w1retap

Introduction

w1retap is a "one-wire" sensor logging application.

Overview

w1retap supports any number of the following sensors / devices from AAG Electrónica (AAG) based on Dallas Semiconductors devices:

- TAI8520 (DS1820 / DS18S20) Temperature sensors
- TAI8540B (DS2438) Humidity Sensor
- TAI8570 Pressure Sensor (DS2406)
- TAI8575 Rain Gauge (DS2423 Counter)
- TAI8515 Weather Station (DS2423,DS18S20,DS2450)
- DS2409 Microlan coupler
- DS2760 voltage / current / temperature
- DS2450 Quad A/D Converter
- DS2490 USB adaptor
- DS1921 Thermochron, instantaneous temperature only
- DS1923 Hygrochron, instantaneous temperature and humidity only
- DS2480 Serial adaptor
- LinkUSB adaptor.

w1retap also supports a number of other sensors, typically "hobby/build your own" and some products from HobbyBoards:

- SHT11 based humidity sensor
- MPX4115A based pressure sensor (via DS2438) https://hobby.dapj.com/david-bray-1-wire-barometer
- MS-TH Humidity sensors (and temperature) based on DS2438 / Honeywell HiH400). This also supports the (old) Hobby Boards Humidity / Temperature sensor
- The new Hobby Boards Humidity / Temperature sensor
- The Hobby Boards Pressure sensor
- The Hobby Boards Solar sensor (and permutations)
- The Hobby Boards UltraViolet sensor
- The iButton MS-TC Temperature and Current sensor.

w1retap is flexible in the way that 1-wire sensor data is logged; a system of "plugin" modules allow the user to choose the most appropriate logging method. Currently supported logging modules are:

- Sqlite3
- PostgreSQL
- MariaDB (MySQL)
- ODBC
- Text file
- CSV file

- XML file.

w1retap is designed to run on the Linux and FreeBSD operating systems and requires that the interface between the computer and the 1-wire system is either a DS2490 USB adaptor or a DS2480 RS232 serial adaptor (or compatible).

Porting to any other operating system that supports the above hardware, the GCC 'C' compiler and dynamically loadable modules should also be possible. w1retap is known to build and run on arm32, arm64, ia32, riscv64 and x86_64 architectures.

A modified Dallas public domain 1-wire SDK is included in its entirety in the project.

w1retap installation includes a program `w1find` which detects devices on the 1-wire network and may be used as a basis for the sensor configuration table / file.

This document is applicable to the w1retap 1.5.0 (and later).

Organisation of the w1retap project

The w1retap project is organised into a number of sub-directories:

src

Contains the w1retap source code

src/libusblinux300

Holds the (modified) Dallas PD 1-wire SDK

doc

Documentation on configuring and using w1retap

contrib/scripts

Contains some example scripts referenced in this document.

Installation

Prerequisite - Choosing the logging method

Installation of w1retap requires that the software is compiled from source, you may first wish to decide which backend logging modules you are going to use (however the build system will build all those it can on your machine). These are build as loadable modules (shared libraries) and include:

Usage	Configuration Name	Built Module
Sqlite (version 3)	`w1sqlite`	`libw1sqlite.so`
PostgreSQL	`w1pgsql`	`libw1pgsql.so`
MariaDB, MySQL	`w1mysql`	`libw1mysql.so`
ODBC	`w1odbc`	`libw1odbc.so`
Plain Text	`w1file`	`libw1file.so`
CSV File	`w1csv`	`libw1csv.so`
XML File	`w1xml`	`libw1xml.so`

For each of the RDBMS loggers, you will need to have the relevant development package (header files and libraries installed). The file based modules have no external dependencies (other than `libxml2` for `w1xml`), and you can always use `libw1file.so` , as this can also provide fall back configuration data.

USB Package

It is necessary to install the development package for `libusb-1.0`.

Build Process

w1retap uses `meson` and in theory will detect the features that it can build on your machine. Backends can be installed and configured while w1retap is running, so you might as well build all you may ever need, assuming you have the dependencies satisfied.

Build and install

w1retap can be installed to a system directory, or to a user local location.

