 00:00:00, 722.2, 24.07 2019-10-01 00:02:00, 788.1, 50.34 2019-10-01 00:04:00, 720.3, 74.35 2019-10-01 00:06:00, 726.6, 98.57 2019-10-01 00:08:00, 777.7, 124.49 2019-10-01 00:10:00, 727.3, 148.73 2019-10-01 00:12:00, 801.0, 175.43 2019-10-01 00:14:00, 723.9, 199.56 2019-10-01 00:16:00, 690.9, 222.59 2019-10-01 00:18:00, 758.5, 247.87 2019-10-01 00:20:00, 683.6, 270.66 2019-10-01 00:22:00, 680.2, 293.33 2019-10-01 00:24:00, 682.4, 316.08 2019-10-01 00:26:00, 683.3, 338.86 2019-10-01 00:28:00, 683.7, 361.65 2019-10-01 00:30:00, 672.1, 384.05 2019-10-01 00:32:00, 675.1, 406.55 2019-10-01 00:34:00, 662.2, 428.62 2019-10-01 00:36:00, 647.2, 450.19 2019-10-01 00:38:00, 584.4, 469.67 2019-10-01 00:40:00, 596.8, 489.56 2019-10-01 00:42:00, 577.4, 508.80 2019-10-01 00:44:00, 611.1, 529.17 2019-10-01 00:46:00, 566.1, 548.04 2019-10-01 00:48:00, 589.2, 567.68 2019-10-01 00:50:00, 536.3, 585.56 2019-10-01 00:52:00, 561.8, 604.29 2019-10-01 00:54:00, 576.2, 623.50 2019-10-01 00:56:00, 457.2, 638.74 2019-10-01 00:58:00, 307.1, 648.98 2019-10-01 01:00:00, 316.7, 659.54				

This is a comma-separated-values listing of all of the data used in the current graph. The first column is the time, subsequent columns are the group values for the traces in the order that they are listed in the options or statistics tables.

There are a couple of options available:

Time Format:

- Date-time string - selects a date and time format acceptable to spreadsheets.
- Seconds-from-start - selects a count of seconds from the start time.
- Unix-time - selects the count of seconds from Jan 1, 1970 UTC.

Null Values:

- Show - include missing or invalid lines with “null” as a value.
- Remove line - Where a line has a null value, remove the entire line from the display.

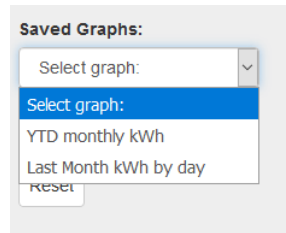
Copy: Copy the contents of the CSV table to the clipboard.

Download: Download the CSV data as a file.

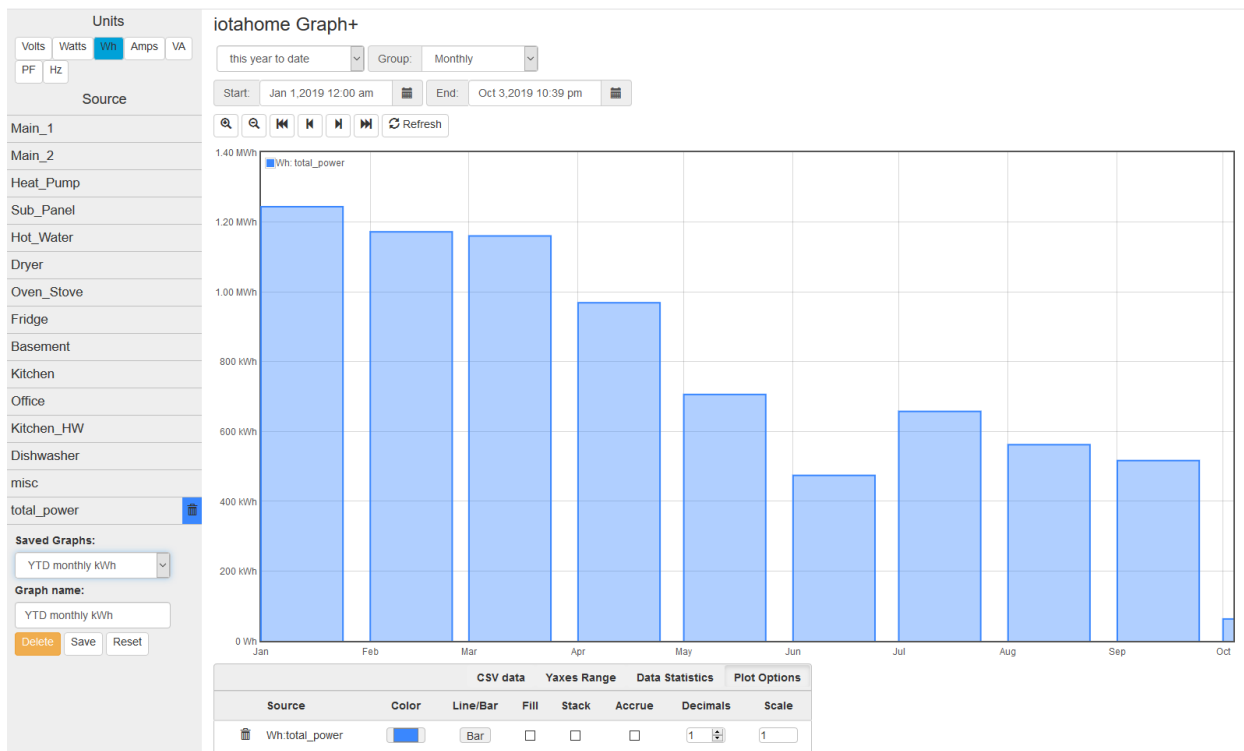
17.5 Saving Graphs

Graph+ is great for ad-hoc queries, but it can take some time to fine-tune a busy graph, and it recreating it weekly or monthly can get old. This is where the ability to save a set of graph specification comes in handy.

