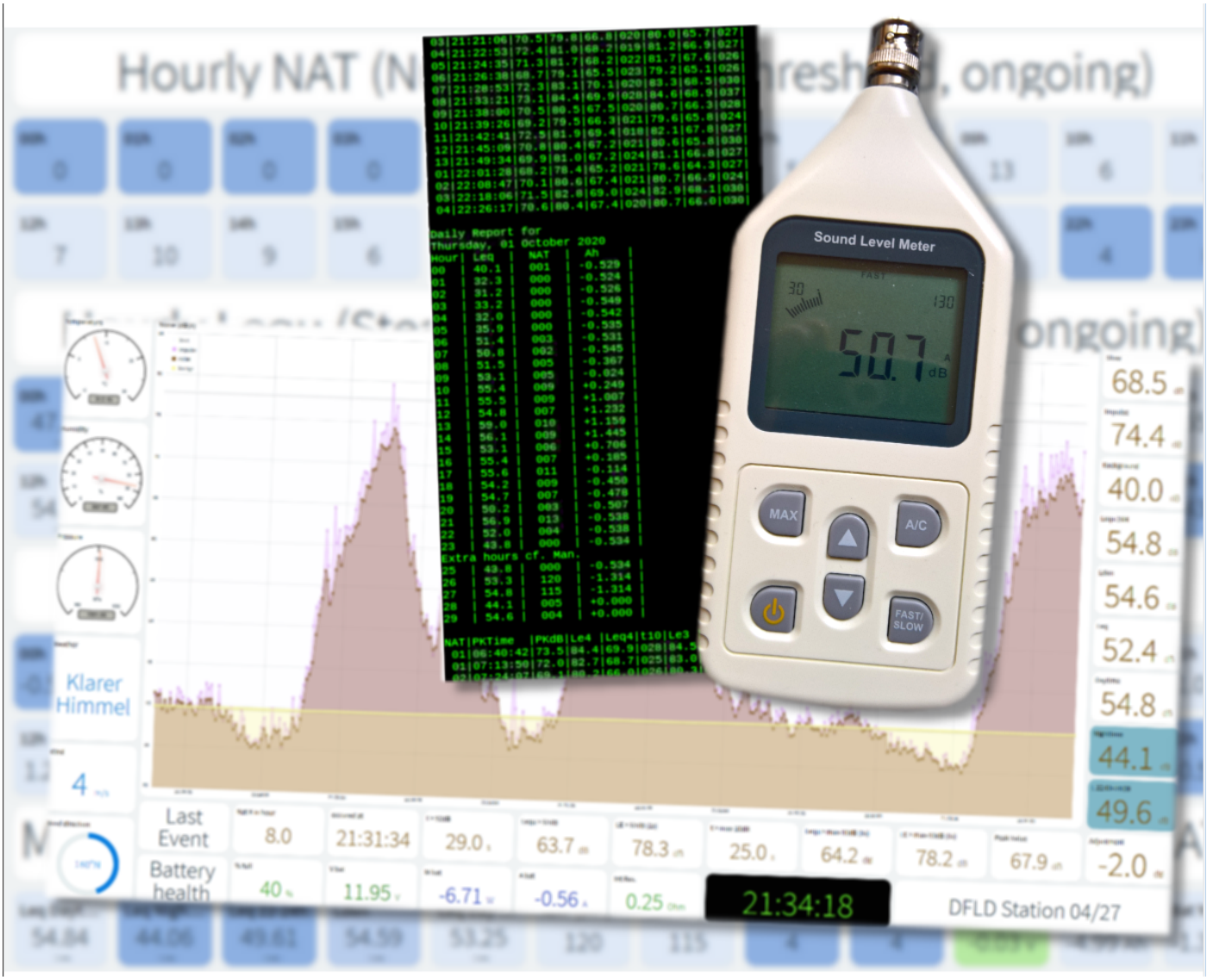
Networking Enhancement

for a Sound Level Meter

(the Swiss Army Knife)☺



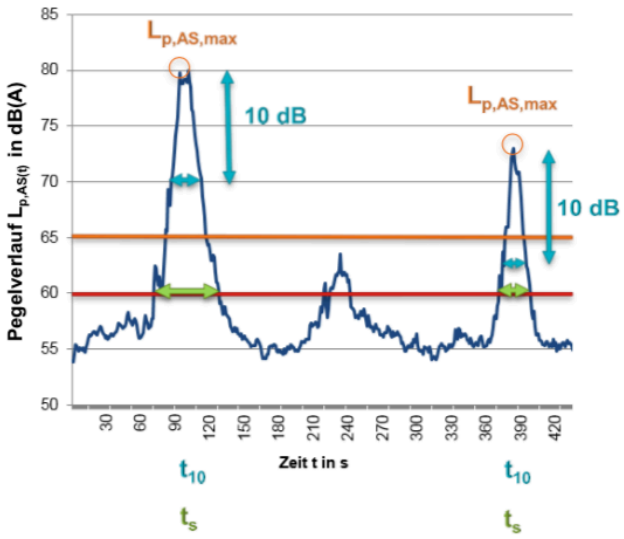
Digest

There is a large choice of sound pressure level meters on the market, from pretty cheap to awfully expensive ones. The cheap ones have frequently a sufficient accuracy for many purposes (albeit not being suitable for a legal enforcement). Most of them have however either no, or extremely primitive reporting abilities.

The purpose of this development is to provide networking functionalities to various consumer-grade SPL meters and deliver advanced acoustic metrics as close as possible to residential aircraft noise monitoring requirements as of: IEC 61672-1:2013 and DIN 45643 and
<https://www.ecac-ceac.org/documents/10189/51566/01.+Doc29+4th+Edition+Volume+1.pdf/bfde6e09-b46b-44e1-b73f-388fc3527aaf>

This is done by adding an ESP8266 WiFi microcontroller to it, costing about $4,-

The modification will provide USB and WiFi connectivity and be programmable to do the coolest things that only very expensive high-end devices offer: advanced noise figures, time related aggregations and event recognition, using fixed thresholds or/and floating thresholds according to ECAC standards:



The software takes care of possible strong fluctuations of the noise level and integrates together within the analysis time consecutive crossings of the limits.

The given program as highly configurable, intensively documented and can easily be adapted to other residential noise monitoring requirements like
- road noise
- railroad noise
- public events and so on..

The sound pressure is evaluated every 125 ms according to the most stringent requirements and processed to provide the following time response standards (simultaneously):

- Fast (Attack t=125mS, Decay t=125mS)
- Slow (Attack t=1S, Decay 4,3dB /sec)
- Impulse (Attack t=125mS, Decay 2,9dB /sec)
- Real peak value by the minute (125mS resolution, not the maximum of readings)
- Background level (t=2000s, excluding events over threshold)

The processing and the statistics occurs using a mathematically correct sound energy averaging according to the equation:

$$L_{p,\mathrm{A,eq},T} = 10\ \mathrm{lg}\left(\frac{t_0}{T}\sum_{i=1}^{N} 10^{L_{p,\mathrm{A,E},i}/10\ \mathrm{dB}}\right)\mathrm{dB}$$

The software provides following metrics according to residential aircraft noise standards:

Instant	exposure/time-averaged/cumulative			single event
Every second	By the minute*	By the hour*	By midnight*	Event triggered*
All above mentioned values (polled).	Leq	Leq	Leq 24h	Peak value
	Maximum in the minute	Number above fixed threshold (NAT)	Leq Night (22:00 - 06:00)	Leq relative to fixed threshold
	Minimum in the minute	meteorological conditions from openweathermap.org	Leq Day (6:00 - 22:00)	Leq relative to variable threshold max-10dB
	Background Noise		Leq (22:00 -24:00)	LE relative to fixed threshold
			Lden	LE relative to variable threshold max-10dB
			NAT Night (22:00 - 06:00)	Time above fixed threshold
			NAT Day (6:00 - 22:00)	Time. above variable threshold max-10dB
			NAT (22:00 -24:00)	Time of the event maximum
			Leq for every hour	
			Nat for every hour	

* Possible 365 Days long-term storage and Excel export

The software provides reporting and plotting over the serial port and/or dashboards over the free cloud service thinger.io

Back to the roots of the whole project, the software can optionally interface to www.dfld.de / www.eans.net, a residential network of more than 600 airport measuring stations providing over 15 years of long-term data and statistics. (a gateway (e.g. a Rasperry Pi) will be required).

Additionally the program grabs weather information from openweathermap.org and provides the corresponding meteorological conditions, inhibiting event recognition and the related false-positives in case of excessive wind.

Hardware

Besides the software itself, I will document several low-budget hardware solutions, from fairly easy with only three wires to solder until a complete concept of outdoor hardened device with monitored off-grid solar powered energy.

The used sound pressure level meter must deliver an analog SPL signal in the range: 0..1 V or 0..2,5 V, linearly progressive to dB sound pressure.
A calibration procedure at two fixed points 94dB and 47dB will ensure that the input measuring range (incl. evtl. An offset) is in line with the device DC output signal.

In parallel and with additional hardware, you can activate a complete solar battery management functionality.
(This optional functionality will be described later in Annexes)

Menu System

The program can output dialogss of different types.

For selection, the software offers a simple command menu consisting of individual characters that are completed with [return] via the serial USB line or now preferably via a wireless Telnet connection.

Device management

'Z': Reset ESP device
'C': Apply 94dB calibration 'c': Apply 47dB calibration /
'U': recall 94 & 47dB default settings
'+': Increase offset by 1 dB '-': Reduce offset by 1 dB
'?': List parameters
'~': List settings for WLAN & radio parameters.

These commands can be arranged in any order: You can enter multiple characters, then press [return]; these commands are executed immediately one after the other: Example of this: U +++ means: "Recall default settings of 94 and 47 dB and increase the offset by 3 dB.

Reports

'A': //serialPage AK
(this is not a printable report, it issues one byte every second to feed the DFLD website)

'P': //Periodical Reports on 'p': //Periodical Reports off
Options for periodical reports:

'D': //Day Report	'd': //no Day Report
'H': //Hour Report	'h': //no Hour Report
'M': //Minute Report*	'm': //no Minute Report (Battery)
'S': //Second Report*	's': //no Second Report (Noise)
'E': //Event Report	'e': //no Event Report

* these reports are designed to produce serial Plotter compatible results.

Example1 : PDHmsE means: Print Daily, Hourly, no minute, no second, Events
Example 2: p means: stop printing reports.
Example 3: P means: resume printing reports with last options
Example 4: Sd means: now with Second reports without Daily reports.

One shot reports

n.b. these reports stop periodical reports, resume with "P" to return to periodical printing.

'L': // Leq Report by 24h
'N': // NAT Report by 24h
*'B': // Battery Report by 24h **
*'b': // Battery Report (Actual measurements) **
'W': // Weather report

Report examples

Minute Hour Day and Events

Menu command MHEP:

```
08:55:59.490 -> Bat_Volt-10:1.928 Bat_Watt:-0.593 Bat_Level:7.200
08:56:59.461 -> Bat_Volt-10:1.927 Bat_Watt:-0.593 Bat_Level:7.200
08:57:59.462 -> Bat_Volt-10:1.926 Bat_Watt:-0.593 Bat_Level:7.200
08:58:59.502 -> Bat_Volt-10:1.925 Bat_Watt:-0.593 Bat_Level:7.200
08:59:59.489 -> BatAhBat:0.000 A0dBLEQ:58.8 WindSpeed:0 Direction:1072693248
08:59:59.536 -> Bat_Volt-10:1.924 Bat_Watt:-0.593 Bat_Level:7.200
09:00:59.478 -> Bat_Volt-10:1.923 Bat_Watt:-0.593 Bat_Level:7.200
09:01:59.466 -> Bat_Volt-10:1.922 Bat_Watt:-0.593 Bat_Level:7.200
09:02:47.466 -> PKTm: 09:01:37 PKdB:52.8 ATdB: 50.2 ATsec:48 PK-10dB: 49.7 PK-10sec: 41 NAT:1
09:02:59.489 -> Bat_Volt-10:1.921 Bat_Watt:-0.593 Bat_Level:7.200
09:03:59.488 -> Bat_Volt-10:1.920 Bat_Watt:-0.593 Bat_Level:7.200
```

Menu command L:

```
Leq : for Saturday, 08 August 2020
 Hour   |  00  |  01  |  02  |  03  |  04  |  05  |  06  |  07  |  08  |  09  |  10  |  11  |
 Leq dB | 58.1 | 58.3 | 58.3 | 58.4 | 58.4 | 58.5 | 58.6 | 58.7 | 58.8 | 61.1 | 65.4 | 65.8 |
 Hour   |  12  |  13  |  14  |  15  |  16  |  17  |  18  |  19  |  20  |  21  |  22  |  23  |
 Leq dB | 66.0 | 66.2 | 66.3 | 62.9 | 63.2 | 63.6 | 63.8 | 54.0 | 56.3 | 56.6 | 57.3 | 57.8 |
```

