HANDBOOK

ESP VICTRON_ON_STEROIDS

Handbook ESP Victron_on_Steroids

TABLE OF CONTENTS

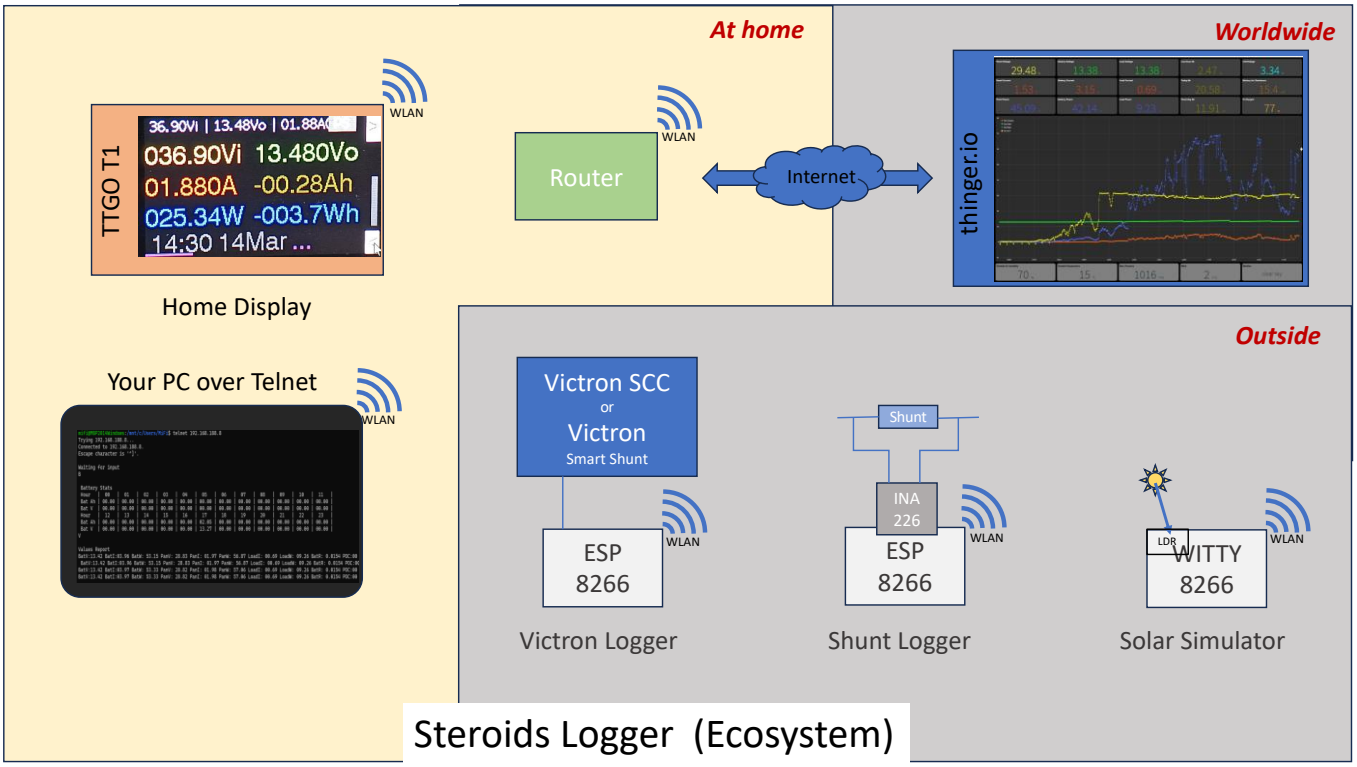
PRESENTATION

WHAT IS ESP VICTRON ON STEROIDS?

It is a powerful DIY open-source reporting solution for small solar systems.

In its simplest variant you just have either a solar installation with:

1. a Victron SCC with a VE output and you will add an ESP8266 module to log the data.
2. A MPPT-SCC of another brand and you have a shunt to the battery that is accessible.
3. You may even have no solar system at all and just want to play around with a simulation.



The system is already self-contained with a single ESP8266 module.
You don't need any additional computer, nor a gateway to access your dashboards worldwide on thinger.io.
If you want, you may access your ESP data using your computer over Telnet to get reports and view internal data.
Optionally, you may also add a home color display using a single TTGO-T1 module.

WHAT IS ESP VICTRON ON STEROIDS ***NOT***?

1. Branded to Victron. Despite its name, I am not affiliated to Victron.
   The project began as a logger for these devices therefore the name, but soon went vendor-agnostic as well.
2. Expensive. It is even damn cheap (the hardware for each option costs less than 15$)
3. Plug and play. You will have to buy ESP modules, solder a few wires, organize a casing by yourself.
   (only the solar simulator and the Home display don't need any soldering)
   I am not selling anything nor am asking for any financial contributions.
   I am happy to help as I can, but it's *your* project!
   You will have to download the software, configure the *configure.h* file to your needs, enter your credentials in *credentials.h*, get a free account at thinger.io and match the software to your dashboard there.
4. Fixed in its functionality. The solution is highly configurable, and I have written the open-source program using as much natural language as possible, avoiding programmers' gibberish.
   You are encouraged to contribute and modify it with your own ideas.
5. Undocumented. The code is fully on Github and the schematics are on Easy-EDA (a variant on KiCAD will be provided as well)
6. A dangerous mess of wires and cables. Everything communicates wirelessly and is self-contained, no ground loops must to be feared.
7. Using MQTT, Grafana these are common, but limited technologies necessitating gateways and extra configuration.
8. A trap into subscriptions. No external hardware is required, no computer, no gateway, no smartphone app with in-app payments. The (optional) thinger.io service is free for up to two devices and 4 dashboards and will ever be.
9. A good off-line solution. Albeit it can work without Internet, il will be very degraded. It needs also Internet to get the system time upon booting.
10. A good reporting solution for PWM controllers. You may use it to report battery values, but will not be able to report panel voltages.

N.B. in many examples given in that manual, I present ridiculously low solar power values:
It's because I live in a condo where I can't deploy solar panels as I want.
But the solution given is fully scalable to your needs.

HARDWARE

Depending on your solar configuration, here are the corresponding bills of materials required by following variants.

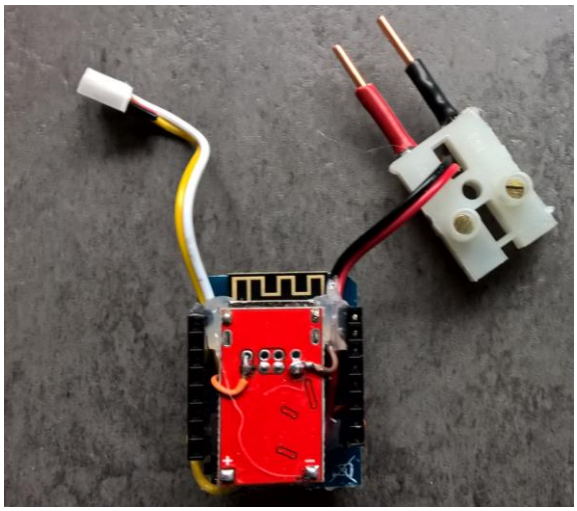
YOU HAVE A VICTRON SCC OR A VICTRON SMART-SHUNT WITH A VE INTERFACE:

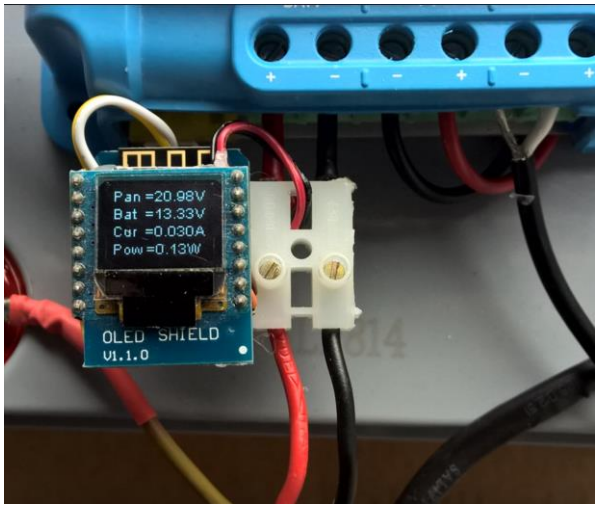
BILL OF MATERIALS

You will need:

1. A micro-DC-DC-converter
   low-voltage example: https://www.ebay.com/itm/285615099912
   high voltage example: https://www.ebay.com/itm/276165506723 *
2. An ESP8266 microcontroller: e.g. Wemos D1 Mini
   sourcing example: https://www.ebay.com/itm/276036540821
3. A Grove / JST micro PH, 2mm type 4 contacts pigtail cable to connect to the Victron
   Sourcing example: https://www.ebay.com/itm/155671812931
4. A two pole screw terminal and some wires.