At the bottom of the sidebar you can save and reload any number of graphs. Each graph is saved in the IoTaWatt, so it doesn't matter if you use a different browser or device to recall them, they will always be there.



Once you have a graph that you like, enter a description in the *Graph name:* box. The **save** button will appear. Click it. The graph has been saved. Click the *Saved Graphs:* selector and a list of all of the saved graphs will appear. Click any selection and Graph+ will load that graph specification.



This is a graph of total monthly kWh to date. Once loaded, I can do a lot with it:

- Change group to Weekly to see it by week.
- Change the period to last-year or anything else.
- Show the CSV table to see list of usage by month.

- Add traces for particular circuits, stacking if appropriate, to see a breakdown.
- Move the data to a spreadsheet to apply your tariff and convert to cost.

Note that you are saving the graph *specification*, not the actual graph. If you save a graph of yesterday and reload it tomorrow, it will plot today. If you want to save a static graph, select the *custom dates* period at the top before saving.

Whenever a saved graph name is in the *Graph name:* box, the **Delete** key will be available. To change a graph specification, simply load it, make the changes and save it again.

17.6 Running Directly with URL

Graph+ can be loaded directly from the IoTaWatt's web server using the URL

<http://iotawatt.local/graph2.htm> [?graph=savedgraph [&embed]]

Substitute your local hostname/IP address if different.

17.6.1 graph=

Optional query parameter to specify a saved graph that is to be loaded initially.

17.6.2 embed

Causes Graph+ to display only the plot window of the selected graph. If the saved graph has refresh enabled, the plot will refresh at the active interval.

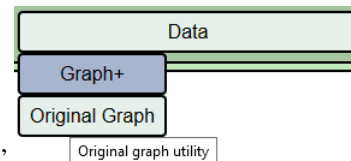
17.7 Reset

Sometimes you just want to start over with a clean slate like the app was just loaded.

CHAPTER 18

Original Graph

The original local graphing feature was derived from the Emoncms graphing application and adapted to report the input and output channels of IotaWatt directly. It is available on devices connected to the same WiFi network as the IotaWatt or on the internet using local port forwarding.

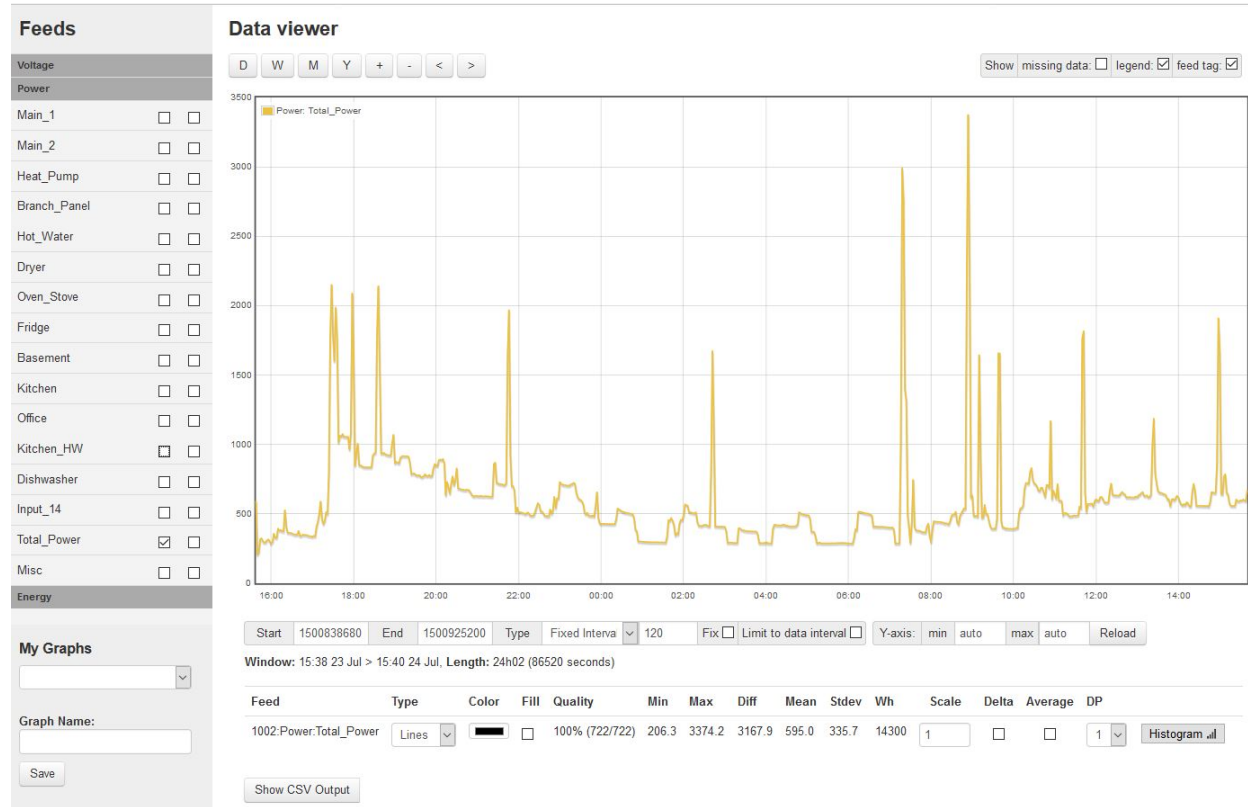


From the configuration app click “Data” and then select “Original Graph”

The screenshot shows the Emoncms Data viewer interface. On the left, there is a sidebar with 'Feeds' (Voltage, Power, Energy) and 'My Graphs' (a dropdown menu, a 'Graph Name:' input field, and a 'Save' button). The main area is titled 'Data viewer' and contains a large empty graph area. Above the graph area are navigation buttons: 'D', 'W', 'M', 'Y', '+', '-', '<', '>'. To the right of these buttons are checkboxes for 'Show', 'missing data', 'legend', and 'feed tag'. Below the graph area, there is a 'Window' section with 'Start' and 'End' input fields, a 'Type' dropdown menu, a 'Fixed Interval' checkbox, a 'Limit to data interval' checkbox, 'Y-axis' settings (min, auto, max, auto), and a 'Reload' button. At the bottom, it displays 'Window: 15:38 23 Jul > 15:40 24 Jul, Length: 24h02 (86520 seconds)'.