Menu command N:

```
NAT : for Saturday, 08 August 2020
 Hour   |  00  |  01  |  02  |  03  |  04  |  05  |  06  |  07  |  08  |  09  |  10  |  11  |
 NAT    |  00  |  00  |  01  |  00  |  01  |  00  |  00  |  00  |  00  |  01  |  24  |  23  |
 Hour   |  12  |  13  |  14  |  15  |  16  |  17  |  18  |  19  |  20  |  21  |  22  |  23  |
 NAT    |  20  |  21  |  19  |  06  |  12  |  09  |  34  |  12  |  13  |  01  |  01  |  00  |
```

*Menu command B:**

```
Battery History :
 Hour   |   00   |   01   |   02   |   03   |   04   |   05   |   06   |   07   |   08   |   09   |   10   |   11   |
 Bat Ah | -0.045 | -0.047 | -0.048 | -0.048 | -0.048 | -0.048 | -0.048 | -0.049 | -0.049 | +0.000 | +0.000 | +0.000 |
 Hour   |   12   |   13   |   14   |   15   |   16   |   17   |   18   |   19   |   20   |   21   |   22   |   23   |
 Bat Ah | +0.000 | +0.000 | +0.000 | +0.000 | +0.000 | +0.000 | +0.000 | +0.000 | -0.044 | -0.044 | -0.044 | -0.044 |
```

** with battery management, see Chapter x*

Ongoing reporting:

You can enable ongoing event + midnight summary reporting:

NAT: (Number of Above Threshold in hour;
PKTime: (Peak Time of the event)
Leq4: (Level equivalent for the time defined by max-10dB to max-10dB on the other side)
t10: (time defined by max-10dB to max-10dB on the other side)
Leq3:: (Level equivalent for the time above threshold)
t AT: (Time above threshold)

```
NAT|PKTime  |PKdB|Leq4|t10|Leq3|tAT|
…
 15|21:52:29|74.7|70.5|026|70.1|057|
 01|22:01:33|74.6|71.2|017|70.3|043|
 02|22:03:44|71.2|67.3|019|66.7|045|
 03|22:13:46|72.3|69.2|026|68.9|055|
 04|22:16:25|71.8|69.3|021|68.8|048|
 05|22:28:19|69.3|66.7|024|66.7|048|

Daily Report for
Tuesday, 15 September 2020
Hour| Leq  |  NAT  |  Ah    |
00  | 35.6 |  000  | -0.427 |
01  | 33.2 |  000  | -0.391 |
02  | 28.8 |  000  | -0.385 |
03  | 29.3 |  000  | -0.399 |
04  | 32.3 |  000  | -0.388 |
05  | 38.8 |  000  | -0.379 |
06  | 49.8 |  005  | -0.379 |
07  | 54.1 |  007  | -0.025 |
08  | 53.1 |  006  | -0.290 |
09  | 61.2 |  020  | -0.219 |
10  | 57.2 |  005  | +0.116 |
11  | 61.8 |  012  | +1.050 |
12  | 56.9 |  006  | +1.594 |
13  | 57.4 |  008  | +1.824 |
```

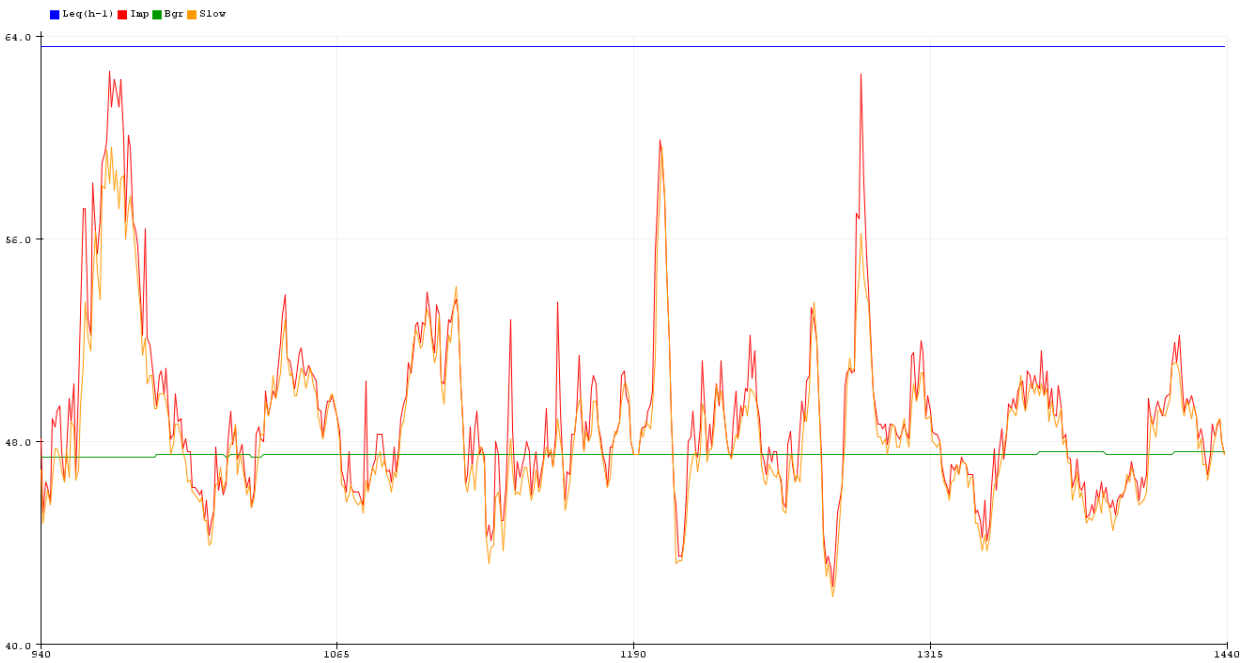
```
14  | 59.4 |  010  | +1.392 |
15  | 57.6 |  007  | +1.080 |
16  | 54.2 |  005  | +1.091 |
17  | 54.6 |  007  | +1.023 |
18  | 58.3 |  008  | +0.290 |
19  | 56.2 |  012  | -0.142 |
20  | 54.1 |  005  | -0.494 |
21  | 59.8 |  015  | -0.461 |
22  | 53.9 |  005  | -0.430 |
23  | 31.8 |  000  | -0.407 |
Extra hours cf. Man.
25  | 31.8 |  000  | -0.407 |
26  | 56.0 |  143  | +4.651 |
27  | 57.6 |  138  | +4.651 |
28  | 45.2 |  005  | +0.000 |
29  | 56.9 |  005  | +0.000 |

NAT|PKTime  |PKdB|Leq4|t10|Leq3|tAT|

 01|05:58:09|72.0|68.0|019|67.5|044|
 01|06:07:10|71.7|69.0|028|68.8|059|
 01|07:05:42|61.1|58.7|018|59.0|031|
...
```

These reports will be standard wise routed over the Serial USB connection to the serial monitor of the Arduino IDE.

They can also be configured to be issued
- to Serial1 on pin D4 (GPIO2) and from there be forwarded to e.g. a thermal receipt printer or over an UART-USB converter to another USB-port of a host computer for reporting.
- to a UDP network port and be dumped on the host computer in a log file.

Graphical reporting:

over the menu command "ESP" (nomen est omen) you can issue a one second paced report that provides data in a format suitable for serial plotting.
Over the Ardunio Serial Plotter you can get a graphical output of the noise evolution history:



The plotter and reporting abilities of the Arduino IDE are however, limited and you can only have one output at a time.

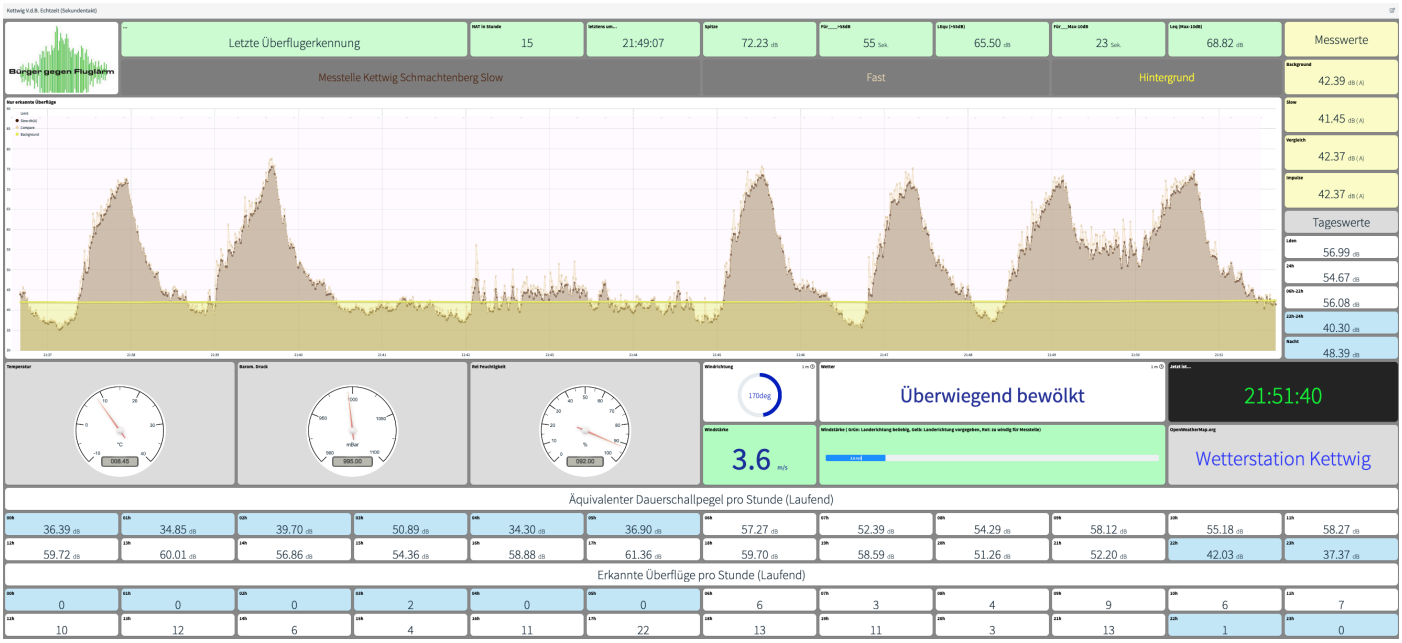
Cloud service Thinger

The most powerful reporting abilities are however achieved upon using the free cloud services of http://thinger.io :

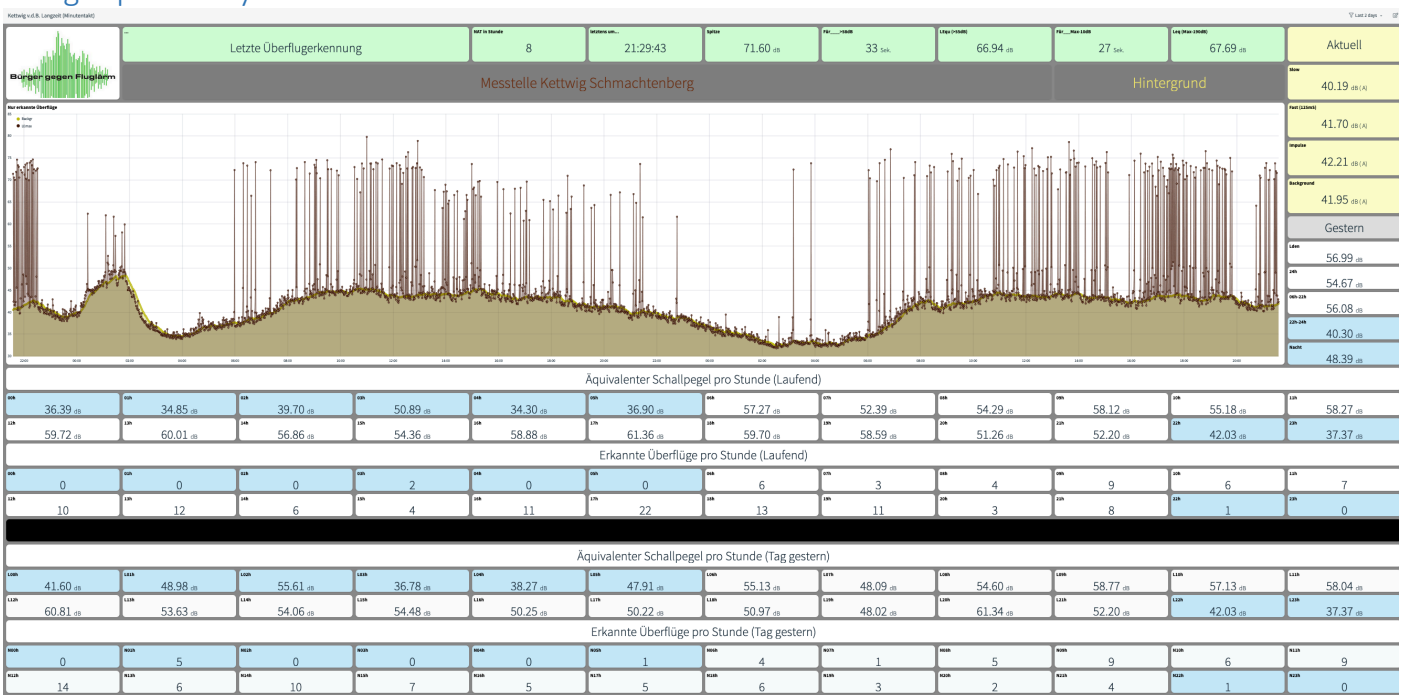
Everyone can register free of charge two devices to plot information and design dashboards in a very versatile way.