Here on the picture my build (the screw terminal taps the battery voltage to the red DC/DC converter)





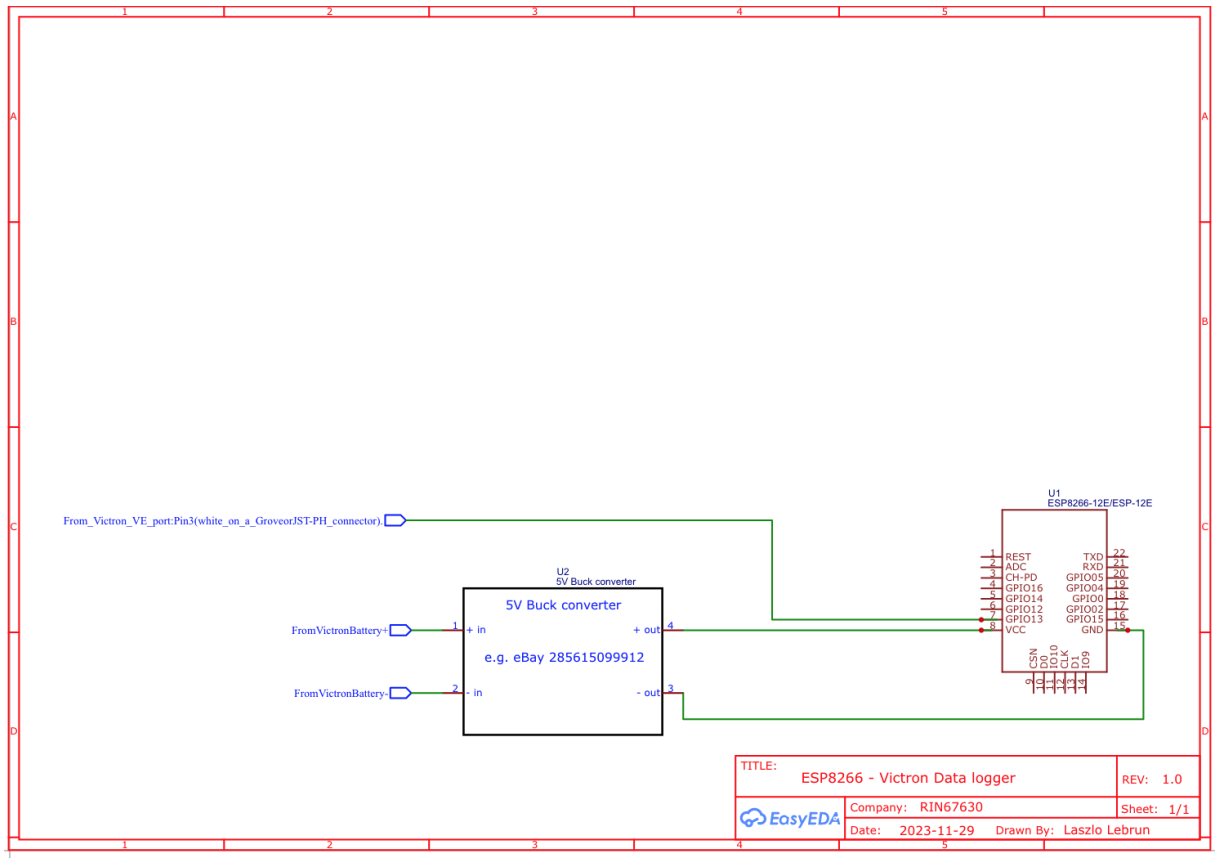
Optionally you may add a local display:

5. An OLED Shield for the Wemos D1 Mini

➢ That BOM is less than 10$, with display, add 3$.

**If your battery has a system voltage > 24V you must carefully choose your micro-DC-DC- converter to accept high input voltages, unless you can tap your battery in the middle to feed the DC-DC converter with less than 32V.*
You should also build the DC-DC converter away from the ESP8266 module and isolate it carefully to avoid the high voltage risk.If you have an inverter running 24/24, you may also just feed the ESP8266 from any USB charger.
You should not try to feed it from the 5V out of the VE interface, which cannot deliver enough power.

SCHEMATIC DIAGRAM:

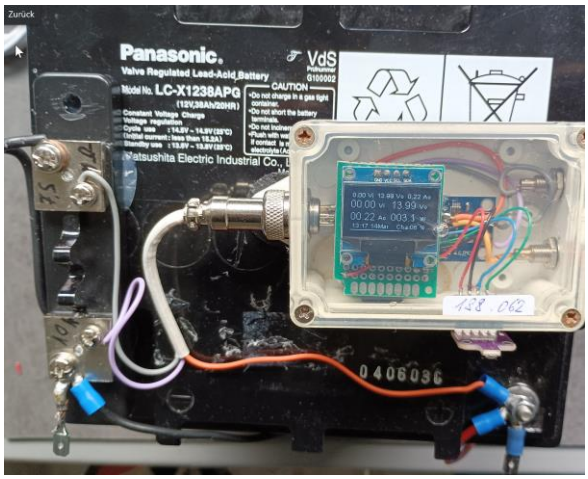


YOU HAVE A MPPT CONTROLLER FROM ANOTHER BRAND, SYSTEM BATTERY <= 24V

BILL OF MATERIALS

You will need:

1. A micro-DC-DC-converter
   low-voltage example: https://www.ebay.com/itm/285615099912
   high voltage example: https://www.ebay.com/itm/276165506723
2. An ESP8266 microcontroller: e.g. Wemos D1 Mini
   https://www.ebay.com/itm/276036540821
3. An INA226 Off-Rail Voltage/Current measuring module
   sourcing example: https://www.ebay.com/itm/404594569545
4. A small prototyping board
5. A shunt corresponding to your needs (which you may already have)
6. A small case, a pair of 3-pole connectors
   (I recommend the mini aviation connectors, cheap and good)
7. Here on the picture the build with a shunt glued on a battery and a pair of aviation connectors:



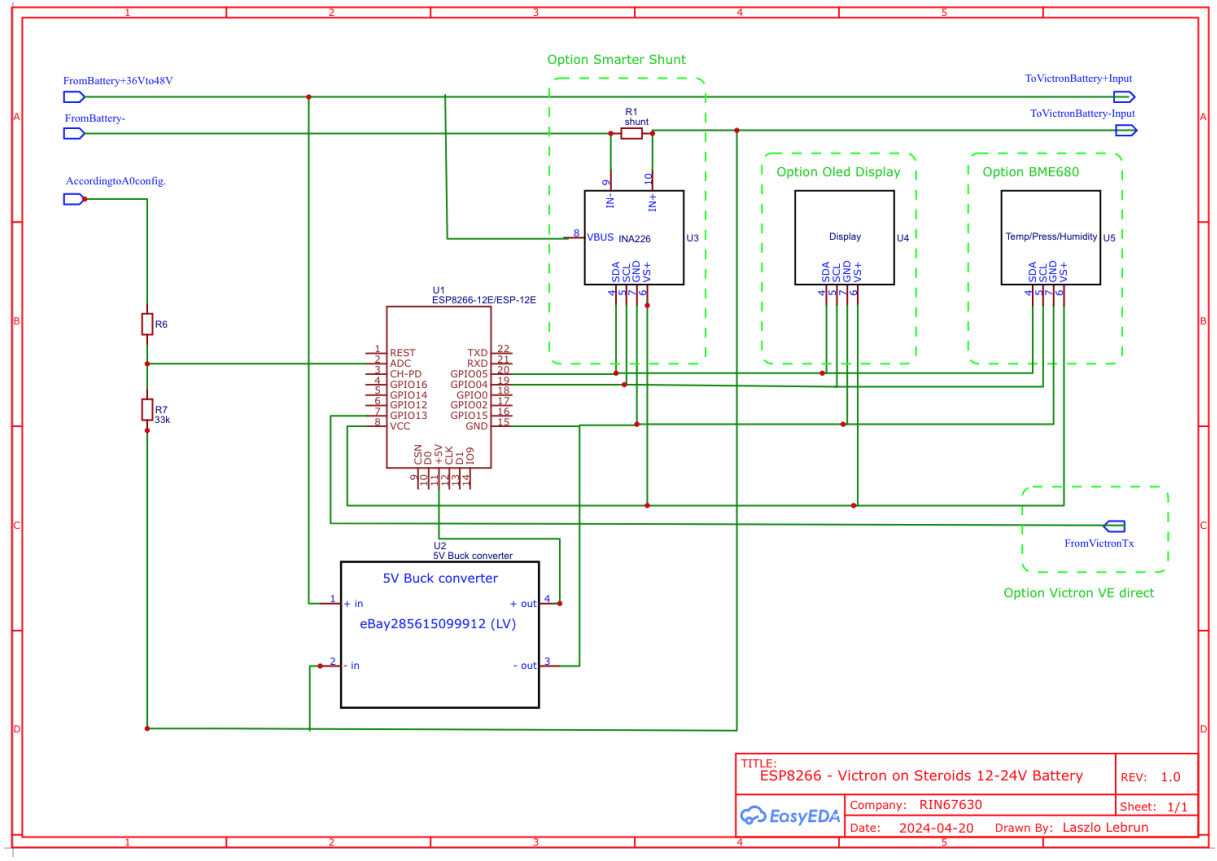
Optionally you may add a local display:

7. An OLED Shield for the Wemos D1 Mini
   (or here a bigger 128 x 64 OLED display on a prototype board and a BM640
   local weather sensor)

- That BOM is less than 15$ + shunt price, with a local display, add 3$.
  N.B. If you don't already have a shunt, you should measure the voltage between the (-) battery and the (-) panel terminal of your solar controller: frequently it's internal shunt is exactly connected there: if you measure a voltage of 0..80mV proportional to the full current to the battery, you don't need an additional shunt, just tap the signal from there.

➢

SCHEMATIC DIAGRAM:



YOU HAVE A MPPT CONTROLLER FROM ANOTHER BRAND, SYSTEM BATTERY > 24V:

BILL OF MATERIALS

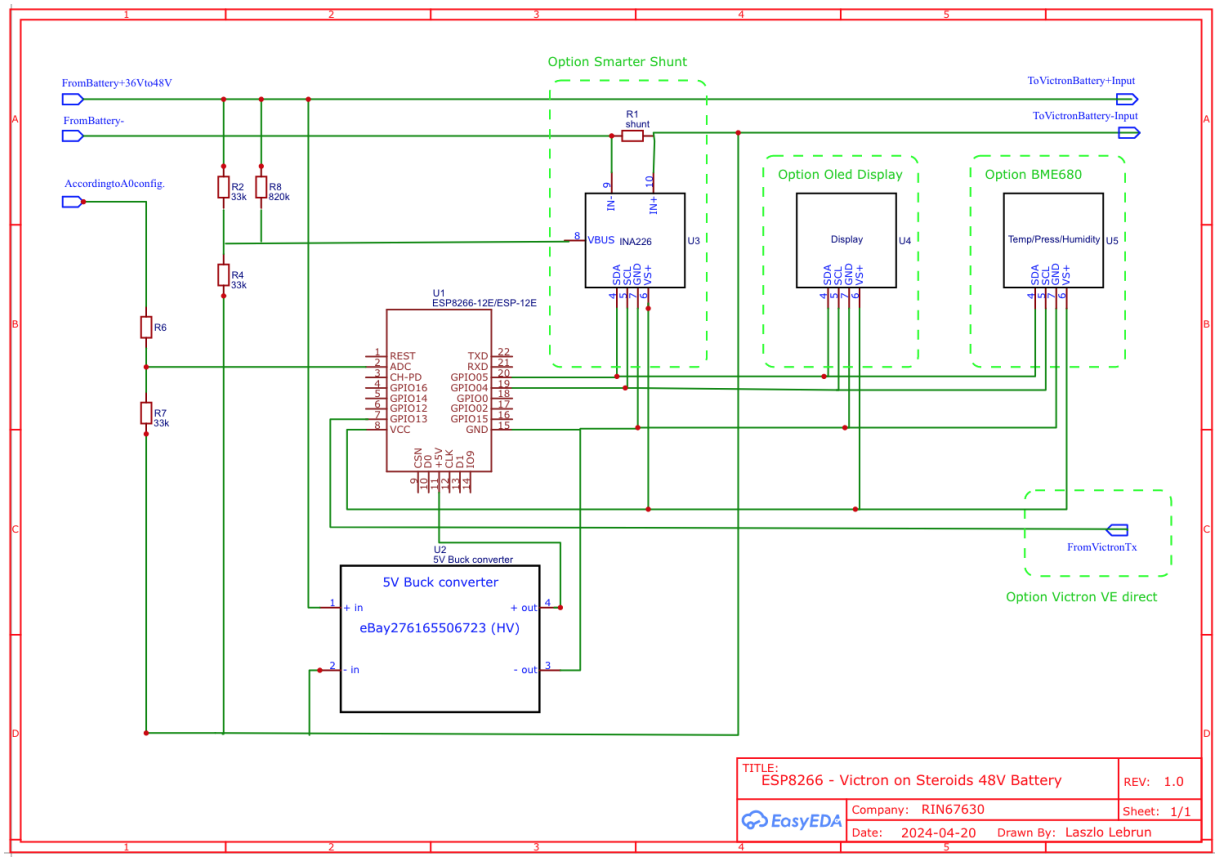
The bill of materials will be the same than above, but

You must carefully choose your micro-DC-DC- converter to accept high input voltages, unless you can tap your battery in the middle to feed the DC-DC converter with less than 32V. You should also build the DC-DC converter away from the ESP32 module and isolate it carefully to avoid the high voltage risk.

You will also need some resistors to build a voltage divider before the INA226 module and another divider for the panel voltage.

N.B. with panel voltages over 48V, I do highly recommend to split the resistor r6 in two resistors in series and to place the resistor connected to the panel voltage outside the PCB on a voltage sensing cable in order to keep high voltages away from the sensitive electronics.

SCHEMATIC DIAGRAM:



YOU HAVE A PWM CONTROLLER.

BILL OF MATERIALS

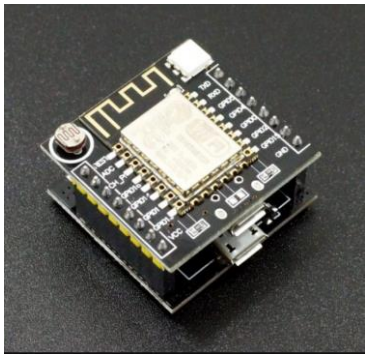
The BoM will be the same than the 12..24V version, you just cannot measure the panel voltage.
Expect your measurements to be quite noisy.

YOU JUST HAVEN'T YET ANY SOLAR SYSTEM AND JUST WANT TO PLAY AROUND

BILL OF MATERIALS

You will need:

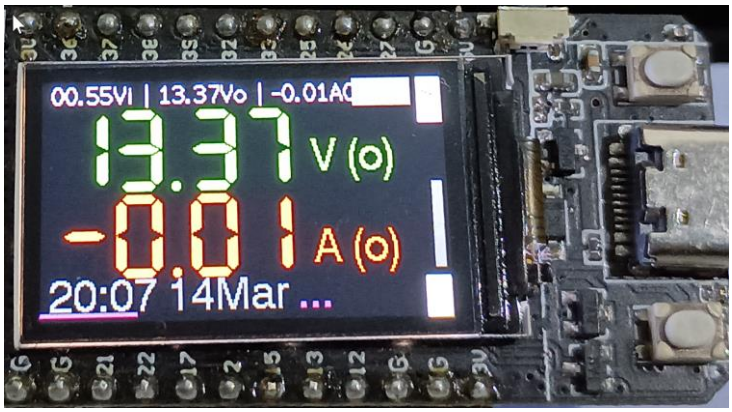
1. A Witty ESP 8266 module
   sourcing example: https://www.ebay.com/itm/304115464199
2. Any matching smartphone 5V charger that you have laying around in your drawer.



- The Witty module and its programming socket cost less than 3$

The Witty module includes a light dependent resistor and is therefore ideal to simulate a solar installation.
The sensor should be placed towards outside light, but not in direct sunlight, as it will be saturated.