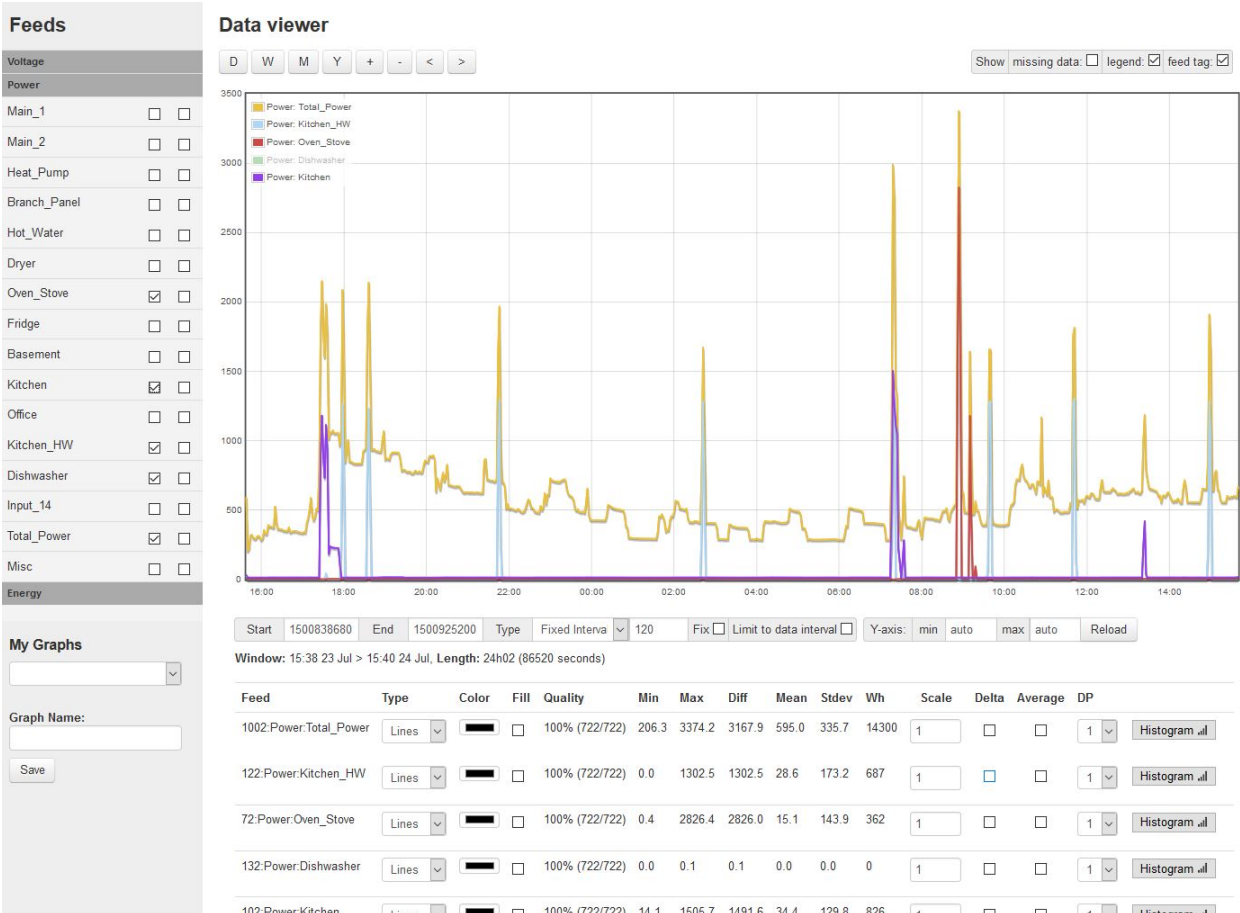
Press the Voltage, Power, and Energy tabs under Feeds to list your IotaWatt input and output channels. Select the Power

tab and then select one or more of your channels listed there. Selecting Total_Power (actually an output channel that is the sum of Main_1 and Main_2) we get the following graph.



Notice that the graph covers a 24 hour period. That's the default. You can select the period using the D, W, M, Y buttons at the top, and you can move the time frame left or right and zoom in or out.

Back to this graph. It reveals that average power is about 600 watts. To be precise, the mean power is 595 watts and the total power for the day was 14,300 watt-hours (14.3Kwh) as indicated in the feed statistics at the bottom. Wondering what might be causing those spikes at various times throughout the day? Lets lay some other input channels on top.



Now the story unfolds. Most of the 1000 watt spikes are cycles of a point-of-use electric water heater in the Kitchen (Kitcher_HW). The big spike around 9pm is the Oven/Stove circuit, and the rest of the spikes seem to line up with kitchen appliances.

That’s just a sample of the power of the graphic presentation possible both locally and with essentially the same tool on Emoncms.org. Practically speaking, those power spikes are not the meat-and-potatoes of home energy use. Other circuits reveal the contribution of workhorse appliances like the refrigerator, freezer, heat-pump (not active this day), clothes dryer, computer and office machines. Kwh can be plotted as well, and compared to the whole. With 14 input channels, its possible to divide household usage into manageable components.