To configure the w1retap build, issue one of the following commands (a one-off):

```
# user local
meson setup _build --buildtype=release --strip --prefix=~/.local
```

```
# system wide (/usr/bin etc).
meson setup _ubuild --buildtype=release --strip --prefix=/usr/
or
# system wide (/usr/local/bin etc)
meson setup _ubuild --buildtype=release --strip
```

Then to build and install:

```
# user local build
ninja -C _build
ninja -C _build install
# note if you always wish to install, just use the second command, which builds and
installs
```

```
# system build
ninja -C _ubuild
# then to install
sudo ninja -C _ubuild install
```

Configuration

Configuration Essentials

The configuration comprises two areas:

- Configuration of the 1-wire sensors. This may be file based or in a relational database;
- Configuration of the application. The user running w1retap may create a configuration file in their home directory under `.config/w1retap` (`mkdir -p ~/.config/w1retap`), or;
- Alternately, a system wide configuration file, `/etc/defaults/w1retap` may be used.

The environment variable `W1RCFILE` may define the full path of a configuration file. Use of `W1RCFILE` allows alternate configurations for testing.

The `~/.config/w1retap` directory should contain the file `rc` which configures the application and optionally, `sensors` which defines the 1-wire sensors (unless the sensors are defined in a RDBMS). If you are using a data base for logging, it is recommended that you also use it to store the configuration.

Creating the database

If you're using an RDBMS for logging, create the RDBMS from the `docs/mksens.sql` or `docs/mksenst.sql` (depending on how you which to store timestamps) files, for example:

```
$ sqlite3 /var/tmp/sensors.db < mksens.sql
```

```
$ psql
create database w1retap
\c w1retap
\i mksenst.sql
\q
```

```
$ mysql
create database w1retap
use w1retap
source mksens.sql
quit
```

NOTE MySQL uses a somewhat strange SQL syntax, and you may need to modify the template files (or the above MySQL example).

Configuration of sensors

w1retap supports any number of the following sensors from AAG Electrónica (AAG) and others, based on Dallas Semiconductors devices. The following devices require configuration.

- TAI8520 (DS1820 / DS18S20) Temperature sensors
- TAI8540B (DS2438) Humidity Sensor
- TAI8570 Pressure Sensor
- TAI8575 Rain Gauge
- TAI8515 Weather Station (Wind Vane)
- DS2760 voltage / current / temperature
- DS2450 Quad A/D Converter
- DS2409 Microlan coupler
- DS1921 Thermochron, instantaneous temperature only
- DS1923 Hygrochron, instantaneous temperature and humidity only
- SHT11 Humidity sensor
- MPX4115A based pressure sensor ('fronted' by DS2438)
- MS-TH Humidity sensors (and temperature) based on DS2438 / Honeywell HiH400). This also supports the Hobby Boards Humidity / Temperature sensor
- The Hobby Boards Pressure sensor
- The Hobby Boards Solar sensor
- The Hobby Boards UV sensor
- iButton MS-TC Current / Temperature sensor.

Sensors are not completely auto-detected. You need to either populate the w1sensors table in the RDBMS or create a delimited configuration file `~/.config/w1retap/sensors`.

The `w1sensors` table and the `~/.config/w1retap/sensors` file both contain the same information, for the table (sqlite3 quoting):

```
CREATE TABLE `w1sensors` (
 `id` integer NOT NULL PRIMARY KEY AUTOINCREMENT,
 `device` text NOT NULL,
 `type` text,
 `abbrv1` text,
 `name1` text,
 `units1` text,
 `abbrv2` text,
 `name2` text,
 `units2` text,
 `params` text,
 `interval` integer);
);
```

id

  Optional primary key

device

  The device address. If you have sensors from AAG, then the address will be printed on the case, otherwise, you can use the **w1find** program to detect the devices.

type

  A description of the sensor type. This defines how w1retap will access the device. One of:

- `DS1820` (DS1820/DS18S20/DS1920 Temperature sensors)
- `TAI8540` (AAG Humidity sensors)
- `TAI8570` (AAG Pressure sensors)
- `TAI8575` (AAG Rain Gauge)
- `SHT11` (SHT11 Humidity sensors)
- `MPX4115A` ("Bray" barometer)
- `TAI8515` ("Weather" station)
- `DS2409` (Microlan coupler)
- `DS2450` (Quad A / D Converter)
- `DS2438` (raw voltages)
- `HB-BARO` (Hobby Board Barometer)
- `MS-TH` or `HWHIH` (MS-TH / Honeywell humidity sensors)
- `DS2760` voltage / current / temperature sensor (high-precision li+ battery monitor)
- `DS1921` Thermochron
- `DS1923` Hygrochron
- `MS-TC` (iButton Current / Temperature)
- `HB-UV` (Hobby Boards UV)
- `HB-HT` (Hobby Boards HT (new board c. 2015))

name

  An arbitrary name of a function of the device.

abbrv

A unique abbreviation (essentially a key) that identifies the device readings in the database.

units

The units that the device records.

params

Any special parameters used by the application to convert readings from the device to meteorological data. This is typically required for some pressure sensors.

interval

Optional, defines the polling interval for the sensor.

During early development, it was erroneously assumed that each device supports one or two functions, each of these is identified by an arbitrary name, an arbitrary (but unique) abbreviation and the units of measurement that the device records in. The presence of the abbreviation field determines if that specific function is logged. Where a device supports two or more functions, for example humidity and temperature, or pressure and temperature, then it is a requirement that the 'name' field describes the function.

Where a device supports more than two functions, it is just necessary to add any additional definition with the same device name and device type for those additional functions. This allows the voltages from a DS2438 incorporated in a humidity sensor to be logged, or the four functions from a DS2760.

TIP TAI8570 Pressure Sensor. This actually contains two 1-wire devices, we need to specify the address of the "reader" device. As well as being printed on the case, this was the first address found by the `w1find` program. There have been reports that the address printed on the cases of some devices in not the “reader “ device; the solution is simple … `w1find` will find both addresses, if one doesn’t work the try the other one.

So a configuration for this device is:

Key	Value
device	`12FC6B34000000A9`
type	`TAI8570`
abbrv1	`OPRS`
name1	`Pressure`
units1	`hPa`
abbrv2	`OTMP1`
name2	`Temperature`
units2	`°C`

This information may either be stored in a database in the w1sensors table, or in the `.config/w1retap/sensors` text file (as : or | delimited values):

SQL:

```
INSERT INTO w1sensors VALUES(0,'26378851000000AB','DS2438','OTMP1','House
Temperature','°C',NULL,NULL,NULL,NULL,NULL);
INSERT INTO w1sensors VALUES(1,'265066B5000000D2','HB-
BARO','OPRS','Pressure','hPa',NULL,NULL,NULL,'22.4077 900.3',NULL);
INSERT INTO w1sensors VALUES(2,'105EE02301080039','DS1820','STMP1','Soil
Temperature','°C',NULL,NULL,NULL,NULL,NULL);
INSERT INTO w1sensors VALUES(3,'286DA467000000AD','DS1820','GHT','Greenhouse
Temperature','°C',NULL,NULL,NULL,NULL,NULL);
```

```
INSERT INTO w1sensors VALUES(4,'2665E1F1000000A0','_DS2438_','SOLAR','Solar
(Vsens)','mV','SVDD','Solar (vdd)','volts',NULL,NULL);
INSERT INTO w1sensors VALUES(5,'10A942C10008009B','DS1820','OTMP2','Garage
Temperature','°C', NULL,NULL,NULL,NULL,NULL);
INSERT INTO w1sensors VALUES(6,'1D4A1B0B000000E1','DS2423','RGC1','Counter1','tips',
'RGC0','Counter0','tips','16218 2152',NULL);
INSERT INTO w1sensors
VALUES(7,'26FD57E700000052','HWHIH','OHUM','Humidity','%','OTMP0','H
Temperature','°C',NULL,NULL);
INSERT INTO w1sensors VALUES(8,'109F7724010800F1','DS1820','CFRAME1','Cold Frame
Temperature','°C',NULL,NULL,NULL,NULL,NULL);
INSERT INTO w1sensors
VALUES(10,'1F1CE7030000009C','DS2409','MAIN','265066B5000000D2',NULL,'AUX','105EE02301080
039 286DA467000000AD 2665E1F1000000A0',NULL,NULL,NULL);
INSERT INTO w1sensors
VALUES(11,'1FEDCB0300000000','DS2409','MAIN','1D4A1B0B000000E1',NULL,'AUX','10A942C100080
09B',NULL,NULL,NULL);
INSERT INTO w1sensors VALUES(12,'1FF9E60300000093','DS2409','MAIN','109F7724010800F1
',NULL, 'AUX','26FD57E700000052',NULL,NULL,NULL);
INSERT INTO w1sensors VALUES(21,'1D9BB10500000089','_TAI8575_','RAIN0','Counter0','
tips','RAIN1','Counter1','tips',NULL,NULL);
```