YOU ALSO WANT A WIRELESS HOME COLOR DISPLAY



BILL OF MATERIALS

You will need:

- A TTGO T1 or LillyGo module:
  e.g. https://www.lilygo.cc/products/t-display-de
- Any matching smartphone 5V charger that you have laying around in your drawer.

It can be used as received, just with the USB charger.
You can order a plastic case for it there:
https://de.aliexpress.com/item/1005006258788885.html

There are also several projects on Github with code for a 3-D printer to fab the case yourself.

The LillyGo module costs less than 15€ (in Europe with VAT and customs, far less in USA)

SOFTWARE

INSTALLING THE IDE

Install Arduino IDE 2.0, if you don't already have it:https://www.arduino.cc/en/software

CONFIGURE THE IDE

In the menu Preferences, enter the address of the additional boards' files:

http://arduino.esp8266.com/stable/package_esp8266com_index.json, https://github.com/Heltec-Aaron-Lee/WiFi_Kit_series/releases/download/1.0.0/package_heltec_esp32_index.json, https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

Take note of your sketchbook location.

INSTALL THE BOARDS AND LIBRARIES ONTO THE ARDUINO IDE

BOARDS:

- esp8266 by ESP8266 Community
- esp32 by Espressif Systems (if you plan to use an ESP32 device)
- Heltec ESP32 Series Dev-boards by Heltec (if you plan to use a Heltec / LoRa device)

GLOBAL LIBRARIES:

- ESP8266 and ESP32 OLED driver for SSD1306 displays by ThingPulse, Fabrice Weinberg
- MobaTools by MicroBahner from which I took the Motobuttons.h lib
- Arduinojson by Benoit Blanchon <blog.benoitblanchon.fr>
- ThingerCore32 and Thinger.io by Alvaro Luis Bustamante
- TelnetStream by Juraj Andrassy
- InterpolationLib by Luis Llamas
- INA226 by Rob Tillaart (if you plan to use the vendor-agnostic shunt version)
- TFT_eSPI by Bodmer (if you plan to use the TTGO remote color display)
  N.B. this library must be configured specifically to run for the TTGO !
  Once installed, you must replace the file
  [your sketchbook folder] \libraries\TFT_eSPI\User_Setup.h" with the file User_Setup.h from
  https://github.com/rin67630/Victron_VE_on_Steroids/blob/main/Config_Files/User_Setup.h

PROJECT'S LIBRARIES:

We will place there the Hardware Abstraction Layer files containing the definitions for specific Victron modules and Shunt characteristics.

Under your sketchbook location, subfolder libraries create a subfolder named /steroids_HAL_files/.
Copy the files from https://github.com/rin67630/Victron_VE_on_Steroids/tree/main/Config_Files into that folder.

I have provided some content for Victron SCC devices:

- MPPT_75_10.h
- MPPT_75_15_0OLD.h
- MPPT_75_15.h
- MPPT_100_20.h
- MPPT_100_30.h

If your Victron device is not listed but has a VE port, open an issue, we will define together a corresponding file.
I have also provided some examples for some shunts:

- SHUNT_1A_25mV.h. (the default on the INA226 modules)
- SHUNT_10A_75mV.h
- SHUNT_10A_100mV.h
- SHUNT_500A_50mV.h

If your shunt is not listed, edit one of the existing shunt file e.g.:

```
// SHUNT_500A_50mV

// ***Electrical parameters***   (Adjust, if you use other shunts than R100)

#define SHUNT0   10000         // Nominal Shunt resistor value in microOhm (Channel 1 of the INA, connected to the battery)

#define AMPERE0   500          // Max expected intensity in operation

#define IFACTORB   1000000   // Battery Current correction factor 1000000 is normal,  -1000000 reverses shunt, change value to correct for wrong Amp values
```

Change the value of SHUNT0 and AMPERE0 to match you hardware and save the file with a corresponding new name.

GET THE ARDUINO IDE FILES FROM GITHUB

https://github.com/rin67630/Victron_VE_on_Steroids/tree/main/Software

From there press on the green Code button and chose Download Zip.
Store the file Victron_VE_on_Steroids-main.zip e.g. to the folder downloads. Unzip it.
Copy the folder Software to your sketchbook location.
Rename that folder with the name of the file beginning with "ESP_Victron_on_Steroids_V"

Start your Arduino IDE, open the corresponding sketch.

The IDE of Victron on Steroids will display many folders. It might feel unusual, but functionally splitting the code into functionally separate folders is extremely facilitating programming and increases the readability of the code. These folders are:

- ESP_Victron_on_Steroids_V04.2024.ino
  (the legal stuff and some useful info), no code there, only comments)
- Config.h
  (the most important file for you, here you define your options to tailor your system)
- Credentials.h
  (please enter here your secret credentials)

The next files contain the program code itself, only fiddle with it, if you know what you are doing:

- a_Libs Vars.ino
  (the libraries and variable declarations)
- b1_Net_Functions.ino
  (the –mostly immuable– network functions to run an online ESP)
- b2_User_Functions.ino
  (application specific functions)
- c_Setup.ino
  (the classic setup routine of Arduino-based code)
- d Menu.ino
  (the processing of the user menu (from Serial or over Telnet) also processing the buttons on modules with display buttons) , runs every 125mS.
- e1_Fast_Data.ino
  (fast data gathering processing, runs every 125mS)
- e2_SlowData.ino
  (slower data gathering processing, runs every 1S)
- f Stats.ino
  (slow extraction of statistics, runs every 1S)
- g_Display.ino
  (processes the information shown on the display, runs every 1S)
- h_Serial.ino
  (processes the information on the Serial or Telnet Reports, runs every 1S)
- i_Wireless.ino
  (processes the information sent to thinger.io or reports transmitted over UDP, runs every 1S)
- s Scheduler.ino
  (the main program; triggering all other modules and providing timing variables)

- t1_Frontpage_Thinger.h
  (this is not a c++ code, but the configuration information of the main dashboard at thinger.io stored here for convenience)

CONFIGURE THE SOFTWARE

CONFIG.H

Here you will match the generic program to your personal configuration

```
// N.B. Compile sketch with following board settings:
//- for option 8266:         LolinWEMOSD1 (Clone)
//- for option TTGO:         TTGO T1
//- for option HELTEC LoRa:    HELTEC WiFi Lora32  (Not V2 !) from HELTEC, not generic
//---------------------- HARDWARE OPTIONS ---------------------------------

#define CONTR_IS_ESP8266 //_IS_HELTEC, _IS_TTGO, _IS_ESP8266, _IS_ESP32;  Choice of ESP controller boards
Here you define which board you use: Wemos-D1Mini, Wemos8266, Witty, go under _IS_ESP8266,
ManyESP32 boards go under _IS_ESP32, the TTGO, LiliGO go under _IS_TTGO, the Heltec-LoRa board,
go under _IS_HELTEC.

#define SCREEN_IS_NONE    // _NONE , _64x48 , _128x64, _TTG0, _HELTEC, _WEMOS32 ;
If your board has no screen, use _IS_NONE,
if it has a 64x48Pixel Oled screen snapped on the WemosD1, use _IS_64x48,
if you are using a 64x128 Pixel Oled, either soldered yourself or onboard on a Wemos Board, use _IS_128x64,
if the board is a TTGO, use _IS_TTGO
if the board is a Heltec-LoRa board, use _IS_HELTEC.

#define SCREEN_IS_REVERSED // _IS_NORMAL, _IS_REVERSED To turn the display 180° if required
Normally _IS_REVERSED is the right setting, if you want the screen turned by 180° use _IS_NORMAL

#define BRIGHTNESS  255   // PWM value for default brightness with TTGO 0=totally dark;255=totally shiny
For TTGO only: if you want to reduce the brightness, enter any value lower than 255.

#define A0_IS_SIMUL      // _NONE , _DOUBLEBATTERY , _HALFBATTERY, _PANEL, _POT, _SIMUL
Here you define what the analog input A0 is used for:
_NONE   A0 I unused.
_PANEL. A0 is used to measure the Panel voltage
_DOUBLEBATTERY for batteries above 24V, the full battery voltage is measured over A0 the lower
  half battery voltage in measured by the INA226 module.
_HALF BATTERY A0 is used to measure the lower half of the battery.
_SIMUL A0 is the light sensor of a Witty module used to generate pseudo values in simulation.
_POT (not yet implemented, A0 is used for a set-point Potentiometer)

#define A0_MAX      38   // if A0 is used, define the voltage of the full range measure
Enter here the full range voltage of what A0 measures.

#define D0_IS_NONE       // _NONE , _RELAY1
_NONE is the default
_RELAY1 if you have written a routine to activate a relay on D0

#define D5_IS_NONE       // _NONE , _RELAY2
_NONE is the default
_RELAY2 if you have written a routine to activate a relay on D5
```

```
#define D6_IS_NONE       // _NONE , _RELAY3, _VE_BLOCK, _VE_START_STOP
_NONE is the default
_RELAY1 if you have written a routine to activate a relay on D6

#define D7_IS_VICTRON       // _NONE , _RELAY4; _VICTRON
_NONE is the default
_RELAY1 if you have written a routine to activate a relay on D6
_VICTRON must be entered, if  you use the VE interface

#include "MPPT_75_15.h"   // Type of Victron controller (Only relevant if D7_IS_VICTRON)
Enter here the filename corresponding to your Victron device. (the files defined in project's libraries)

#define INA_IS_NONE      // _NONE , _226
_NONE is the default for Victron use, Remote display or Simulation
__226 used for smart shunt variants

#include "SHUNT_10A_75mV.h" // Shunt Characteristics  (only relevant if INA_IS_226)
If you use a smart shunt varant, , enter here the filename corresponding to your shunt value.
(the files defined in project's libraries)

#define INA_VBUS_IS_FULL  // _FULL _HALF   (when high voltage schematic is used)
_IS_FULL is the default, if the smart shunt is measuring the half battery voltage, enter _HALF.

#define NUMBER_CELLS     4   // Number of cells in Battery
Define here the total number of cells in series of your battery.

#define AH_CELLS         100 // AH of the battery
Define here the total Ah rating of your battery.

#define TYPE_IS_LIFEPO    // _LIFEPO, _LEAD, _LIION
Define here the chemistry of your battery. (_IS_LIFEPO, _IS_LEAD, _IS_LIION

#define POC_IS_TABLE     // _TABLE, _VICTRON   defines if POC is estimated from a table or comes from Victron
_IS_VICTRON: the POC of your battery is read from the Victron interface (only if Victron provides it)
_IS_TABLE: the POC is estimated from internal tables
```