19.1 IoTaWatt file systems

IoTaWatt has two file systems. The primary file system is maintained on the internal SDcard and is formatted as FAT32. Typical SDcard size is 8Gb. All of the web server files as well as the data and message logs are maintained on the SDcard.

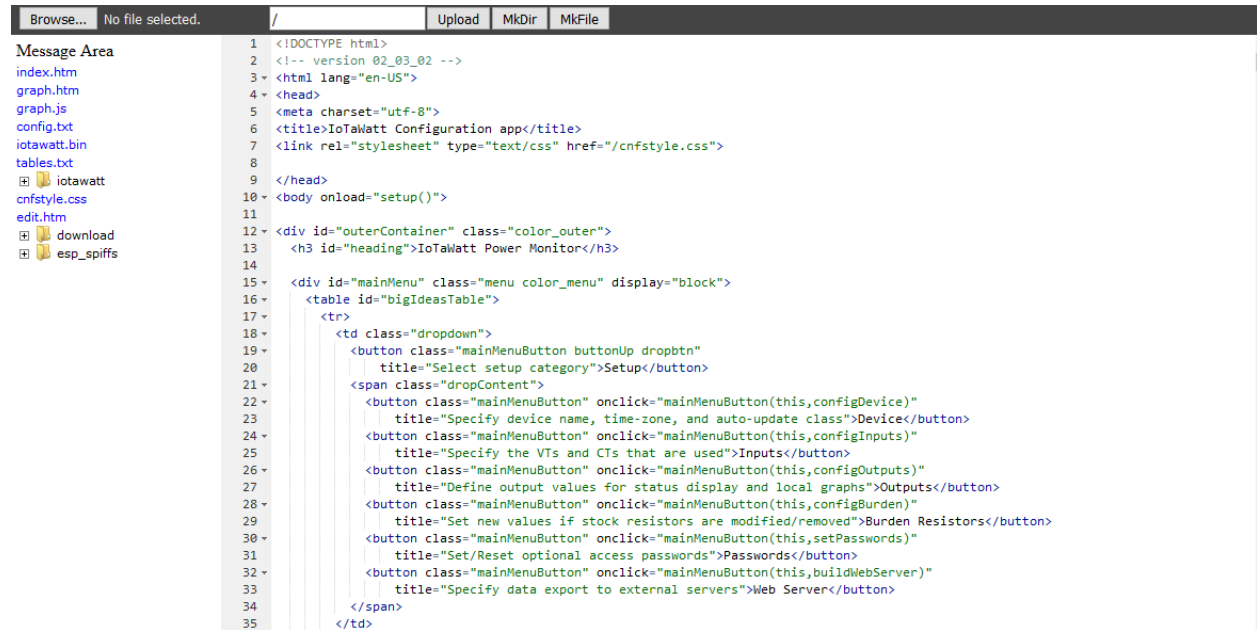
The ESP8266 also has an onboard file system called the SPI Flash File System or SPIFFS that is much smaller at 1Mb. IoTaWatt uses that area for limited device specific information. This file system uses a flat directory structure, however IoTaWatt presents it as hierarchical through the file manager.

19.2 File Manager

The integrated web server can be used to access and manage the files on both the SDcard and in the SPIFFS file system. The web server is based on the generic ESPWebserver developed by the folks at the ESP8266/Arduino project. The file manager application is the unmodified application that they distribute. The major functions are:

- Display file system
- Delete files
- Upload files to the IoTaWatt
- Download files from the IoTaWatt
- Edit files

The File Manager is accessed from the dropdown buttons in the Tools main menu.



The left column is the file list. Directories can be expanded by clicking the + sign as in the typical paradigm. Action can be taken on individual files by right clicking and selecting from the options:

- Edit
- Download
- Delete - Proceed cautiously. There is no confirmation popup. Delete is delete now!

19.3 Downloading Files

To download a file, right-click it in the file manager as above and select the Download option. Exactly how it is handled and where the file is downloaded is a function of your operating system and browser, but it will follow the normal protocol.

19.4 Uploading Files

Files can be uploaded to the IoTaWatt by clicking the Browse button at the top and selecting a file in the manner provided by your browser. Once selected, click Upload to transfer the file to the IoTaWatt file system. You can edit the pathname before uploading. Additional buttons are provided to create a new directory or file.

19.5 SPIFFS

The SPIFFS is presented as the directory esp_spiffs. Expanding that directory will reveal the contents of the SPIFFS with a pseudo hierarchical directory structure. Files can be accessed in the same way as on the SDcard.

19.6 ACE Editor

One of the most powerful features of this app is the editor. It's a version of the open [Ace Editor](#) and is very fast and capable. It's not particularly useful on mobile devices, but does very well on a keyboard equipped browser. You will need to learn the keyboard shortcuts, but even with just a few of the usual suspects, you can be very productive (ctrl-S save, ctrl-Z undo, ctrl-F find, etc.)

The IoTaWatt configuration app was developed exclusively using this editor and an IoTaWatt. In fact, that's the file that appears in the editor window when you start the file manager app.

20.1 Overview

The IoTaWatt Query API provides access to the historical data in the datalogs using a restful interface that produces a table of JSON or CSV data. The table columns can be time, IoTaWatt inputs and IoTaWatt outputs. The table rows are datalog values grouped by a fixed time period or relative time periods like days, weeks, months and years.

Values can be requested for a variety of measurements such as Watts, Volts or Amps.

The Query API provides data to the Graph+ utility.

20.2 Query types

There are two basic queries currently supported:

20.2.1 show

Used to obtain a list of all inputs and outputs available to the query.

20.2.2 select

Used to select a set of series for a particular time period and return a table of values in JSON or CSV format.