- 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…

Thinger real time sound plotter and weather information



Thinger plotter by the minute



Thinger data buckets

Bucket Data

Date	NAT	Above Thresh Dur...	Above Thresh Leq	Less10d B...	Less10d BLeq	Peak Time	Peak Value
2020-09-15T17:58:53.615Z	6	50	63.759464263916016	25	63.759464263916016	17:57:46	66.83229064941406
2020-09-15T17:55:44.613Z	5	52	60.45946502685547	33	59.72776794433594	17:54:41	63.62342071533203
2020-09-15T17:36:51.593Z	4	70	65.37852478027344	33	65.58905792236328	17:35:39	68.84777069091797
2020-09-15T17:32:05.590Z	3	60	62.27629089355469	31	62.16830825805664	17:30:52	65.78192901611328
2020-09-15T17:25:03.581Z	2	23	60.09807205200195	14	59.51743698120117	17:24:04	62.880531311035156
2020-09-15T17:18:42.575Z	1	45	55.47260284423828	36	54.57136535644531	17:17:42	58.484275817871094
2020-09-15T16:43:27.547Z	17	53	55.736690521240234	44	54.831336975097656	16:42:24	59.37811279296875
2020-09-15T16:29:55.536Z	16	54	58.39207458496094	33	57.83545684814453	16:28:52	60.47865295410156
2020-09-15T16:26:30.533Z	15	94	58.1973762512207	60	57.705810546875	16:25:20	62.396209716796875
2020-09-15T16:25:18.532Z	14	78	58.256988525390625	51	57.75934982299805	16:25:17	62.28064727783203
2020-09-15T16:23:46.532Z	13	105	63.85183334350586	53	63.81764602661133	16:23:30	66.52909088134766
2020-09-15T16:22:25.531Z	12	70	60.97349548339844	42	60.431671142578125	16:21:54	63.992164611816406

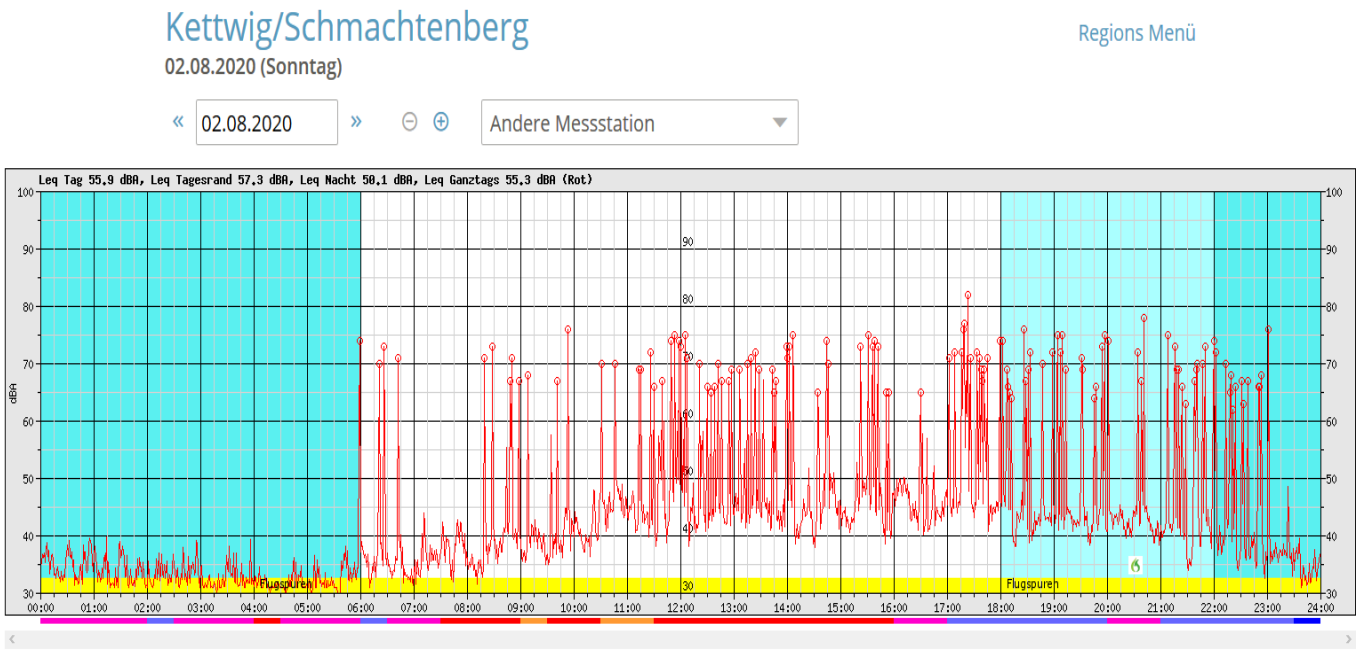
A detailed example of the configuration of the thinger.io services will be provided in a separate document.
(have a look at GitHub)

Residential/communal Noise service http://dfld.de

Last but not least the system can issue, additionally to Thinger, and alternatively to the Serial output a single byte per second according to a proprietary "AK-Modulbus protocol" to a feeder program running e.g on a Raspberry Pi forwarding hourly reports to the European aircraft noise network . http://www.eans.net/EANSindex.php

This network is providing a very long time (over 10 years) lobby-independent storage of aircraft/railway noise information managed by residentials and communities, currently totaling about 700 privately and communal operated noise stations throughout Europa.

Noise record from DFLD/EANS.

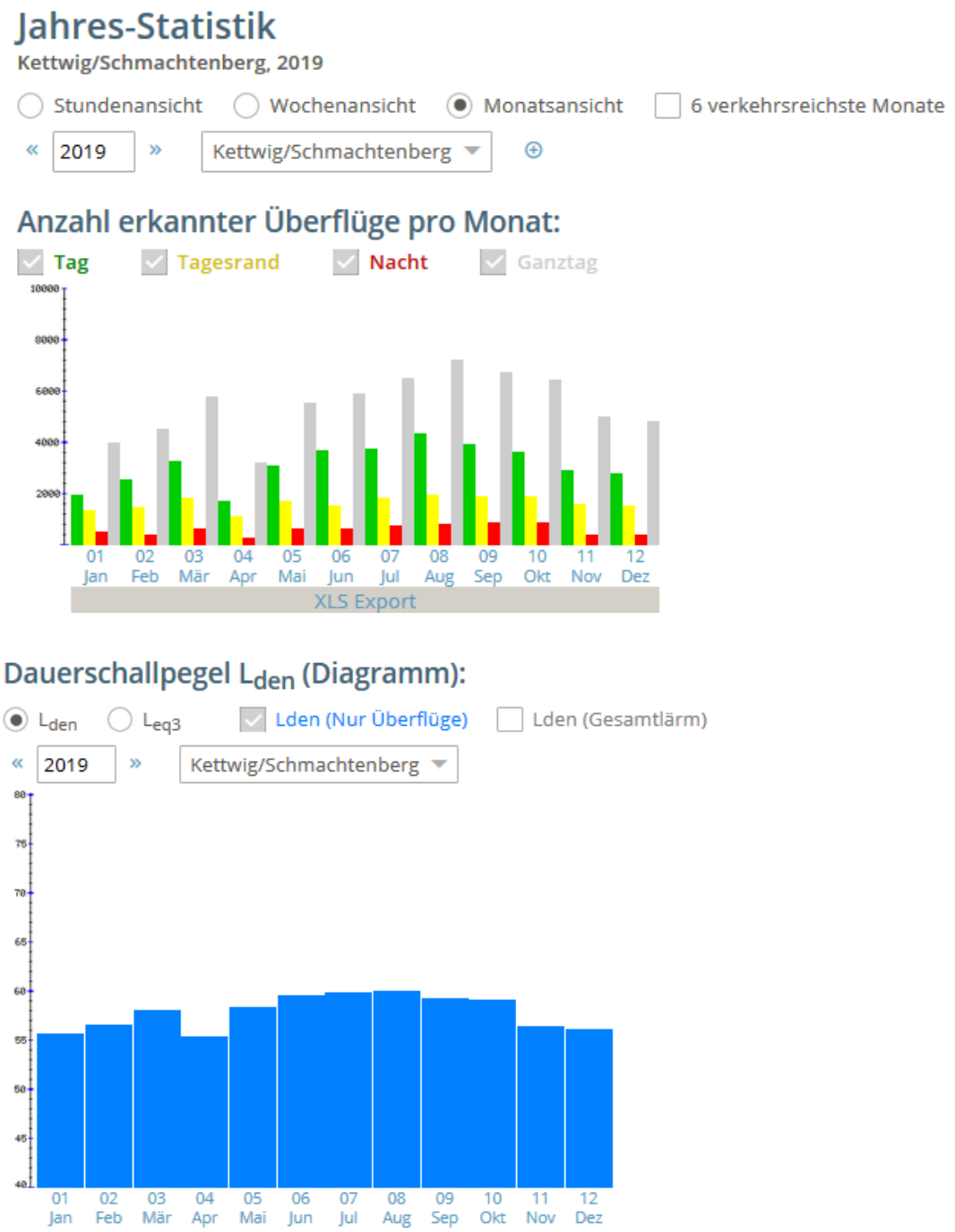


Daily stats from DFLD/EANS

Anz. erkannter Überflüge
Aufgeschlüsselt nach 5 dB_A großen Maximalpegelbereichen

Zeitraum	Überflüge														Dauerschallpegel Überflüge		Gesa
	<50	50–54	55–59	60–64	65–69	70–74	75–79	80–84	85–89	90–94	95–99	≥100	Σ	Max. [dB_A]	L_{eq3}	L_{den} (1)	L_{eq3}
Tag	0	0	0	0	4	12	5	0	0	0	0	0	21	77	50.8		56.3
Tagesrand	0	0	0	0	0	0	0	0	0	0	0	0	0	----	----		39.7
Nacht	0	0	0	0	0	0	0	0	0	0	0	0	0	----	----		39.4
Σ	0	0	0	0	4	12	5	0	0	0	0	0	21	77	49.0	49.0	54.6

Year stats from DFLD/EANS

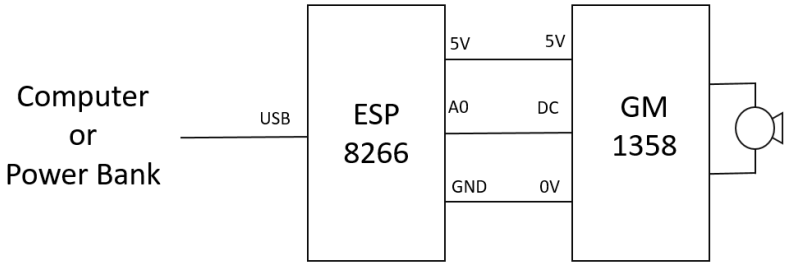


A detailed example of the configuration of the DFLD/EANS gateway (Raspberry Pi) will be provided in a separate document.
(work in progress)

Hardware

If you do not already have a Sound Level Meter, may I propose that model?
GM 1358 (see Bill of materials) it is dirty cheap and fairly accurate.
That one has the unique feature of providing a linear 0..1V DC signal output that is tied to GND
and has the ability to autostart when powered with 5V.
It can interface with an ESP 8266 with only 3 wires, which is really a huge advantage and makes everything simpler.