N.B. Enter the parameters exactly as shown in CAPITALS and with UNDERLINES where indicated.
Else the parameter will be ignored and the compilation may fail or give wrong results.

```
//---------------------- SOFTWARE OPTIONS -------------------------------

#define WEATHER_IS_OWM    // _NONE , _OWM , _BME680       // (Source of the Weather Information)
_IS_NONE: no weather service
_IS OWM:  the free internet-weather service of Open Weather Map will provide weather data.
  you will enter the city data and your user number in credentials
_IS_BME680 you have a local BME680 sensor option to provide weather data.

#define DASHBRD_IS_THINGER   // _NONE , _THINGER            // (Internet Dashboard)
_IS_NONE if you don't' use thinger.io to display your data over the internet (or if a display module makes the
connection)
_IS_THINGER if you use thinger.io with this module you will enter the device data and your user data in
credentials.

#define TERM_IS_TELNET    //  _TELNET, _SERIAL , _SOFTSER, _2SERIAL // defines where do Menus and Reports
occur: _SERIAL and D7_IS_VICTRON are mutually exclusive )
_IS_TELNET: if you use the menu over telnet (recommended)
_IS_SERIAL: if you use the menu over the serial port (cannot be used together with Victron VE on an ESP8266)
_IS_SOFTSERIAL to use the over the serial port on an ESP8266 together with Victron VE (not recommended).
_IS_2SERIAL to use the over the serial port on an ESP32 together with Victron VE

#define UDP_IS_SEND      // _NONE , _SEND , _RECEIVE     // (UDP Inter-ESP Communication)
_IS_SEND: communication between ESP modules, the field ES is the sender…
_IS_RECEIVER: …and the other one is the receiver.

#define ESP_UDP_ADDR      "192.168.188.85"       // (IP of the receiving ESP having UDP_IS_RECEIVE)
enter the IP address of the receiving ESP module (reported during boot)

#define ESP_UDP_PORT      4200  // (Port used to send/receive Values to other ESP)
leave the port as is, change only if you use another pair of ESP modules in the same LAN.

#define COM_IS_NONE      // _NONE , _HOURLY, _MIDNIGHT            // Periodical reports to computer
(the ESP can send periodical reports to a computer, which appends them to a file)
_IS_NONE: if you don't us that function
_IS_HOULY: if you want hourly appends to your computer
_IS_MIDNIGHT: if you want midnight appends to your computer

#define COM_UDP_ADDR      "192.168.188.96"            // (IP of the receiving computer for night reports)
enter the IP address of the receiving computer.

#define COM_UDP_PORT      4230                    // (Port used to send/receive Values to other computer)
leave the port as is, change only if your must use another port

#define DEVICE_NAME       "Witty"                 // Name of the device used as Hostname and at Thinger.io
define a unique name for your module (used in the Arduino IDE and by thinger.io

#define DEVICE_NUMBER     0                       // 0 if no ESP32 supervisor, else 1..4
currently nor yet used, leave 0



//-------------------------- WiFi Options ------------------------------------
```

```
#define WIFI_REPEAT        1000    //mS for retry
leave the port as is, change only if you use another pair of ESP modules in the same LAN.

#define WIFI_MAXTRIES      30
leave the port as is, change only if you use another pair of ESP modules in the same LAN.

#define WIFI_POWER         7.5
you can reduce WiFi power if your router is close or increase it if your router is distant.
1    (for lowest RF power output, supply current ~ 70mA,
20.5 (for highest RF power output, supply current ~ 80mA

//----------------------DO NOT EDIT until you know what you do ----------

#define SERIAL_SPEED     19200

#define LDR              A0

#define RELAY1           16   // D0  Relay or FET control 1

#define RELAY2           14   // D5  Relay or FET control 2

#define REDLED           15   // D8

#define BLULED           13   // D7 <-- Victron

#define GRNLED           12   // D6
```

CREDENTIALS.H

```
#define WIFI_SSID          "Secret"  // Change it your own SSID
Replace “Secret” with your SSID

#define WIFI_PASS          "Secret"  // Change it your own Password
Replace “Secret” with your Password

// ***Time zones***

#define NTP_SERVER "de.pool.ntp.org"   // Change de. to your own country
E.g. if you are in Canada enter “ ca.pool.ntp.org”

#define TZ          1             // (utc+) TZ in hours
Enter your time difference to UTC in hours.

#define DST_MN      60
Leave as is

#define GMT_OFFSET_SEC 3600 * TZ
Leave as is

#define DAYLIGHT_OFFSET_SEC 60 * DST_MN
Leave as is

//Thinger

#define THINGER_USERNAME          "Secret"  // Change it your own UserName
Change it your own UserName

#define THINGER_DEVICE_CREDENTIALS "Secret"  // Change it your own Credential
Change it your own Credential which will be matched to your device created in thinger.io

#define THINGER_DEVICE           DEVICE_NAME
Leave as is

#define INFLUXDB_URL            "https://eu-central-1-1.aws.cloud2.influxdata.com"
Not yet fully implemented, for Grafana users. Please ignore unless you want to contribute to a Grafana interface

#define INFLUXDB_TOKEN           "Secret"  // Change it your own token
Not yet fully implemented, for Grafana users. Please ignore unless you want to contribute to a Grafana interface

#define INFLUXDB_ORG            "your@email"
Not yet fully implemented, for Grafana users. Please ignore unless you want to contribute to a Grafana interface

#define INFLUXDB_BUCKET          "Steroids Bucket"
Not yet fully implemented, for Grafana users. Please ignore unless you want to contribute to a Grafana interface

//Location for weather
Open a free account at openweathermap.org, got ot https://home.openweathermap.org/api_keys
Create an API key for your location

#define OPEN_WEATHER_MAP_APP_ID     "Secret"    // Change it your own API Key
use the API key generated above.

#define OPEN_WEATHER_MAP_LOCATION_ID "Location ID"  // Change it your own Location ID
goto “Weather in your city” find your city, note the number in the URL il will be the location ID
```

```
#define OPEN_WEATHER_MAP_LANGUAGE    "en"
enter your language code

#define OPEN_WEATHER_MAP_UNITS       "metric"    //Americans: learn metric ;-)
enter metric, if you want US-imperial: search for the correct syntax yourself, I will not help you… 😉
```

COMPILE YOUR SKETCH

Once all data has been entered, you are almost ready to compile your sketch.
If you have chosen DASHBRD_IS_THINGER (the regular case) , you must before have defined the thinger.io device and buckets configuration (next chapter).