20.3 query?show

This is the only form of the show query:

```
HTTP://iotawatt.local/query?show=series
```

This query simply lists all of the available inputs and outputs along with their respective primary unit of measure. The format is always JSON.

The response lists the series names that the select query will recognize. Here is a typical response from a simple configuration:

```
{ "series": [ { "name": "voltage", "unit": "Volts" }, { "name": "mains1", "unit": "Watts" },
{ "name": "mains2", "unit": "Watts" }, { "name": "solar", "unit": "Watts" },
{ "name": "heat_pump", "unit": "Watts" }, { "name": "mains", "unit": "Watts" },
{ "name": "used", "unit": "Watts" } ] }
```

20.4 query?select

20.4.1 select=[*series1* [, *series2* ...]]

Required parameter. Specifies the series to be returned in the columns of the response. There are three types of series that can be requested:

time [*.local* | *.utc*] [*.iso* | *.unix*] Returns the time of the beginning of the reporting group. Modifiers can be used to specify local (default) or utc time, and iso date format (default) or unix seconds since 1/1/1970 format.

<voltage input or output> [*.volts* | *.hz*] [*.d<n>*]

A voltage input or output. A unit modifier can be used to specify:

- *.volts* (default)
- *.hz* (frequency).

The *.d<n>* modifier overrides the default number of decimal digits.

<power input or output> [*.watts* | *.amps* | *.wh* | *.va* | *.var* | *.varh* | *.pf*] [*.d<n>*]

A power input or output. A unit modifier can be used to specify:

- *.watts* (default)
- *.amps*
- *.wh* (watt-hours)
- *.va* (volt-ampere)
- *.var* (volt-ampere-reactive)
- *.varh* (volt-ampere-reactive hours)
- *.pf* (power factor)

The *.d<n>* modifier overrides the default number of decimal digits.

An example might be: *select=[time.local.unix,mains.watts.d0,solar.wh.d1]*

20.4.2 &begin=<time specifier>

Required parameter. Time can be specified in a variety of relative and absolute formats. See *time specifiers*

20.4.3 &end=<time specifier>

Required parameter. Time can be specified in a variety of relative and absolute formats. See *time specifiers*. end must be greater than begin.

20.4.4 &group={ auto | all | <n> {s | m | h | d | w | M | y}}

Optional parameter. The datalog contains measurements at 5 second (current log) and 1 minute (history log) intervals. This parameter specifies how to group those measurements into rows in the response table. The simple way, useful for detailed examination of short time periods, is to use some fixed number of seconds or minutes. When looking at longer time periods, it can be useful to group by hour, day, week etc. IoTaWatt knows where these boundaries are and will return the correct grouping taking into account daylight time changes, days-in-month, and leap years.

The default is **auto**, which selects a fixed time group to yield about 360 rows (resolution=low) or 720 rows (resolution=high).

all will cause all of the data in the time period to be treated as a single group. For most units, this will result in the average value over the entire period. For Wh, it will result in the total Wh for the entire period.

To override, specify a time unit preceeded by a multiplier, as in:

- 10s (ten seconds)
- 5m (five minutes)
- 1h (one hour)
- 1M (one month) *note case sensitive m=minutes, M=months*

20.4.5 &format={ json | csv }

Optional parameter specifies the format of the query response. The default is **json**.

json Json format where the response is a Json object enclosed in brackets { } and the data table is a json array "data":[[series1,series2,...],[series1...]]

csv Comma Separated Values table.

20.4.6 &header={ no | yes }

Optional parameter specifies if a header is to be included to describe the columns (series) included in the response. Default is **no**.

For *&format=csv*, a row is prepended to the data with a comma delimited list of the series names.

For *&format=json*, the array "labels":[series1 [,series2 ...]] is added to the response. Another array "range":[begin, end] is added where begin and end are the 10 digit absolute unix begin and end times of the response.

20.4.7 &missing={ null | skip | zero }

Optional parameter specifies what to do when a missing value is encountered when building a response row.

null Use the value null.

zero Use the value zero.

skip Suppress the entire response row.

20.4.8 &resolution={ low | high }

Optional parameter specifies the relative resolution of the response table when *&group=auto*. The default is **low**. For more information see *&group=* above.

20.4.9 &limit={n | none }

Optional parameter overrides the default output limit. The default is 1,000 lines.

n Maximum lines generated

none No limit, query runs to completion

Query is a blocking request. The IoTaWatt does not sample power while responding to a query. Short queries, as issued by Graph+, are of little consequence. They process in a second or less. To avoid unintended long lapses, a limit is placed on the number of lines (groups) that are returned by the query. To understand the time required for longer queries, you can experiment with a subset and scale the time up.

If the limit is reached, output will stop with a full line. If the format is json and header=yes, the response will include an object called “limit” with a value of the UTC timestamp of the next line that would have been produced. If the format is CSV, the following message will be appended with the UTC timestamp of the next line that would have been produced.

```
Limit exceeded at <UTCtime>
```

20.5 time specifiers

A time specifier can define a date/time in absolute or relative terms. Three different formats are allowed:

- *Unix time*
- *ISO time*
- *Relative time*

20.5.1 Unix time

Unix time is the count of seconds or milliseconds since Jan 1, 1970. a Unix time specifier is simply a 10 digit integer for seconds or a 13 digit integer for milliseconds. IoTaWatt will always round the time to a multiple of 5 seconds.

20.5.2 ISO time

A subset of the ISO 8601 standard can be used to specify an absolute date and time. The supported format is:

```
YYYY [-MM [-DD [Thh [:mm [ :ss [Z]]]]]]]
```

As you can see, the only thing required is the year, which must be four digits. That is optionally followed by:

-MM a two digit month 01-12

-DD a two digit day in month 01-31

Thh two digit hours 00-23

:mm two digit minutes 00-59

:ss two digit seconds 00-59

z indicates the time is UTC rather than local time

Some examples are:

2018-01-01 Start of the year 2018, equal to 2018-01-01T00:00:00 or just 2018

2019-04-15T11:42:15 April 15, 2019 11:42:15

20.5.3 Relative time

Specifies a point in time relative to the current time. Makes it possible to specify “today”, “yesterday”, “last week” etc. All relative time specifiers begin with a base date or time as follows:

Relative dates all begin at 00:00:00 local IoTaWatt time.