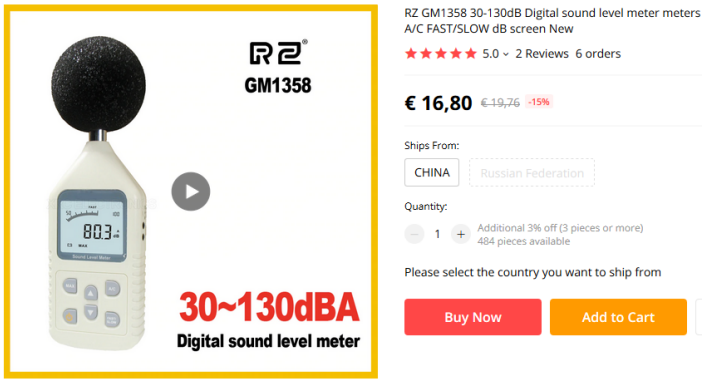
Option 1: Simple version with GM1358



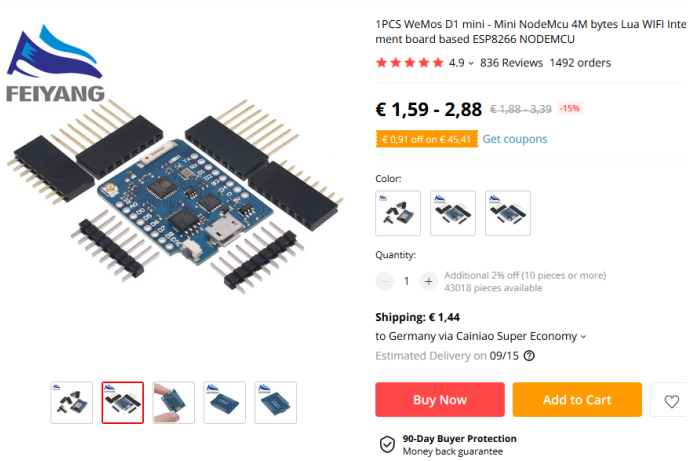
Bill of material

You can find some devices on eBay.com, but you will find the best offers on AliExpress or Temu to largely varying prices. Anything below 40 € is OK…

Currently they are hard to find, but they are always coming back…

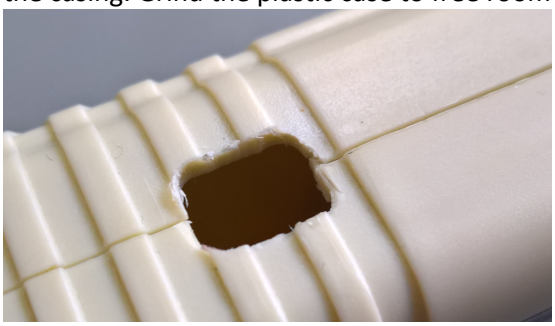


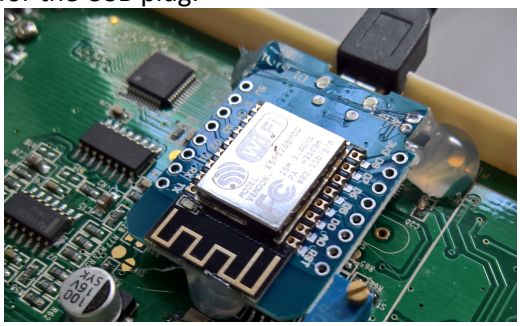
Then you need an ESP8266



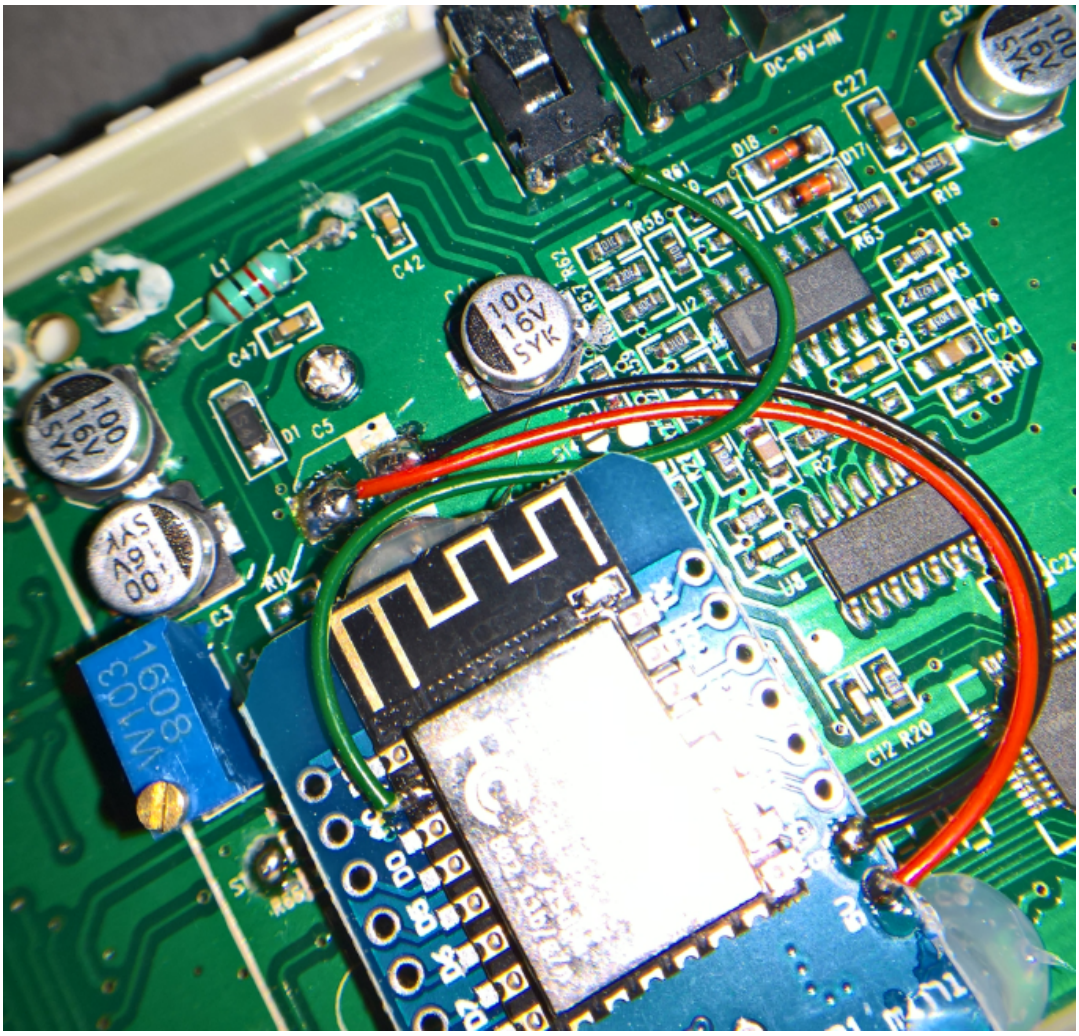
Modification ESP8266 built into the GM1358

The GM1358 is easy to dismantle, remove 4 screws: 2 at the top and 2 in the battery compartment and you can open the casing. Grind the plastic case to free room for the USB plug.



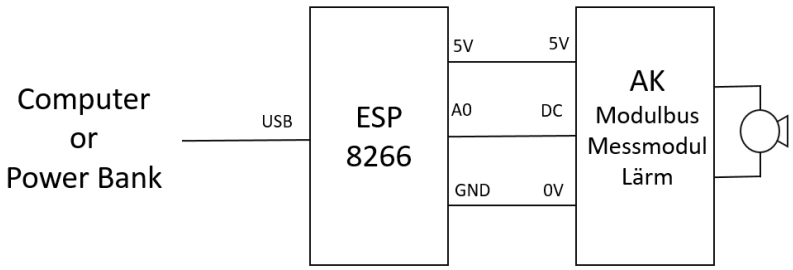


Glue the ESP8266 as shown with an USB connector in place to align the module correctly.



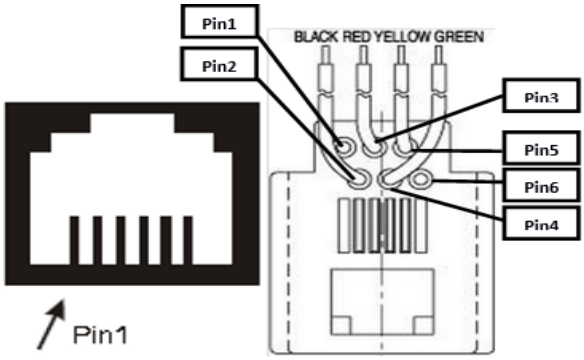
-Solder a wire between the “5V” terminal and the narrow “C3” solder-pad as shown
-Solder a wire between the “G” terminal and the broad “C3” solder-pad as shown
-Solder a wire between the castellated terminal of the ESP module in front of “A0” and the inner contact of the DC jack.
(do not solder on the “A0” terminal, which provides a 3V voltage divider, that we do not want here.)
Screw the case back. Done!

Option 2 Simple version with AK-Modulbus Sound adapter



Upgrade for an existing AK-MODUL-BUS Converter solution.

This is a good option for those of us having already an AK-Modulbus Sound-Transducer used together with a dedicated interface db-Online. https://www.ak-modul-bus.de/stat/messmodul_laerm.html
This module has a RJ12 telephone type connector.
An ESP8266 can replace and emulate (+dramatically improve) the dB-Online interface.



You will need a RJ12 prolongation and cut the male connector.
If you use a regular German standard cable (usually with reversed wiring), you will wire the:

- red(brown) wire (pin3) to A0*
- green wire (pin4) to ground
- yellow wire (pin5) to +5V

and discard the other wires. If you appear to have the (unusual) direct wiring, you will wire the:

- green wire (pin3) to A0
- black wire (pin4) to ground

- red wire to (pin5) +5V

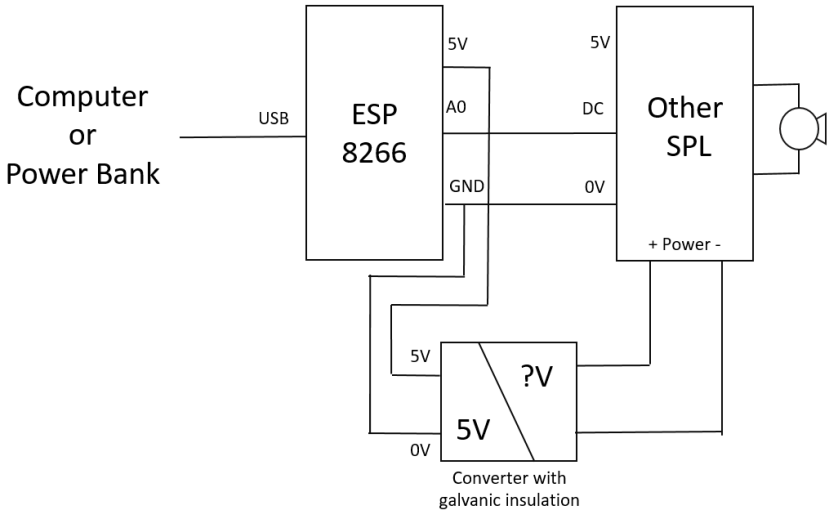
* note: in the AK transducer version, we will connect the red(brown) wire to the A0 connector and not to the ESP, as it was done for a GM1358, since we want this time to use the built-in 3V-1V voltage divider.

Use a bit of hot glue to fix the wires. You are free to use a matchbox-sized casing of your choice, wrap in a shrink sleeve, or even let it blank.

That will save the dB-Online interface, the power supply and the serial-to USB converter.
… and a lot of room and cable mess.

Option 3: Using other SPL devices

You may also use other SPL devices, some of them being of higher quality, up to Class 1 devices.
The only requirement is that they provide a DC-output with a dB proportional value 0..1 V or 0..2,5 V which are the most frequently ranges found in commercial devices. The drawback is that the DC-output of those devices is usually floating with respect to the power supply and so you cannot use a common power supply between the ESP and the SPL. Most SPL meters will also not autostart when they get power, but must be manually started over an on-off button. That makes everything a bit more complicated.



Energy considerations usage on battery

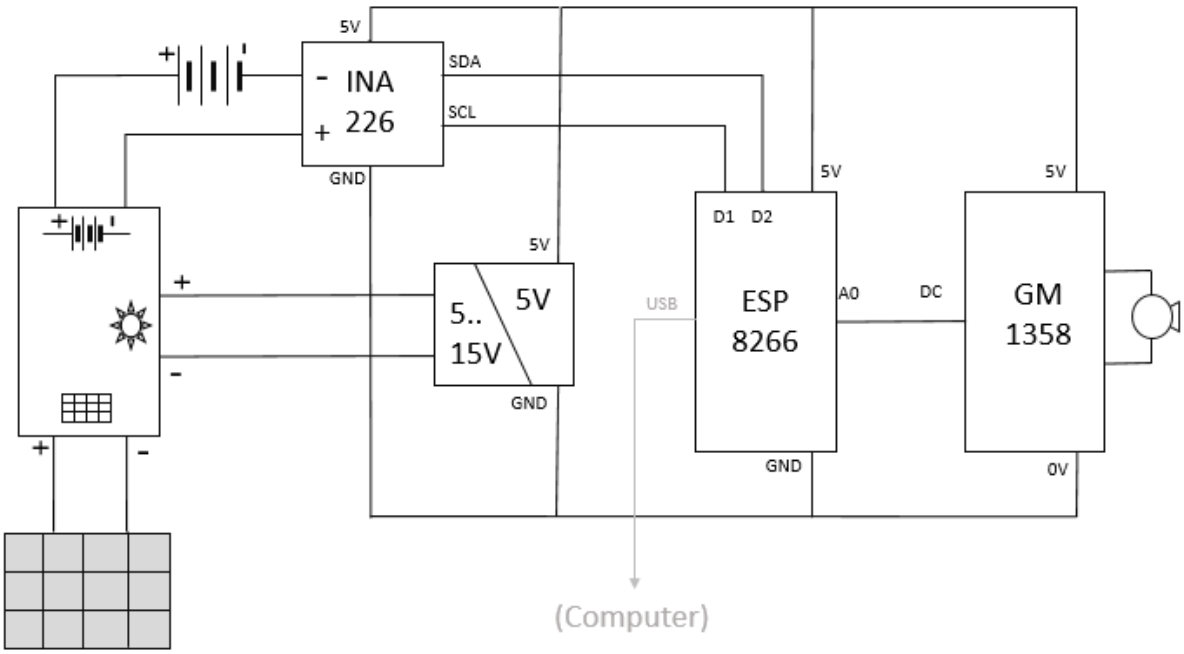
The GM1358 can power the ESP8266 for a wireless operation. The built-in 9V alkaline battery block, provides about 24 hours of untethered operation (@ 11mA) on the SPL alone. With the retrofitted ESP8266 the total consumption will be about 48 mA, which will drain the battery block within about 8 hours only.