Before compiling, make sure your compiler is set to the right board type:

- For Witty, WEMOS D1 min, WEMOS8266 chose the board Wemos D1 (Clone).
- For WEMOS 32 chose ESP32 WROOM-DA Module
- For TTGO or Lilygo, chose ESP32 TTGO-T1
- For HELTEC-LoRa chose Heltec WiFiL oRa32 (not V2!)

For the first time, your board must be connected through its serial port, and you should activate the serial monitor (set to 19600 Baud).

You should get this kind of boot log:

```
17:01:10.810 -> Compiled from: D:\Activities\3_Maker\ESP_Victron_on_Steroids_V04.2024\c_Setup.ino
17:01:10.844 -> on Mar 12 2024 at 13:03:07
17:01:10.880 -> Victron/INA226 Logger at work: WEMOS Steroids Serial @ 19200 Baud
17:01:10.988 -> Connecting to GW-FM-7390 from Flash E (2645) wifi:sta is connecting, return error
17:01:12.012 -> Done: RRSI= -46dB, IP= 192 . 168 . 188 . 003
17:01:19.990 -> Got Epoch, Now: 18:01, 23 March 2024
17:01:20.025 -> Start OTA, Start UDP: 4200
17:01:20.510 -> Menu ready on Telnet
```

The important steps achieved must be the received IP Address and the time (Epoch).
If your device has a display some boot information is shown on the display too.

Note the IP address.

Once the device successfully accessed your local WiFi, you may choose to upload next versions “over the air” using WiFi.

You can also access your device menu over Telnet* wirelessly from your computer invoking:
telnet <the IP address of your device>

```
$ telnet 192.168.188.8
Trying 192.168.188.8...
Connected to witty.fritz.box.
Escape character is '^]'.

Waiting for input
```

If you get compiling errors, make sure that all libraries are present and your parameters are matching your hardware.

SETUP THINGER.IO

Setting up thinger.io is much easier to set up than using the conventional way with MQTT, Influx and Grafana. You do not have two take care of…

Everyone can register free of charge two devices to plot information and design dashboards in a very versatile way.

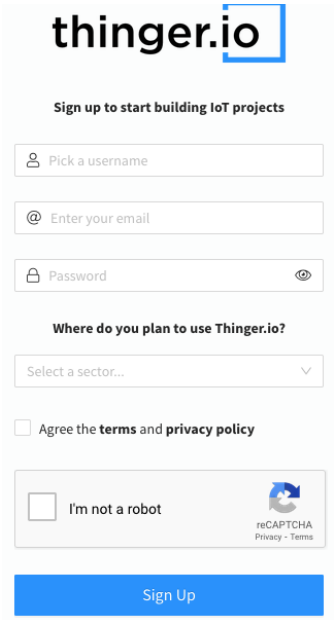
- Fast dashboards that built up trends over time, down to a one second pace.
- Historical dashboards that display information gathered by thinger.io over the last 24 hours.
- Date buckets that gathers periodical information over up to 365 days, from which you can download

CSV data to e.g Excel.

- Rules, to forward warnings over e.g. eMail.
- Interact with the device to drive a relay or initiate calibration
- and much more...

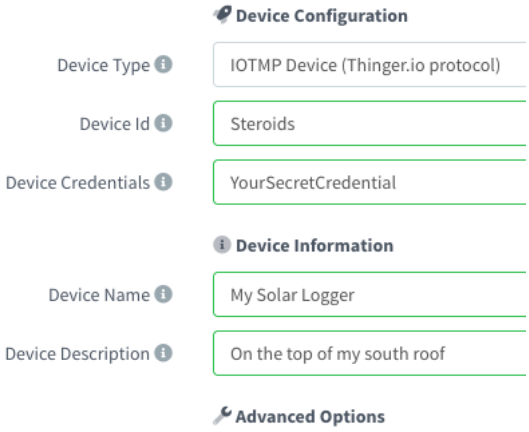
CREATE AN ACCOUNT

First create a free account at thinger.io:
https://console.thinger.io/login



SETUP A DEVICE

Set up device as follows :
https://console.thinger.io/#!/console/devices/add



Remember this information, as well as your username and password.

The device ID MUST match the device name of your sketch.

RESOURCES AND PROPERTIES

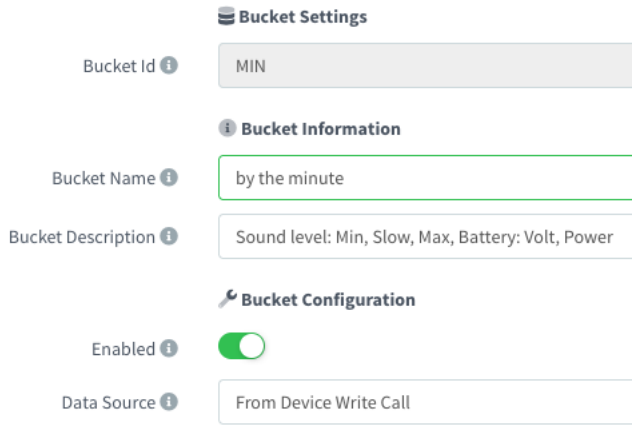
Within a device you have

- “Resources” written by the device or polled by a dashboard
- “Properties” written /read back by the device or polled by a dashboard

containing the current values of your devices.
These are defined by the program, you don’t have to create them manually.
Properties have the advantage over resources that they store values permanently and can be retrieved by the program as would an external memory or an EEPROM do.

BUCKETS

Additionally, across devices, you have the so-called buckets, used to store long-term data for reporting and to draw timeseries charts. You have to create the buckets manually for your project.



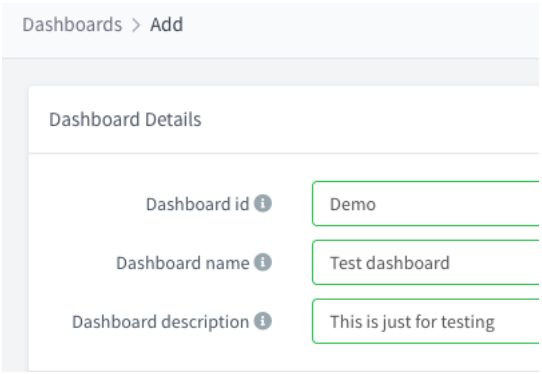
Set up the following “data buckets”

- MIN. used for Minute recording
- EVENT used for asynchronous events like Battery low, Battery full,
- DAY used for Daily report – Ah summary, Battery voltage summary
- HOUR used for Hourly Data battery values, Weather

The bucket ID must match the 4 given names, the bucket name and bucket description can be freelyy chosen..

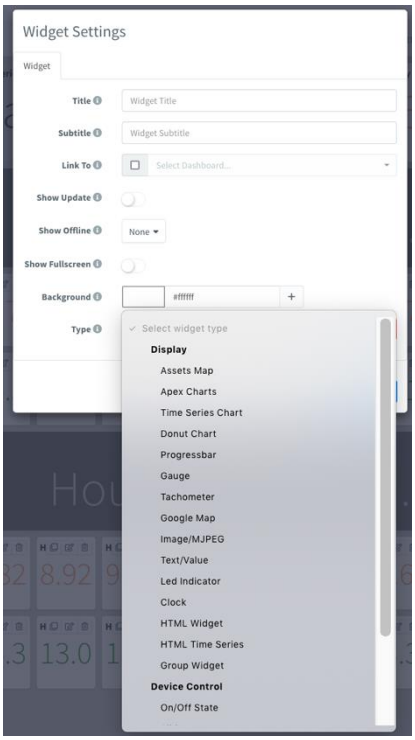
DASHBOARDS

With a free account, you may create up to 4 dashboards.



In a dashboard you place “widgets”. You can add / change remove widgets any time, right from the dashboard by entering the configure mode at the top right of the dashboard.

The widgets will then show to become duplicated, configured, deleted



The configuration of a complex dashboard with many widgets is easy but can be time consuming.