- **y** - Jan 1, of the current year
- **M** - The first day of the current month
- **w** - The first day of the current week (weeks start on Sunday)
- **d** - The current day

Relative time.

- **h** - first minute and second of the current hour.
- **m** - First second of the current minute.
- **s** - The current second (rounded down to 5 second multiple).

So if “today” is 2019-04-15T16:11:42:

Base	ISO time
y	2019-01-01T00:00:00
M	2019-04-01T00:00:00
w	2019-04-14T00:00:00
d	2019-04-15T00:00:00
h	2019-04-15T16:00:00
m	2019-04-15T16:11:00
s	2019-04-15T16:11:40

Base time may be followed by one or more offset modifiers to add or subtract from the base time. The format is:

{ + | - } [n] { y | M | w | d | h | m | s }

Examples:

Base with modifiers	Effective time
d-1d	00:00:00 yesterday
d-18h	06:00:00 yesterday
s-3h	Three hours ago
y-1M	Last December
w-1w+3d+12h	Noon on Wednesday of last week
s	Now

By using relative time for both **begin** and **end**, relative time periods can be specified:

begin	end	period
d-1d	d	yesterday
M-1M	M	Last month
d	s	Today to date
s-12h	s	Last 12 hours
w-1w+2d	w-1w+3d	Tuesday of last week
y	s	Year to date

20.6 Responses

20.6.1 400 invalid query.

The query has a missing or invalid specification. The response is a json object “error”:”*<error details>*”.

query:

```
HTTP:// ... /query?select=[time.iso,heap_pump,misc]&begin=d-1d&end=d&group=h
```

response:

```
{"error":"invalid query. Invalid series: heap_pump"}
```

20.6.2 200 Success

The query succeeded and the response is sent.

csv Response is the table of csv formatted lines.

query:

```
/query?select=[time.iso,Heat_Pump,misc]&begin=d-1d&end=d&group=h&format=csv&  
↪header=yes
```

response:

```
Time, Heat_Pump, misc  
2019-10-16T00:00:00, 333, 125.5  
2019-10-16T01:00:00, 332.2, 121.4  
2019-10-16T02:00:00, 446.8, 116.8  
2019-10-16T03:00:00, 416.8, 114.3  
2019-10-16T04:00:00, 415.4, 109.9
```

(continues on next page)

(continued from previous page)

```

2019-10-16T05:00:00, 582.9, 111.4
2019-10-16T06:00:00, 711.8, 113.3
2019-10-16T07:00:00, 783.5, 117.1
2019-10-16T08:00:00, 619.6, 117.5
2019-10-16T09:00:00, 333, 116.4
2019-10-16T10:00:00, 339.8, 164.5
2019-10-16T11:00:00, 345.1, 180.6
2019-10-16T12:00:00, 345.6, 114.5
2019-10-16T13:00:00, 345.3, 111.8
2019-10-16T14:00:00, 344.3, 130.9
2019-10-16T15:00:00, 343.4, 302.5
2019-10-16T16:00:00, 343.1, 271.6
2019-10-16T17:00:00, 342, 264.5
2019-10-16T18:00:00, 342.3, 114.1
2019-10-16T19:00:00, 343, 117
2019-10-16T20:00:00, 342.7, 118
2019-10-16T21:00:00, 343.9, 136
2019-10-16T22:00:00, 344.9, 120.2
2019-10-16T23:00:00, 345.7, 124.2``

```

json Response is a json object.

query:

```

HTTP:// ... /query?select=[time.iso,Heat_Pump,misc]&begin=d-1d&end=d&group=h&
↪format=json&header=yes

```

response:

```

{"range": [1571198400, 1571284800],
 "labels": ["Time", "Heat_Pump", "misc"],
 "data": [ ["2019-10-16T00:00:00", 333, 125.5],
  ["2019-10-16T01:00:00", 332.2, 121.4],
  ["2019-10-16T02:00:00", 446.8, 116.8],
  ["2019-10-16T03:00:00", 416.8, 114.3],
  ["2019-10-16T04:00:00", 415.4, 109.9],
  ["2019-10-16T05:00:00", 582.9, 111.4],
  ["2019-10-16T06:00:00", 711.8, 113.3],
  ["2019-10-16T07:00:00", 783.5, 117.1],
  ["2019-10-16T08:00:00", 619.6, 117.5],
  ["2019-10-16T09:00:00", 333, 116.4],
  ["2019-10-16T10:00:00", 339.8, 164.5],
  ["2019-10-16T11:00:00", 345.1, 180.6],
  ["2019-10-16T12:00:00", 345.6, 114.5],
  ["2019-10-16T13:00:00", 345.3, 111.8],
  ["2019-10-16T14:00:00", 344.3, 130.9],
  ["2019-10-16T15:00:00", 343.4, 302.5],
  ["2019-10-16T16:00:00", 343.1, 271.6],
  ["2019-10-16T17:00:00", 342, 264.5],
  ["2019-10-16T18:00:00", 342.3, 114.1],
  ["2019-10-16T19:00:00", 343, 117],
  ["2019-10-16T20:00:00", 342.7, 118],
  ["2019-10-16T21:00:00", 343.9, 136],
  ["2019-10-16T22:00:00", 344.9, 120.2],
  ["2019-10-16T23:00:00", 345.7, 124.2]] }

```


CHAPTER 21

Message Log

The message log is the IoTaWatt's Diary. It makes notes about various events that occur, both ordinary and unusual. These messages can be helpful in understanding what is happening with the device, or to provide insight into why something doesn't appear to be working as it should.