Fortunately, you can also power the system over the USB socket of the ESP8266 and power the GM1358 without a

battery block. A 3000mA lithium 18360 Power-bank provides over two days of continuous operation.
A 12V/12Ah Lead-Acid block with appropriate voltage converter considerably more.

Solar Powered Version with battery reporting

If you want a 24/24/365 off-grid operation on solar power, depending on your location
(e.g. at a 45° latitude) you will need a 12V 30Ah battery and (at least) a 20W solar panel,
you can hardly imagine how little energy such as solar panel still delivers on a rainy winter day!

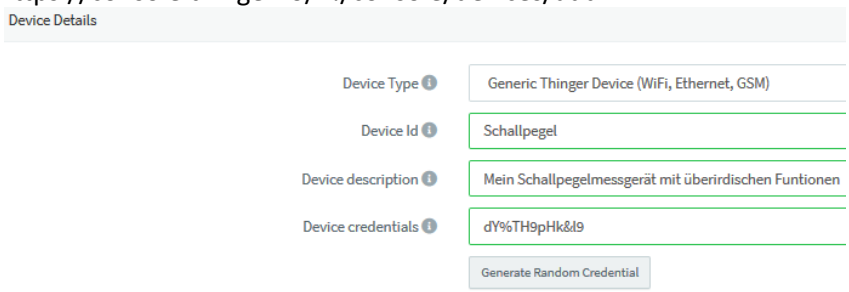


In fact, as soon as you consider a solar-powered off-grid operation, you will have to harden the equipment to cope with the harsh outdoor weather environment: rain, heat, frost, wind, ice…
I will provide a separate description for this variant, which requires much more skills and will be overkill to describe here.

Programming

Registration with Internet services

Before you begin the configuration, you must log in to the services you will use and remember the login details.
1. Create a free account at www.thinger.io http://www.thinger.io
Set up device as follows (you can manage two devices for free):
https://console.thinger.io/#!/console/devices/add



remember this information, as well as your username and password.
2. Set up the following “data buckets”:
Bucket Name Description

- MIN Minute recording Sound level: Min, Slow, Max
- EVENT Event List Above Threshold, Battery warnings
- DAY Daily report LDEN, LDaytime, LNighttime, NAT24h, NAT22-24
- HOUR Hourly Data LEqu, Weather, Battery Normal

3. Create a free account at https://home.openweathermap.org/users/sign_up
to get weather information for your location.
After registering, enter your place of residence here:

Note the location number in the URL: https://openweathermap.org/city/2928810 ,
as well as your username and password.
4. For already members of the DFLD, note the region and station number.

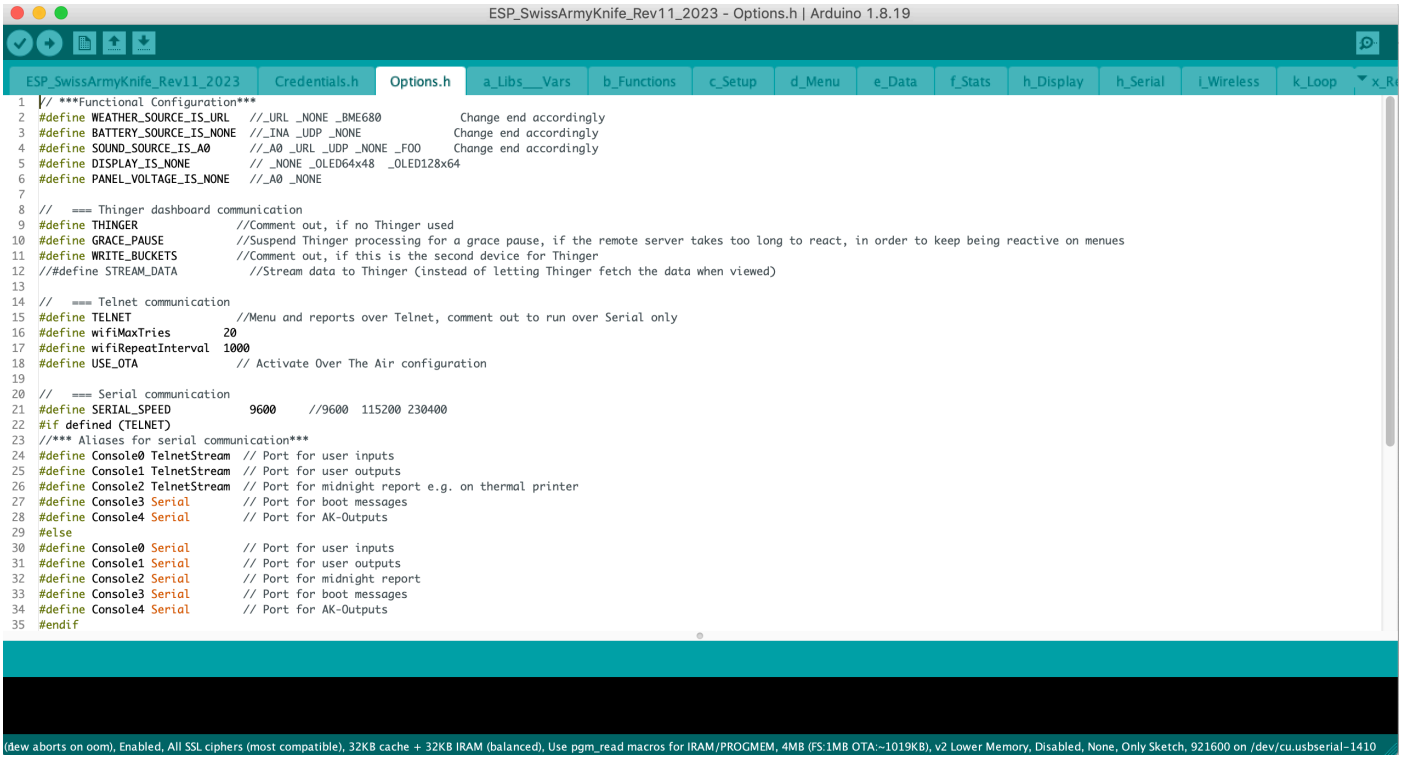
5. You need a WiFi connection and an internet connection to operate, but you already have that…

Get the software from Github and compile.

The ESP8266 is usually programmed over a USB-Connection using the free open source Arduino IDE.
For that, if you do not have it already, you should install the Arduino programming Interface (IDE) from
www.arduino.cc <http://www.arduino.cc>, on your favourite PC, and then install the ESP framework.
Cf these instructions:
https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/
The current files of my program are hosted on GitHub:
https://github.com/rin67630/SPL-Booster

You can download all files in a .Zip file using the button 

.
Unzip the file on your computer. Then copy the folder name from “SPL-Booster-master” to
“ESP_SwissArmyKnife_Rev11_2023” in the Arduino IDE workspace and double-click
“EESP_SwissArmyKnife_Rev11_2023.ino” in this folder to start the Arduino IDE:



I have built the program on the top of my software framework “ESP-Karajan” for the ESP8266 on GitHub: https://github.com/rin67630/ESP-Karajan , which takes care of ancillary tasks like: booting to the network and providing scheduling and timing functionalities.

The code provides also the possibility to upload / maintain the software over the air:
http://arduino.esp8266.com/Arduino/versions/2.0.0/doc/ota_updates/ota_updates.html#arduino-ide

The Arduino IDE provides tabs to split the program into well-structured subparts, so you I could jump easily during development between every subpart:
Credentials.h (where you put your private credentials)
a0) options to use the program *(the only two parts you really need to modify)*

a1) libraries and global variables used
b) different functions used in the program
c) setup process
d) menu
e) data processing
f) display (not used here...)
g) serial reports
h) wireless processes
k) finally the scheduler itself which will periodically start the routines listed from d) to h).

Additionally I added for convenience two tabs with comments only:
x_ReadMe and y_ParkedCode, where I put reminders and code examples.

In my programming style, I deliberately refrain to use too abstract c++ concepts in order to make the program accessible to the majority of people with an average of Arduino experience.
I also did put a great attention to comment my code indicating the reason why I have done most steps in that way.

Different modes of operation

To use this software in your own WiFi network, you will need to enter some parameters and credentials network.

Simple mode of operation

-the software gets the weather information from the Internet
-There is no battery management
-The input of the noise level comes from a SPL over analog input A0.
-You will use the thinger services with one device only
-All serial output will be either unused or go to the standard serial port

You will have to enter credentials and options to the first part of the configuration tab only.

Extended modes of operation

The software can however provide additional elaborated modes of operation, some with two or more ESP8266:

a) an ESP8266 without additional hardware retrieving sound information from an URL of DFLD to display dashboards/provide statistics.
b) split operation: e.g. there is a primary ESP8266 outside connected to the sound pressure level meter, which sends over UDP the noise information to a secondary ESP8266 inside connected to the computer over WiFi UDP.
c) split operation: e.g. there is a primary ESP8266 outside connected to the sound pressure level meter, powered by solar, having battery management hardware, which sends over UDP the noise information and the battery metering information to a secondary ESP8266 inside connected to the computer and/or Thinger.
d) split operation: e.g. there is an ESP8266 dedicated to battery management, and a second one processing the sound pressure level, one of them forward the information to thinger.io
e) a primaryESP8266 processes the battery management and the sound pressure level signal and forwards the values to a second receiver ESP8266, without additional hardware to print reports and interact with Thinger.
f) multifunction: a single ESP8266 processes the battery management and the sound pressure level signal alone.
g) two independent noise processors on the same thinger account. (not recommended, but possible: only one should write onto buckets)
h) other combinations of tasks between ESP8266…

You will only have to enter configuration information in the second part of the configuration tab if you intend to use one of these extended modes of operation.

Configuration

To operate the noise station you will need a WLAN connection and internet connectivity.
You will need to upload the program as described under the Chapter Programming.

To configure the options, unless you know exactly what you are doing, you only need to make changes to the Tab “a0 parameters”

Here is an example of the Credentials content:

```
//  ***Credentials***
#define WIFI_SSID           "Public SSID"         //default SSID
#define WIFI_PASS           "Secret Password"     //default Password
#define UDP_TARGET          "192.168.?.?"         // Destination IP for your own Raspberry Pi
#define HOST_NAME            "Thinger_Device"     // Must match Thinger device name
#define THINGER_USERNAME     "Thinger_User"       // Must match Thinger user name
#define UDP_TARGET           "192.168.001.001"    // Destination IP for Raspberry Pi (enter real local IP)
#define UDP_PORT             4321                 // Port to send data, next port to send reports
#define THINGER_CREDENTIALS  "Thinger Credentials"// Must match Thinger device credentials (Not user password)
#define DFLD_REGION         "???"                 // Airport region at DFLD
#define DFLD_STATION        "???"                 // Station number at DFLD
#define Ao94                 747                  // 747  Input AK with offset and 2,5v
#define Ao47                 461                  // 461  Input AK with offset and 2,5v
  // ***Event parameters***
#define  WIND_LIMIT                  15 // upper limit of Wind speed to record NATs
#define  UPPER_LIMIT_DB              88 // upper limit of Thinger plots
#define  LOWER_LIMIT_DB              31 // lower limit of Thinger plots
#define  EVENT_THRESHOLD_LEVEL       58 // Begin of Exceedance level
#define  NAT_THRESHOLD_LEVEL         58 // Begin of Exceedance level
#define  MEASUREMENT_THRESHOLD_LEVEL 55 // Begin of measurement level
#define  MIN_EXCEEDANCE_TIME         20 // Minimum duration of an event
#define  MAX_EXCEEDANCE_TIME         80 // Maximum duration of an event
```

```
#define  LISTENING_TIME              20 // mimimum time between events

// ***Time zones***
#define NTP_SERVER "de.pool.ntp.org"
#define MYTZ TZ_Europe_Paris
#define TZ   1               // (utc+) TZ in hours

// ***Weather server***
#define OPEN_WEATHER_MAP_APP_ID      "7f02173c19eed016c4797f4ca4251fcc"
#define OPEN_WEATHER_MAP_LOCATION_ID "2876401"      // Lohausen
#define OPEN_WEATHER_MAP_LANGUAGE    "de"
#define OPEN_WEATHER_MAP_UNITS       "metric"
```