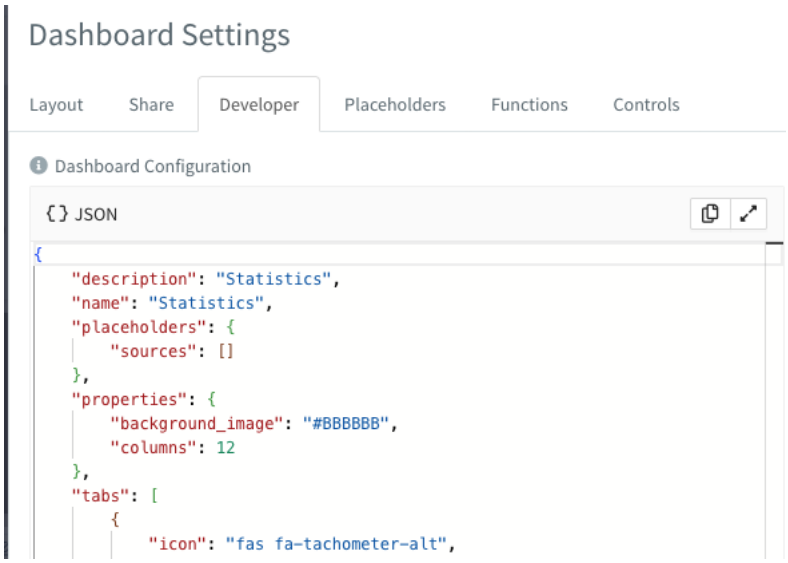
PRECONFIGURED DASHBOARD

To start with ease, I am providing preconfigured dashboards in the Arduino IDE.
of Victron_on_Steroids.
The file <t1_Frontpage_Thinger.h> contains a preconfigured dashboard.

You must however first adjust the device name and your user name before using it.
Using the editor of Arduino IDE, replace every occurrence of <"id": "Steroids">, with your device name.
(You may skip that, if you left your device with the default name “Steroids”)

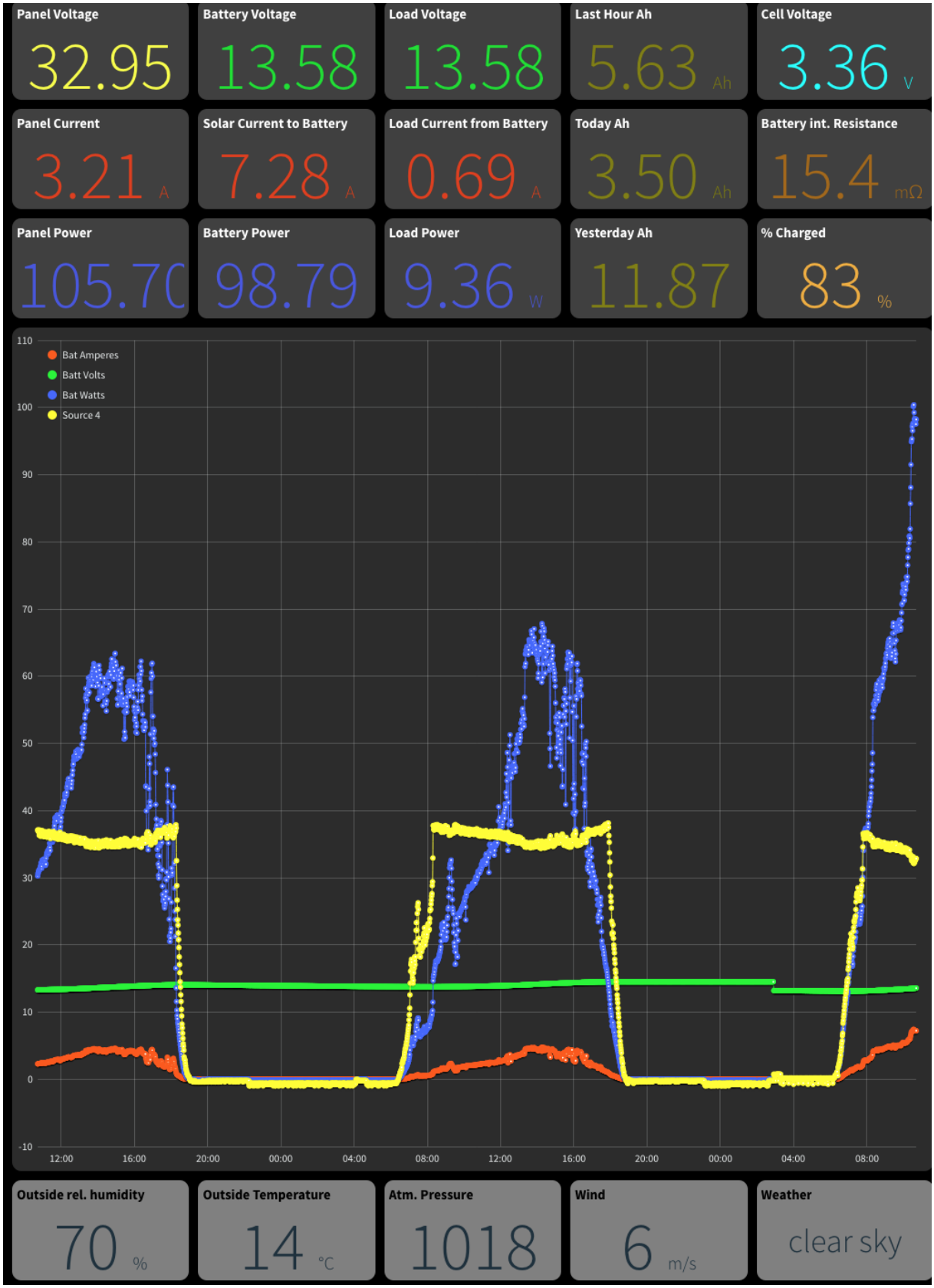
Then replace every occurrence of "user": "rin67631" with your own user name.
select all text of that file and place it into your clipboard.

In https://console.thinger.io/console/dashboards open your newly created dashboard.
click on Settings,
then on Developer.

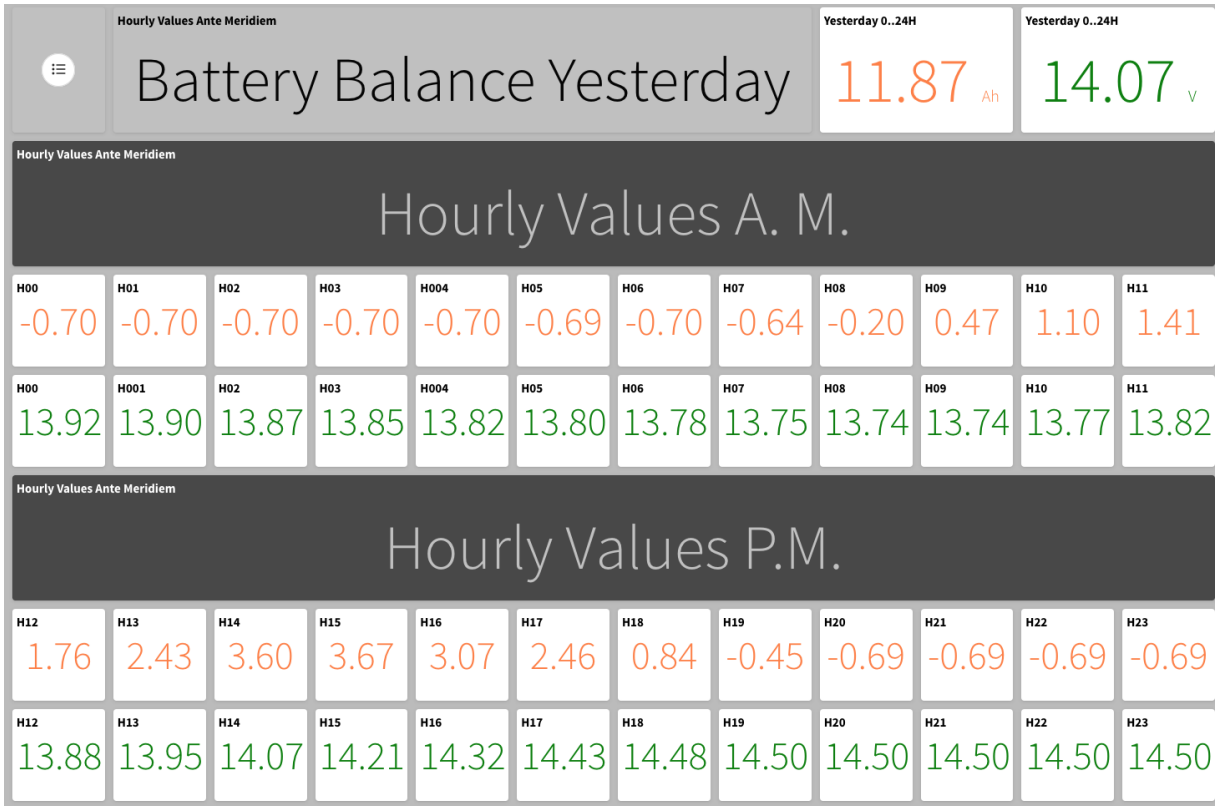


replace the text in the frame JSON with the content of your clipboard.
Save.