The primary method of accessed the message log is by hovering over the Tools main menu button and clicking the Message Log button. The config utility will cause your browser to download and display the most recent 10000 characters in the log. You can access the entire log by using the File Manager app to download the file `/iotawatt/iotamsgs.txt` and viewing with a text file editor.

With the exception of the first messages after startup, all of the messages in the log will have a date and time stamp. Messages issued by one of the IoTaWatt Services will begin with the name of the service followed by a colon, as in this message issued from the datalog Service indicating it has successfully started.

```
12/23/17 21:57:06 dataLog: service started.
```

While it would be impossible to maintain a list of all of the possible messages, the messages issued at startup are pretty straightforward and provide a lot of information.

```
** Restart **

SD initialized.
1/08/19 19:39:43z Real Time Clock is running. Unix time 1546976383
1/08/19 19:39:43z Power failure detected.
1/08/19 19:39:43z Reset reason: External System
1/08/19 19:39:43z ESP8266 ChipID: 427289
1/08/19 19:39:43z IoTaWatt revision 4.9, firmware version 02_03_20
1/08/19 19:39:43z SPIFFS mounted.
1/08/19 14:39:44 Local time zone: -5:00
1/08/19 14:39:44 Using Daylight Saving Time (BST) when in effect.
1/08/19 14:39:44 device name: IotaWatt
1/08/19 14:39:47 Connecting with WiFiManager.
1/08/19 14:39:53 MDNS responder started for hostname IotaWatt
1/08/19 14:39:53 LLNMR responder started for hostname IotaWatt
1/08/19 14:39:53 HTTP server started
```

(continues on next page)

(continued from previous page)

```
1/08/19 14:39:53 WiFi connected. SSID=flyaway, IP=192.168.1.102, channel=1, RSSI -69db
1/08/19 14:39:53 timeSync: service started.
1/08/19 14:39:56 statService: started.
1/08/19 14:39:56 Updater: service started. Auto-update class is BETA
1/08/19 14:39:57 dataLog: service started.
1/08/19 14:39:57 dataLog: Last log entry 01/08/19 14:39:35
1/08/19 14:39:57 historyLog: service started.
1/08/19 14:39:57 historyLog: Last log entry 01/08/19 14:39:00
1/08/19 14:39:58 influxDB: started, url=192.168.1.101:8086, db=iotawatt, interval=10
1/08/19 14:39:58 Updater: Auto-update is current for class BETA.
1/08/19 14:39:58 influxDB: Start posting at 01/08/19 14:39:40
```

Some things to look for:

```
Real Time Clock is running. Unix time 1546976383
```

Except for initial setup, the RTC should always be running. If it is not, suspect the battery needs replacement.

```
Power failure detected.
```

This indicates that power interrupted prior to the restart.

```
Reset reason: External System
```

This is the reason for the restart. External System is a normal cause. Reasons like WDT and Exception indicate program faults and, if frequent, should be posted to the support forum.

```
IoTaWatt revision 4.9, firmware version 02_03_20
```

The hardware and firmware versions. Good to know if things go wrong, and also useful to get the corresponding version of this documentation.

```
Local time zone: -5:00
```

The local time zone specified in your config file. This message and all subsequent messages should be timestamped with local-time as opposed to UTC which was indicated in prior messages with by the suffix 'z'.

```
device name: IotaWatt
```

This is the external name of the unit. It will be used as the hostname when connecting to WiFi and it is the name that you must use to access the device as in <http://iotawatt.local>.

```
WiFi connected. SSID=flyaway, IP=192.168.1.102, channel=1, RSSI -69db
```

Indicates connection to WiFi and the IP address assigned. The RSSI is an indication of WiFi signal strength. A good number would be between -50 and -78 or so. If you are having WiFi problems, this metric along with the channel number can be helpful in resolving.

```
timeSync: service started.
```

All of the regular services will log their startup. Some will also provide additional information about their configuration or state. All messages from system services begin with the name of the service followed by a colon.

22.1 Led Indicator

The IotaWatt has a bi-color external LED indicator that is usefull in determining the current state and for explicitly indicating serious problems. Whenever there is a question about whether something may be wrong, it is important to look at the LED indicator. The LED can blink RED (R) or GREEN (G). When indicating a device state, it will repeat a sequence of colored blinks.

22.2 Dull Green Glow

This typically means that the device is connected and working properly.

22.3 Dull Red Glow

This indicates that the device is working properly, but either there is no WiFi connection or the real time clock is not initialized and the time cannot be established from time-servers on the internet via WiFi. As long as the real time clock is initialized, the IotaWatt can run indefinitely without WiFi. If you are logging to Emoncms or another server, that will be suspended, but IotaWatt will recognize when the WiFi connection is restored, change the led to dull green, and send all of the data that it was unable to send during the outage.

To determine if IotaWatt is connected to the WiFi network, try to run the IotaWatt app from a browser on a device that is connected to the same WiFi network. If it works, the problem was probably the real time clock.

If the app doesn't start, there is a problem with the WiFi connection. To be sure, restart the IotaWatt by disconnecting the 5VDC power momentarily and then observe the led during startup. Follow the troubleshooting guide for led indications during startup.

22.4 Not Illuminated

Under virtually all circumstances, the led should be illuminated. If not, the most probable cause is that there is no 5VDC supply. The power supply may be faulty or the USB plug may not be inserted all the way. Check these things first. Try unplugging the 5VDC USB supply and reconnecting. Try a different 5VDC supply if available. If all else fails, a more serious problem with the device must be considered. Consult the supplier or someone who can troubleshoot electronic the hardware.