Please enter the following information and credentials in green:

a) a freely chosen hostname in your network (for your WiFi and e.g. for Thinger)
b) Your WiFi credentials*
c) the login information and parameters for OpenWeatherMaps
d) the credentials and parameters for Thinger
e) the time zone
f) The default calibration parameters to match your sound pressure with the analog input *
g) the event parameters for event detection (or leave the default settings)
h) Leave everything under "//Electrical Parameters" unchanged.

```
* // ***Acoustical parameters***
If you are using the recommended sound pressure meter GM1368 or another that provides a proportional DC output of 10
mV per dB (range 0..1.0 V), leave the parameters unchanged.

If you are using a sound pressure meter that provides a proportional DC output of 2.5 mV per dB (range 0..2.5 V),
change the parameters to:
#define Ao94 712 and #define Ao47 356.

If you use an AK Modulbus transducer that has a DC output of 25 mV per dB
(range 0..2.5 V) with offset, change the parameters to:
#define Ao94 747 and #define Ao47 461

If you are using another sound pressure meter that provides a proportional DC output with offset, then go to the Notes
chapter “Setting parameters for other sound level meters”
```

* you may use smartconfig https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_smartconfig.html to enter your WiFi credentials. This is the most versatile method and enables adapting to various WiFi networks without recompiling, but need to run an app on your smartphone. Alternatively you can comment out Smartconfig and enter your credentials directly:

```
//define SMARTCONFIG  // (WiFi Credentials defined over smartphone App SmartConfig)

// alternatively to Smartconfig App, you can comment out the line Smartconfig
// and enter your credentials to initalize for a new WiFi:
#define WIFI_SSID          "SSID"
#define WIFI_PASS          "Password"
```

** -if you use the recommended SPL meter GM1368 or another one delivering a proportional DC output 10mV per dB (Range 0..1,0 V) , leave the parameters unchanged,
-if you use a sound transducer rom AK Modulbus delivering a biased DC output 25mV per dB (Range 0..2,5 V) , change the parameters to #define Ao94 747 and #define Ao47 461,
-if you use another SPL meter delivering any proportional DC output between 0..3V, you will have to find out which A0 ADC value corresponds to SPL measures at 94dB and at 47dB and enter the values respectively, using the '?': //List parameters serial menu command.
n.b. these are the defaults parameters only, you may exactly calibrate any time over the serial menu.

Table of Contents

Digest ***2***
Hardware 3

Menu System ***4***

Device management **4**

Reports **4**
One shot reports 4

Report examples **5**
Minute Hour Day and Events 5

Ongoing reporting: **5**

Graphical reporting: **6**

Cloud service Thinger ***7***

Thinger real time sound plotter and weather information **7**

Thinger data buckets **8**

Residential/communal Noise service http://dfld.de ***9***

Noise record from DFLD/EANS **9**

Daily stats from DFLD/EANS **9**

Year stats from DFLD/EANS **10**

Hardware ***11***

Option 1: Simple version with GM1358 ..... **11**
Bill of material 11
Modification ESP8266 built into the GM1358 12

Option 2 Simple version with AK-Modulbus Sound adapter **13**

Option 3: Using other SPL devices **14**

Energy considerations usage on battery **14**

Solar Powered Version with battery reporting **15**

Programming ***16***
Registration with Internet services 16
Get the software from Github and compile. 16

Different modes of operation **17**
Simple mode of operation 18
Extended modes of operation 18

Configuration ***18***

Table of Contents ***20***

Apendixes ***22***
Extended modes of operation 22

Norms and Regulations on Residential Noise ***26***

Noise Metrics used. **26**
Leq4 metric 26
Leq3 metric 27
LAE metric 27
LE metric 27
The residual sound level 27
Day-night average sound level (Lden) 27

Weather impact 27
Event assessment and evaluation *28*
Assessment: **28**
State o 28
State a 28
State b 28
State c: 28
State d 28
State e 28
Evaluation **28**
Data sources for Thinger *29*
Device resources provided **29**
Noise 29
Energy 29
DAY 29
HOUR 29
MIN 29
EVENT 29
Device properties **29**
Nat 29
lequ 29
BAT 29
persistance 30
Data buckets **30**
MIN 30
EVENT 30
DAY 30
HOUR 30
Thinger configuration *30*
Create buckets **30**
Create dashboards **30**

Apendixes

Extended modes of operation

The parameters for extended modes of operation are only required if you intend to operate with extra hardware or already have an noise station at DFLD running and want to use an ESP8266 as an extra reporting device.

You can configure:

a) Whether you will be using Weather information, change the end accordingly:
   _URL to get the information from openweathermap.org.
   _NONE if you do not need the weather information.

b) Configure the source of the battery information, change the end accordingly: :
   _NONE, if you don't have any battery information.
   _INA if you have an INA226 module to measure current and voltage
   _UDP if you have in the same network another ESP8266
   running the same sketch as a master (`//#define PUBLISH_BATTERY`)

c) Configure the source of the sound information
   _NONE, if you don't have any battery information
   _A0 if the sound level is coming from the analog input A0
   _UDP if you have in the same network another ESP8266
   running the same sketch as a master (`//#define PUBLISH_SOUND`)
   _URL if the sound level is coming from a JSON URL of www.dfld.de <http://www.dfld.de>
   (then you need to provide below Region/Station parameters of the DFLD station which will provide the sound information)

If you have defined _INA as a source of a battery information, you should verify the parameters under optional electrical parameters for battery management.
You should match the value of your breakout's current shunt and define the measuring ranges according to your battery.

If you have defined _URL as the source of sound information you should enter the region and the station number of the DFLD station you are referring to.

The line #define THINGER is determining, if you are using the cloud services of thinger.io. If you do not and just use serial reporting, please un-comment that line.
The line #define WRITE_BUCKETS is determining, if this device is allowed to write the buckets.
If you have two devices reporting to Thinger.io, only one master device should write the buckets and you should comment out that line.

The lines with #define CONSOLEx determine where some parts of the sketch are issued.
Normally reports issued to Serial, with additional UARTS ports, you can for instance use Serial1 for output.
Serial1 can e.g. be wired to a thermal printer for printed logging.

If you have a networked computer e.g. as a gateway to DFLD, you should un-comment PUBLISH_DFLD
You may also use PUBLISH_REPORT to get the events and the midnight report issued to that computer.
You must then enter the parameters UDP-TARGET and UDP-Port accordingly.

If you are using split configuration, enter PUBLISH_BATTERY and/or PUBLISH_SOUND on the primary device and define UDP-TARGET as the IP address of the targeted ESP8266 and match the UDP-Port between both devices.
It is not possible to PUBLISH_BATTERY and/or PUBLISH_SOUND to another ESP-device and PUBLISH_REPORT to a computer simultaneously.

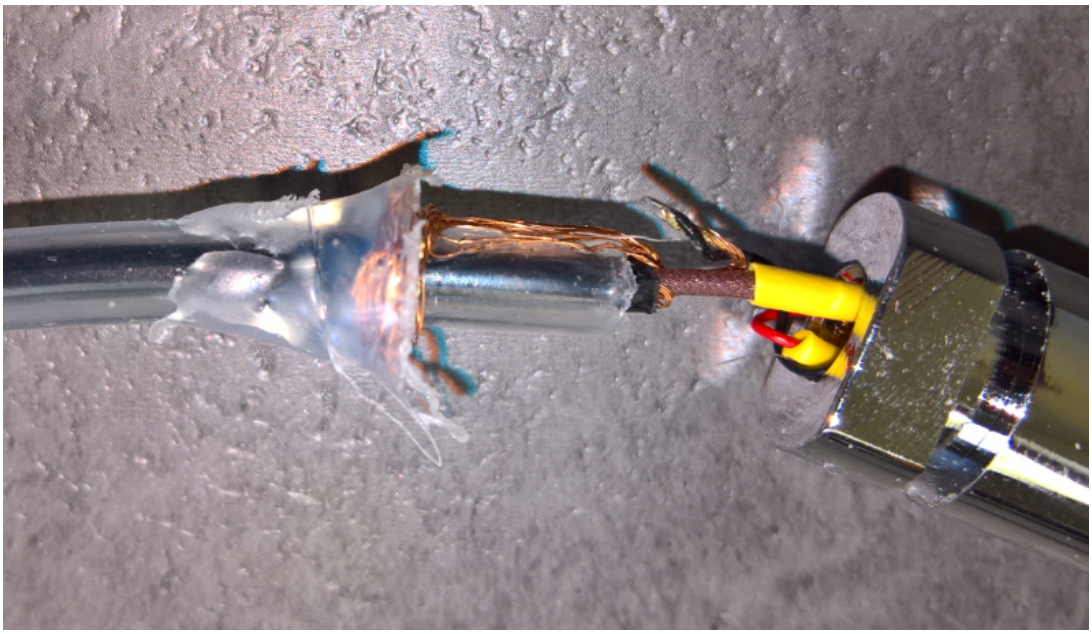
Hardware #3: GM1358 with remote outdoor microphone.

If you want to use the GM1358 24/24/365 for measuring sound pressure outdoors, may I strongly recommend the following outdoor hardening modification?

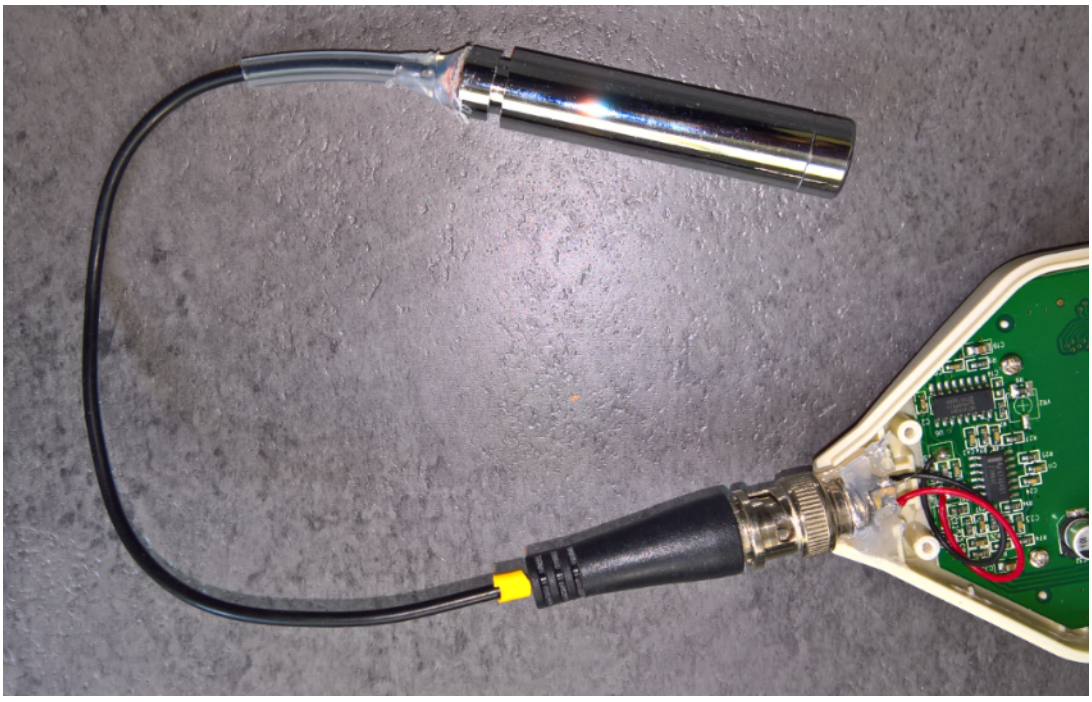
You will need an electrical DN20/DN25 cable gland, a DN40 sewing pipe and its top stopple, a 12mm stainless steel shrimp filter for aquarium amateurs, a piece of 5mm acrylic hose, a stainless steel frypan scourer and some kind of high quality audio shielded cable with connectors + another pair of corresponding connectors for the microphone and the sound pressure level meter.

I have chosen here BNC connectors, which provide a good quality, and you may find on the decaying analog video market a lot of bargain offers.
You may also use stage grade audio-jacks and the corresponding cables...

You will first have to unsolder the microphone boom from the meter and solder the female connector in place.



On the microphone side, you will sleeve the 5 mm acrylic hose over the male cable, connect and insulate the hot wire with the red microphone wire, solder a part of the braid to the black microphone cable (you don't need to insulate it) and leave the other parts of the braid run along the acrylic hose, then push everything inside the microphone boom and hot glue everything together.

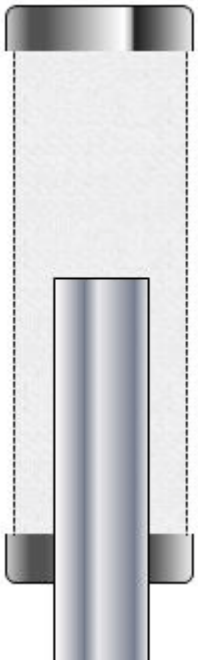


Having the braid uninsulated and pressed against the metallic boom is an improvement: the boom will be a shield as well.

You now have a removable microphone boom for local calibration.

The next step will be to harden the microphone for an outdoor usage.