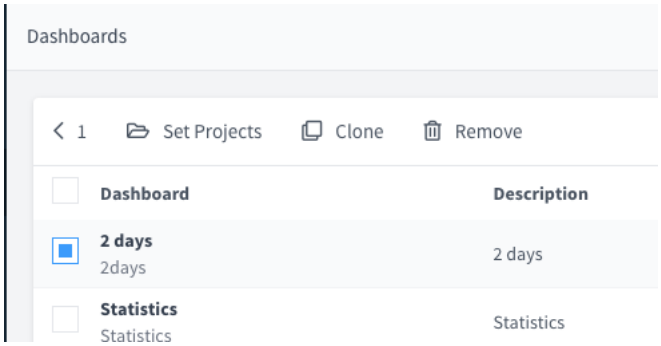
Admire your new dashboard:



You can do the same with the next dashboard preset: t2_Statistics_Thinger.h:



We will then can clone the first dashboard and modify its tine series charts to show fast instant values.

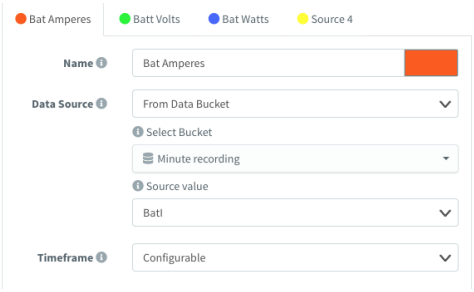


Press on the tab Clone and give a new name.
Open the clone.

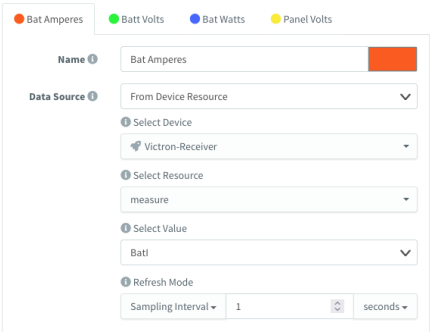
enter the configuration mode  at the top right of the dashboard.

go to the configuration settings of the time series chart  and click in the middle

modify the content of



into



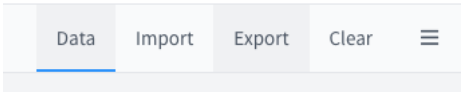
Go from top to bottom and repeat with the green BatVolts, the blue Bat Watts, and the yellow PanVolts

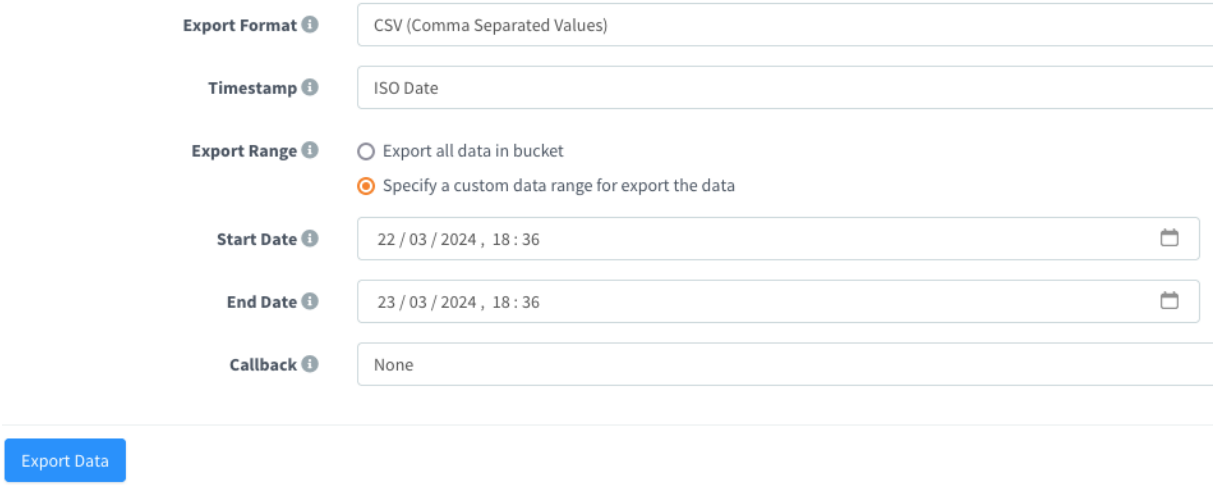
You now have a new dashboard getting its values from fast device resources instead of from the minute bucket.
That should demonstrate how easy and versatile thinger.io is.

Download Bucket data

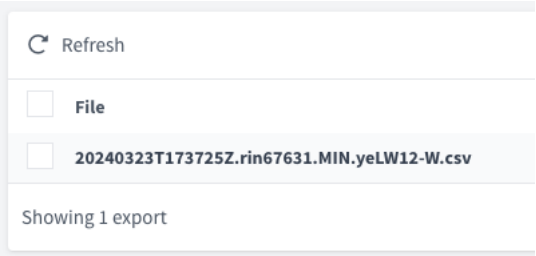
You can download the data from the buckets in csv format to be reused e.g. in Excel:

Open the bucket an click on <export>:





Enter the start and end dates then push <export>
Refresh the export list and click on the .csv data.
you can then open the CSV in your favorite spreadsheet or database.



TELNET/SERIAL MENU

You can use your computer to access the device menu over Telnet or the serial port:
telnet <your device's IP> or using the serial monitor

```
$ telnet 192.168.188.8
Trying 192.168.188.8...
Connected to witty.fritz.box.
Escape character is '^]'.

Waiting for input
```

The menu options are invoked with a single character:

V. display the regular device values every second

```
Values Report
BatV:13.70 BatI:00.08 BatW: 01.05 PanV: 01.14 PanI: 00.99 PanW: 01.12 LoadI: 00.69 LoadW: 09.45 BatR: 0.0154 POC:91.3
BatV:13.70 BatI:00.08 BatW: 01.05 PanV: 01.14 PanI: 00.99 PanW: 01.12 LoadI: 00.69 LoadW: 09.45 BatR: 0.0154 POC:91.3
BatV:13.70 BatI:00.08 BatW: 01.05 PanV: 01.14 PanI: 00.99 PanW: 01.12 LoadI: 00.69 LoadW: 09.45 BatR: 0.0154 POC:91.3

BatV:13.70 BatI:00.08 BatW: 01.05 PanV: 01.14 PanI: 00.99 PanW: 01.12 LoadI: 00.69 LoadW: 09.45 BatR: 0.0154 POC:91.3
```

D display the debugging device values every second

```
Debug Report
A0: 21 BatV:13.70V BatI:0.077A CellV: 3.423V BatPoC:91.3% IOhm:0.0154R State: Off
A0: 21 BatV:13.70V BatI:0.077A CellV: 3.423V BatPoC:91.3% IOhm:0.0154R State: Off
A0: 21 BatV:13.70V BatI:0.077A CellV: 3.423V BatPoC:91.3% IOhm:0.0154R State: Off
```

J (upper case) Display the job execution time monitoring
This shows the current execution time in mS, for each periodical job 100ms being the absolute maximum bearable. The second column displays the maximum until reset.

```
Job Timing

Job Durations(mS) Current - Max
Sche:001 - 001
Fast:000 - 004
Slow:007 - 012
Stat:000 - 298
Disp:000 - 001
Seri:-1000 - 004
Wifi:001 - 002

Job Durations(mS) Current - Max
Sche:000 - 001
Fast:000 - 004
Slow:008 - 012
Stat:000 - 298
Disp:000 - 001
Seri:-998 - 004
Wifi:001 - 002
```

j (lower case) resets the job execution time monitoring.

W display the weather list

```
Weather list :
Temp: 6.1°C Hum: 77.0% Press: 1009mBAR WindSpeed: 8.8m/s Direction: 310°N Clouds: 75% Summary: broken clouds
```

Z resets the device: it will reboot. This cuts the Telnet transaction.

Q Quits the Telnet session