22.5 Continuous Red-Green-Red-Green. . . .

Downloading new firmware release. Updates are published from time to time and most IotaWatt subscribe to an auto-update class for which the current release may change. When a new release becomes available, the device initiates a download of a large file that contains all of the elements of the new release. The download usually takes about 5-6 seconds, but can take longer. While the download is in progress, the IoTaWatt does nothing else and will blink the led RED-GREEN continuously. At the completion of the download, the release will be installed and your IoTaWatt will restart. The entire process should never take more than a minute.

22.6 Led Sequences

22.6.1 red-green-green

IoTaWatt is having trouble connecting to the WiFi network. If this is a new IoTaWatt, or the network has changed, you will need to specify a new network. Otherwise, wait several minutes and the LED pattern should change to a new code. Follow the advice for that code.

22.6.2 red-green-red

A corrupted datalog (current or history) has been discovered. The file is being scanned from beginning top end and a diagnostic file will be created. This can take an hour or more for large logs. Allow to run to completion, after which the damaged log will be deleted and a new log created.

22.6.3 red-red-green

The configured WiFi network is unavailable and the real-time-clock is not running. IoTaWatt can run without WiFi, but it cannot accumulate log data if it doesn't know what time it is, and it sets the real-time-clock from the internet. Determine the problem with your WiFi network or connect to a new network.

22.6.4 green-red-red

IoTaWatt is having trouble accessing the SDcard inside the device. This typically happens during a restart of the device. You will need to power off the device and open it up by removing the four screws located under the rubber feet. Check that the SDcard is firmly seated in the socket and retry.

If problem persists, your card may have failed. You can try to insert the card into a computer that accepts SD cards to see if it can be read. If so, the problem is probably the IoTaWatt SD card socket and the device will need to be replaced. The good news is that the card should work in a replacement device and pick up where you left off with all historical data intact.

If your card cannot be read by another device, it is probably the card itself. Unfortunately, these things do fail from time-to-time. To be sure, insert any other available SDcard and try to restart. If the error indication changes to something else, it is probably a failure of the SDcard. Sadly, you will need to obtain a new one and initialize it with the files that are in the SD directory of the GitHub project. All historical data is lost and you will need to reconfigure the device as if were new.

22.6.5 red-red-red

This is a catch-all panic code, where the firmware has detected a situation that should not happen, or is impossible to deal with. Possibly there will be some diagnostic clues in the message log, or it may require connecting a serial terminal to the USB port to obtain further diagnostics. Problems in this category are beyond the scope of this document.

22.6.6 green-red-red-red

The config.txt file was not found on the SDcard. Possibly the SDcard has been damaged, or the file was inadvertently deleted. The card must be removed, examined in another computer, and a new config.txt file provided.

22.6.7 green-red-red-green

The config.txt file format is invalid (not valid Json) and could not be used. This could be the result of editing the file improperly, or an error in saving the configuration. It may be possible to repair the file by mounting the SDcard in another computer and using a Json linter to find the errors. Otherwise, it may be necessary to replace the config.txt file.

23.1 How does it work?

The IoTaWatt connects to your local area network via WiFi. The procedure for connecting is outlined [here](#). When your router accepts the connection, it assigns an IP address and necessary routing information to the IoTaWatt using the Dynamic Host Configuration Protocol (DHCP). Depending on your router settings, that IP address will either be a fixed local IP address, or a semi-random IP address chosen from a pool of available addresses.

When you enter **iotawatt.local** into your browser, your computer uses one of several similar [zeroconf](#) protocols to discover the IP address that has been assigned to the IoTaWatt. These protocols go under several names: Bonjour (Apple) and LLMNR (Microsoft). See the [zeroconf](#) link for the detailed Wiki.

Essentially, when you type iotawatt.local into your browser, the underlying networking layer in your computer broadcasts a datagram message available to all members of the LAN, asking if there is anyone out there that called iotawatt.

At startup, IoTaWatt creates a process that listens for those datagrams. When it hears it's name, it responds to the sender saying "I'm iotawatt and my IP address is xx.xx.xx.xx". The requestor makes a note of this address and uses it to send subsequent transactions to iotawatt.local.

23.2 How does that *not* work?

Sounds simple doesn't it? What could possibly go wrong?

Plenty. First off, these zeroconf protocols are not part of the standardized internet. If your computer doesn't have some version of the protocol installed, it won't work.

There are other issues. Remember that your computer resolves the name and then stores the IP for future use? Well, if the IoTaWatt gets assigned an IP address from a pool of addresses, it can change without your computer knowing it. That is a common problem. Your computer can talk to the IoTaWatt for days or weeks and then suddenly it stops. It's possible the IoTaWatt restarted or it's DHCP lease expired and it got a different IP address. Some computers will simply never think to broadcast a new query, insisting on unsuccessfully retrying the old IP address forever.

23.3 How to make it work better.

It is recommended that you to assign a static IP to the IoTaWatt right after installation. You do this in your router and should assign an IP address that is not in the DHCP pool, so there is no opportunity for conflict. Your router associates the specified IP address with the MAC address of the IoTaWatt and always gives it that address during the DHCP handshake at startup. Write it down. If you subsequently find that you can't access via `iotawatt.local`, you can use the IP address by typing `HTTP://xx.xx.xx.xx` as a URL in the browser.

If you are reading this because you didn't assign a static IP and now can't access your IoTaWatt with `iotawatt.local`, you may need to turn off your computer and the IoTaWatt, restart your router, then restart your computer and the IoTaWatt.

23.4 What else?

Throughout this section, we have been using the name `iotawatt.local`. If you changed the name of the IoTaWatt using the Device setup of the configuration app, you would appended `.local` to that name. If you changed the name to *ttawatoi* you would use `ttawatoi.local`.

23.5 The last word

Fix the IP and write it down.