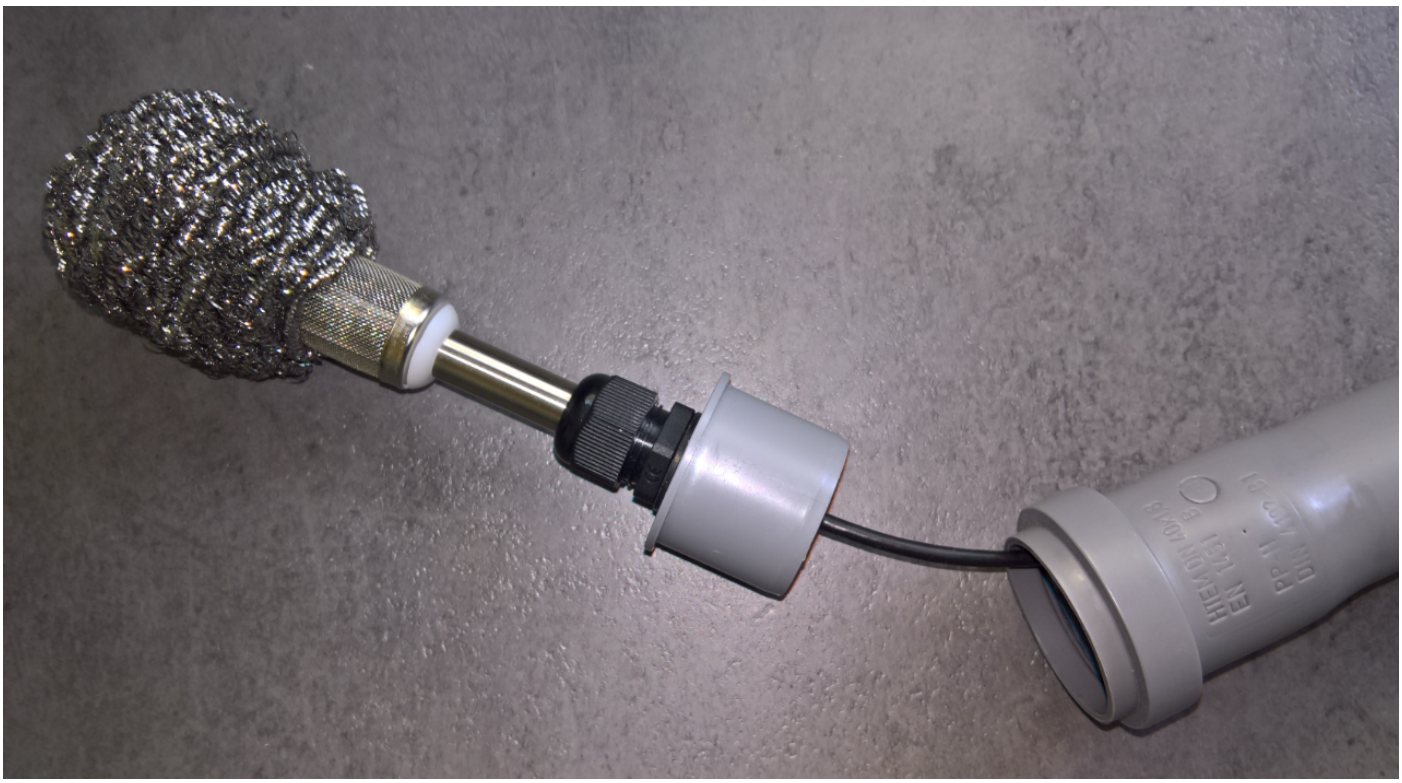
The microphone must be protected from the harsh outside conditions:
rain, wind, frost and snow. Putting it under a roof is not an option; there must be no obstacle between the microphone and the sky.
A simple shrimp filter can do that job amazingly well. You can pull it over the microphone boom: any water will be caught into the stainless steel mesh by capillarity and the microphone capsule will stay dry.



I have tested that on a rainy November month with a dummy tube filled with salt. After 30 days outside the salt was not melt!

Then you need to reduce wind noise. I do not recommend the regular rubber foam wind caps as used in the acoustical industry: they would fit; the outside dimension of the filter is roughly 25mm. However, they would not last very long: the UV radiation destroys them within weeks. The acoustical industry is not worried about that, they are literally making their money out of service agreements to replace the caps every two weeks. ;-)
We will do it better and use a fully maintenance-free solution: you just pull a plain frypan-scourer over the aquarium filter! It will last forever and does not retain rain: the water will just flow through.

I have tested the solution in a wind chamber at the University of Düsseldorf: at 10m/s the wind noise was 56dB: that was a good as the indoor wind cap from Bruel & Kjaer
The $1 scourer would fully meet the legal requirement to be under 60dB @ 10m/s!



The last point is now to find a hold for the microphone and to protect the connector from water and dust.
One of the cheapest and most efficient way is to use a piece of sewage pipe, and the corresponding stopple.

Drill a hole of 25 mm diameter into the stopple, put the microphone boom into the cable gland, screwed into the stopple, pull everything (including the other side of the microphone cable extension) together e.g. into e.g. the holder of a garden sunshield or tape the pipe to a pole in the garden. The microphone should be accessible; you will appreciate to be able to unplug the microphone without tools, leaving the microphone cable extension in place, to recalibrate it indoors from time to time.

Here you are! That outdoor microphone defies the weather conditions since years without requiring any maintenance.

With a good shielded audio cable, you can run the mic up to 6 meters from the SPL meter.
That should be enough to place the SPL device, weather-protected, under a roof.
For longer cable length, I have a solution with a microphone preamplifier as well, described in another document.

Norms and Regulations on Residential Noise

Noise Metrics used.

The programs aims at complying to ECAC (European Civil Aviation Conference) International agreement Volume 1: https://www.ecac-ceac.org/documents/10189/51566/01.+Doc29+4th+Edition+Volume+1.pdf/bfde6e09-b46b-44e1-b73f-388fc3527aaf

Equivalent sound level (LEQ)

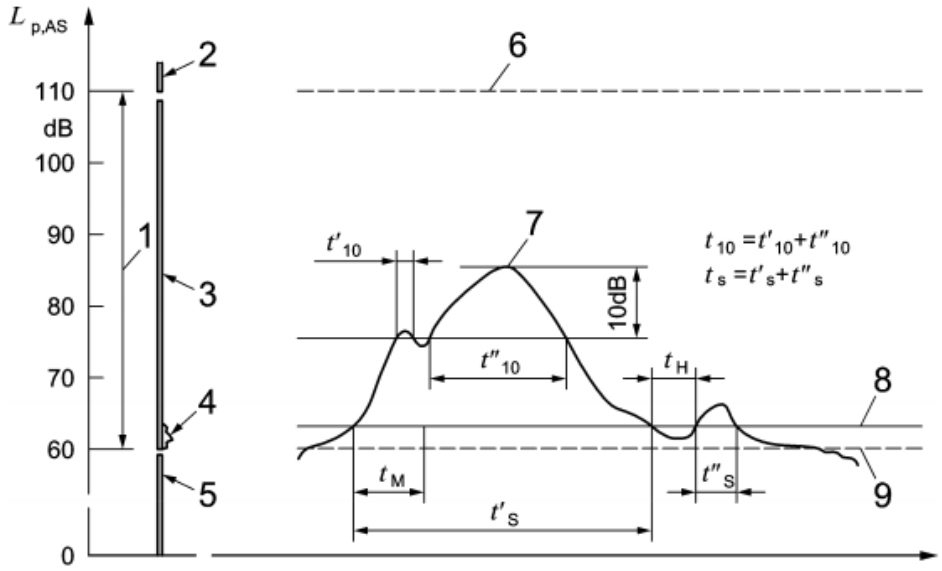
That metric measures the average acoustic energy over a period of time to take account of the cumulative effect of multiple noise events. This could, for example, provide a measure of the aggregate sound at a location that has airplane flyovers throughout the day. LEQ is defined as the level of continuous sound over a given time period that would deliver the same amount of energy as the actual, varying sound exposure.

The Leq metrics and all metrics used in residential noise computations are not simple averages, but are the result of a complex calculation:
The sound level is first delogarithmed (exponential function) to convert decibel into the linear sound pressure, averaged, and subsequently logarithmed again to return the Leq result:

$$L_{p,\mathrm{A,eq},T} = 10\ \mathrm{lg}\left(\frac{t_0}{T}\sum_{i=1}^{N} 10^{L_{p,\mathrm{A,E},i}/10\ \mathrm{dB}}\right)\mathrm{dB}$$

Leq4 metric

The noise metric represents (unfortunately only legally, not fully) all the acoustic energy (a.k.a. sound pressure) of an aircraft noise event, delimited by the boundaries of its own maximum level minus 10 dB.
This figure will be the only accepted metric for the noise event, all (really existing and already annoying part below that part are not considered). ☹
The T10 (t"$_{10}$) metric is the duration of that event.



Keys used in figure according to DIN:
1 Primary indication range/dynamic range, 2 Overload range, 3 Range included in the assessment
4 Range not included in the assessment, 5 Range not transferred, 6 Upper limit of the primary indication range/dynamic range, 7 Maximum sound level Lp,AS,max, 8 Measurement threshold level Lp,AS,MSchw
9 Lower limit of the primary indication range/dynamic range
t 10 10 dB-down-time, H Listening time, M Minimum time, t s exceedance time

Leq3 metric

The Leq3 metric represents (unfortunately, not legally) all the acoustic energy (a.k.a. sound pressure) of an aircraft noise event delimited by a given threshold (e.g. 55dB), which is much closer to the real annoyance.
The tAT (t'_s)metric is the duration of that event.

LAE metric

The LAE metric (Single event sound exposure level) is the sound level an event defined would have if all its sound energy were compressed uniformly into 1 second practically 1 second. This is a standard single event value, described e.g. in ISO 1996.
This scale thus takes account of the duration of the event as well as its maximum intensity.

Lmax metric

The maximum, value of L(t)-slow that occurs during an event.

LE metric

The LE metric is a special form of the LEQ metric. It concentrates all acoustical energy of an event into one single second.
This has the advantage of categorizing events with a single value independent of their duration. However the nominal result lead to values well above the maximum of the event, which might be confusing for nonspecialists.
The LE, calculated for one minute instead of one second, would have been 17.8 dB lower and hence more acceptable, but I cannot change the norm. ☹
However, the LE is a convenient intermediary stage for averaging the sound burden to hours, days or months.

The residual sound level

The residual sound metric is the long-term averaged sound level excluding the noise events.
The noise events must (legally) at least exceed the residual sound by 5dB, which is in my opinion ridiculously low. I considered at least 10dB distance from residual sound, to accept the noise event.

Day-night average sound level (Lden)

That noise metric reflects a person's cumulative exposure to sound over a 24-hour period, expressed as the noise level for the average day of the year based on annual aircraft operations, given an additional 10dB for events occurring during the legal night (from 22:00 to 06:00).
Lden is the standard noise metric used for all FAA studies of aviation noise exposure in airport communities.

Weather impact

The microphone used in the weather station should not put more than 10dB additional noise upon a constant laminar wind strength of 10 m/s.
I have tested my microphones at the Düsseldorf University of technology in a wind funel and the test has proved the design compliant.
However according to the regulations, for wind and wind gusts above 10m/s, the overflight detection is disabled.
The weather condition is gathered form openweathermaps.org. Due to the cost and complexity, I do not yet consider an individual anemometer.

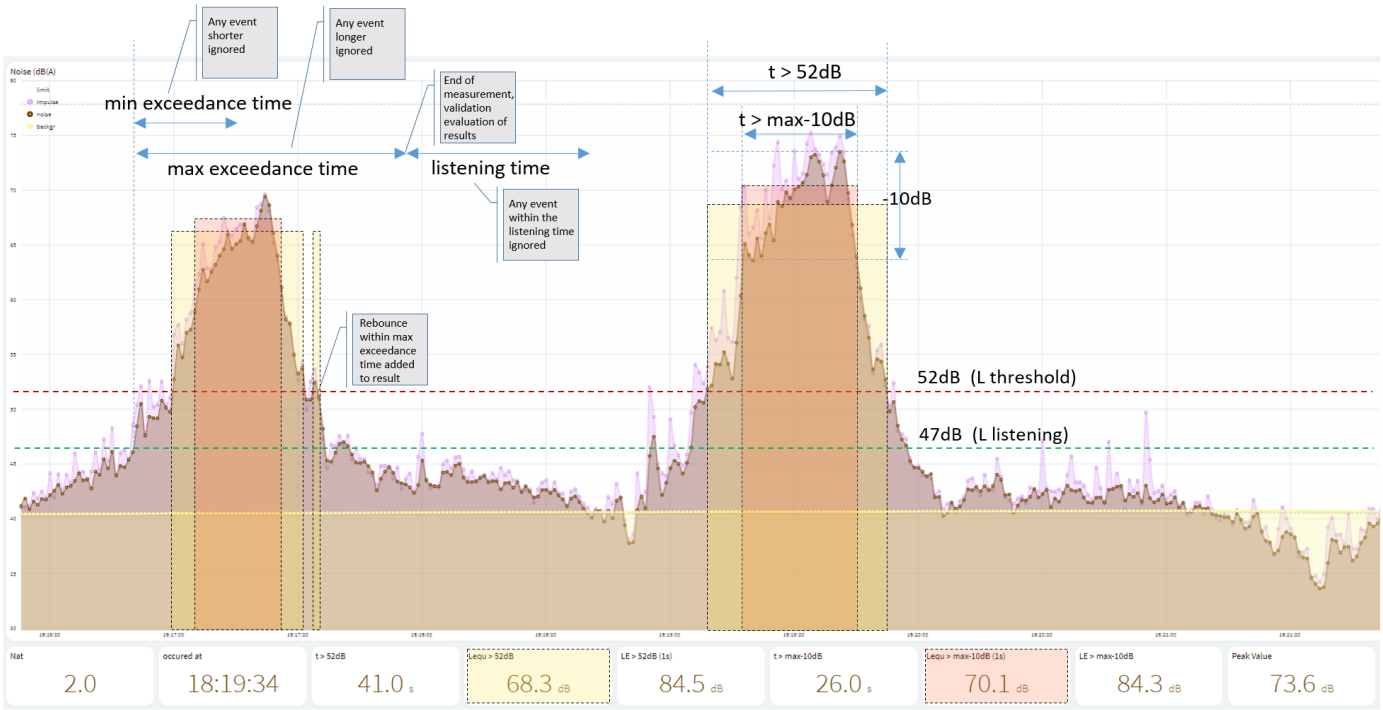
NAT metric

The number above threshold is the number of detected events according to the Leq4 metric within a given period.
Legally, one considers NAT limits only during night.
Airports usually report NAT for day (6:00 -22:00) and night (22:00 -6:00) only.
I am reporting NAT hourly to identify the most noise impacted hours of the day

Further reading on metrics

ICCAN: A review of aviation noise metrics and measurement July 2020: https://iccan.gov.uk/wp-content/uploads/2020_08_11_ICCAN_review_of_aviation_noise_metrics_and_measurement.pdf

Event assessment and evaluation



Assessment:

The assessment of the noise events occurs in the program through a state machine.

State o:

at the end of a recognition cycle is the idle situation where the program waits for the listening timer to expire, then for leaving the listening level → state a

State a:

whenever the noise level exceeds the listening level, the timers for minimum exceedance time and the maximum exceedance time are started and the event recording begins→ state b

State b:

should the noise level fall below the listening level → state e, else if the event duration is longer than the minimum and shorter than the maximum, the event is validated, → state c

State c:

just wait for the maximum duration to expire → state d

State d:

the timer for the listening time is started, the recording stops, the event assessment is successful and will then go into numeric evaluation:

- the recording will then be played back searching for the maximum noise level of the event, then computing the level maximum -10 dB.
- the recording will be evaluated integrating the sound energy for every sample exceeding the level maximum -10 dB, the corresponding results will be saved.
- the recording will be evaluated again integrating the sound energy for every sample exceeding the fixed threshold level, the corresponding results will be saved. → state e

State e:

waiting the listen time for the next event → state o

Evaluation

Every event is evaluated according to two referentials: the referential „maximum minus 10 dB" and the "fixed threshold" referential.

For every referential, the program integrates the actual related sound energy of the samples exceeding the referential (that includes eventual rebounces) and the total duration of the matching (1s) samples.

- The LE metric will be the logarithm of the total sound energy
- The Leq metric will be the logarithm of the total sound energy divided by the total duration.

Data sources for Thinger

The software provides various data information for usage within Thinger as widgets.
The widget "time series chart" permits to display plots of these values, out of device resources or device properties. These will built up up on time, when you call the dashboard. You cannot display that way data that was generated before the call of the dashboard. The update rate may be as fast as 1 second, but do not put too much on one page, it eats quite a lot of resources…

You can also use the data buckets as a data source, in this case, the data will be fetched from the history in the data buckets and be displayed immediately. The update rate may be at least one minute.

Other widgets prevent the display the value numerically or as bar graphs, tachometers etc...

Device resources provided

Device resources are the basic possibility to provide information to widgets in a dashboard. The resources can be either polled from the dashboard whenever the dashboard is on screen or permanently forwarded proactively from software when it is internally updated.

Noise

- Min, Slow, Max, Background, Fast, Impulse, Limit,

forwarded every second

Energy

- voltage, current, power, internal resistance, percentage charged,

forwarded every minute

DAY

- Ah four the current hour, voltage at four o'clock, voltage delta from one day to the following, measured at four o'clock, Leq values for every hour of the day, NAT for every hour of the day, battery Ah for every hour of the day.

forwarded at the end of the day

HOUR

- temperature, humidity, air pressure, wind speed, wind direction, whether description, battery current, battery voltage,

forwarded at the end of the hour

MIN

- Noise min, max, background, leq, battery voltage, power to/from the battery

forwarded every minute

EVENT

- event peak time, peak value, above threshold LE, above threshold LEQ, above threshold duration, less 10dBLE, less 10dBLeq, less 10 duration, NAT

forwarded when an event has been detected.

Device properties

Device properties are an additional possibility to provide information to widgets in a dashboard. The properties are updated from the software every 6 min. The properties also read back from Thinger upon initialization of the ESP 8266 to ensure persistence of information across reboots.

Nat

Object { 00h: 0, 01h: 0, 02h: 0, 03h: 0, 04h: 0, 05h: 0, 06h: 16, 07h: 10, 08h: 9, 09h: 2, 10h: 10, 11h: 6, 12h: 0, 13h: 6, 14h: 8, 15h: 5, 16h: 5, 17h: 8, 18h: 10, 19h: 8, 20h: 6, 21h: 3, 22h: 0, 23h: 1, Day: 90, Daytime: 89, NAT: 0, NAT22-24: 1, Nighttime: 1 }

lequ

Object { 00h to 23h, Day, L-Daytime, EV10, EvOT, L22-24h, Lden, Leq, L-Nighttime}

BAT

Object {Ah Battery by the hour, from 00h to 23h}

persistance

(the purpose of the device resource persistence is mainly to provide to the program continuity across reboots: the data is not primarily intended for usage in a dashboard. The list is only given to be complete)
Object { A047, A094, A0dBBgr, A0dBSumExp, Ah/hour, Ah/yesterday, aboveThreshLEint, corrdB: -2, currentInt, direction, humidity, last_update, less10dBLEint, nCurrent, pressure, resistance, summary, temperature, voltageAt4h, voltageDelta, wind}

Data buckets

Data buckets are kind of long-term persistent files within Thinger, able to store information for up to one year.
The widgets in the dashboards can display the last value of them or provide a trend of the values for selected the time span.
The content of data buckets can be exported to CSV and hence to Excel etc…

MIN

-Sound level: Min, Slow, Max, Battery: Volt, Power

EVENT

-Above Threshold, Battery warnings

DAY

-Lden, L Daytime, L Nighttime, NAT24h, NAT22-24,BatteryBalance

HOUR

-LEqu, Weather, Battery

Thinger configuration

Mandatory default configuration to match the device:
Create device
https://console.thinger.io/#!/console/devices/add + add device

- Device Type =Generic Thinger device
- Device Id = [matching THINGER_DEVICE]
- Device description = Swiss_Army_Knife_GM1358 (or anything else)
- Device credentials =[matching THINGER_CREDENTIALS]

Create buckets

https://console.thinger.io/#!/console/buckets/add +create the 4 buckets: MIN , HOUR, DAY, EVENT

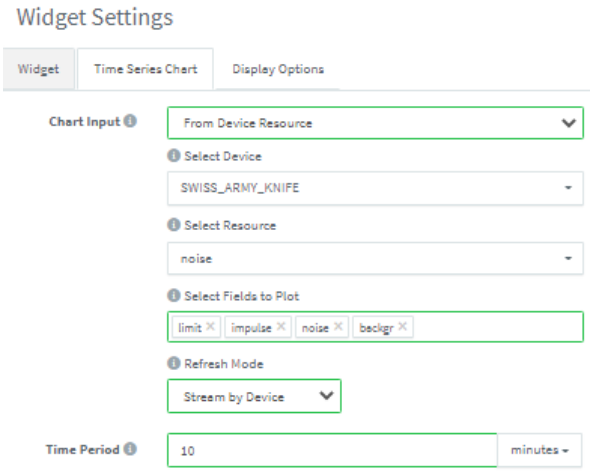
- Bucket id = [MIN]
- Bucket name = free description
- Bucket description= free description
- Enabled [yes]
- Data Source [From Device Write Call]

Create dashboards

Optional dashboards, up to your wishes.
https://console.thinger.io/#!/console/dashboards/add you my create 3 dashboards: REALTIME, MIN, DAY

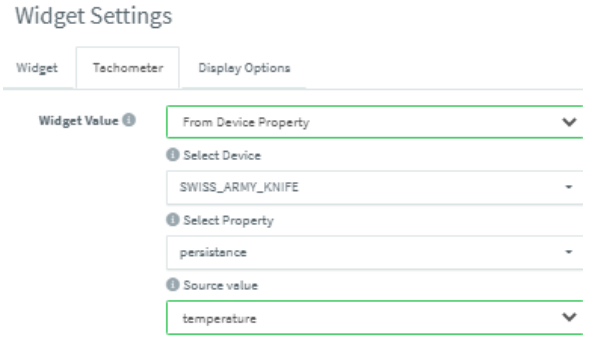
- Dashboard id
- Dashboard name
- Dashboard description

Recommended setup:

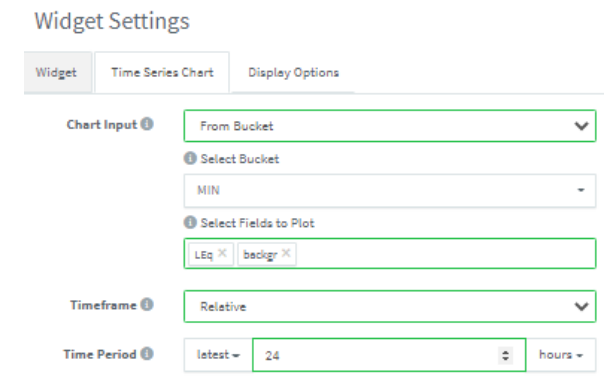


a) in Dashboard REALTIME:
Fast plotter, last 10 minutes, resolution 1 second:
click on +add widget, choose "time series chart ", select input source = "from device resource", select your device, then select the resource "noise", and from these resource, select the fields "limit", then click again just behind "limit" and add the other values "impulse", "noise" "backgr". Choose "stream by device" and a time period of 10 min.
Then select the next tab [display options] remove the option "multiple axes", add the options "fill splines" and "legend"
for "limit" select the color white, for "impulse" select a faint color as e.g. #e4aafd, for "noise", select a strong color (this should be the emphasis) e.g. #c28d19, and for "backgr" at faint color again #eeeb91.

Save your widget and redimension it to your taste, the plot will build on when the dashboard is called.

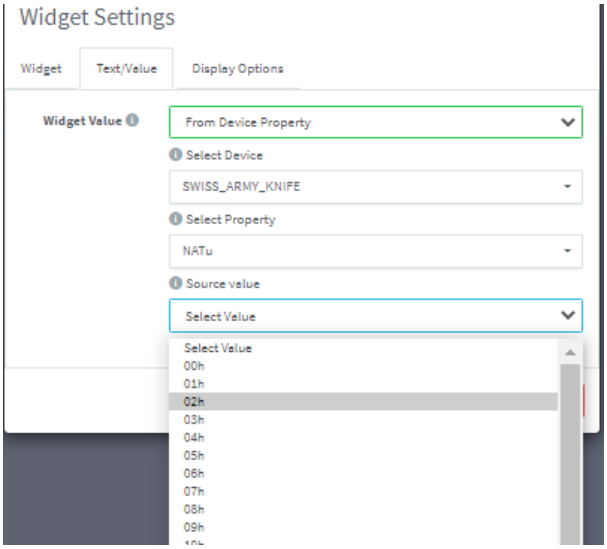


You may add other widgets, e.g. the meteorological data, select input source = "from device property", select your device, then select the resource "persistance", and from these resource, temperature, then with another widget, humidity and so on…
You can experiment with different widgets types and use all the data sources listed in the previous chapter.



b) in Dashboard MIN:
Historic plotter, last 24 hours, resolution 1 minute:
click on +add widget, choose "time series chart ", select input source = "from bucket", select the bucket “MIN”, then select the fields "Leq", then click again just behind "Leq" and add the other values "backgr". Choose timeframe "Relative" and a time period of 24Hours.
Then select the next tab [display options] remove the option "multiple axes", add the options "fill splines" and "legend"
for "noise", select a strong color (this should be the emphasis) e.g. #c28d19, and for "backgr" at faint color #eeeb91.

Save your widget and redimension it to your taste, this plot will be fully available after 24h, then be immediately on screen.



c) In dashboard DAY
Daily exposure/time-averaged/cumulative values.
You may use a series of “value” widgets from the properties NATu and Lequ to make a “table” of hourly values like the one shown on page 9.
You just must work your tail of through all values, good luck!
It is up to you to experiment and design you very own dashboards.