Note any `type` starting with an underscore is ignored.

As a `.config/w1retap/sensors` file:

```
26378851000000AB|DS2438|OTMP1|House Temperature|°C||||
265066B5000000D2|HB-BARO|OPRS|Pressure|hPa|||22.4077 900.3|
105EE02301080039|DS1820|STMP1|Soil Temperature|°C||||
286DA467000000AD|DS1820|GHT|Greenhouse Temperature|°C||||
2665E1F1000000A0|_DS2438_|SOLAR|Solar (Vsens)|mV|SVDD|Solar (vdd)|volts||
10A942C10008009B|DS1820|OTMP2|Garage Temperature|°C||||
1D4A1B0B000000E1|DS2423|RGC1|Counter1|tips|RGC0|Counter0|tips|16218 2152|
26FD57E700000052|HWHIH|OHUM|Humidity|%|OTMP0|H Temperature|°C||
109F7724010800F1|DS1820|CFRAME1|Cold Frame Temperature|°C||||
1F1CE7030000009C|DS2409|MAIN|265066B5000000D2||AUX|105EE02301080039 286DA467000000AD
2665E1F1000000A0|||
1FEDCB0300000000|DS2409|MAIN|1D4A1B0B000000E1||AUX|10A942C10008009B|||
1FF9E60300000093|DS2409|MAIN|109F7724010800F1 ||AUX|26FD57E700000052|||
1D9BB10500000089|_TAI8575_|RAIN0|Counter0| tips|RAIN1|Counter1|tips||
```

Note that the greenhouse temperature sensor only has one function, so the `abbrv2`, `name2` and `unit2` fields are not defined (or NULL).

Devices incorporating the DS2438 and the DS2760 (*inter alia*) may provide more than the two functions that the w1retap database schema appears to permit. This has been addressed in w1retap v1.2.2 and later; when w1retap reads the w1sensors table (or configuration file), it will group functions by Device ID and device type. This means that for a DS2760 that supports four functions, we can define the functions using two rows, and all the data will be read in one place, for example:

```
INSERT INTO w1sensors VALUES (NULL, '30EB9B6112000018','DS2760', 'MS_Volts','Moisture
Voltage','V','MS_Current','Moisture Current','A',NULL);
INSERT INTO w1sensors VALUES (NULL, '30EB9B6112000018','DS2760', 'MS_Temp','Moisture
Temperature','°C', 'MS_Accum','Moisture Accumulator','Ahrs',NULL);
```

For a DS2438, there are two options if you want the voltages as well as the 'applied' device (e.g. a TAI8540 humidity sensor); you could just define everything as the 'applied' device, which causes one read of the device, e.g.

```
42|26378851000000AB|TAI8540|OHUM|Humidity|%|OTMP0|Outside Temperature|°C|
43|26378851000000AB|TAI8540|Vdd|Vdd|V||||
44|26378851000000AB|TAI8540|Vad|Vad|V|Vsens|Vsens|mV|
```

or as two devices, which is less efficient, as the device is read twice:

```
17|26378851000000AB|TAI8540|OHUM|Humidity|%|OTMP0|Outside Temperature|°C|
18|26378851000000AB|DS2438|Vdd|Vdd|V||||
19|26378851000000AB|DS2438|Vad|Vad|V|Vsens|Vsens|mV|
```

Coupler / Parameters

For the DS2409 Microlan coupler, HB-Baro or a MPX4115A based pressure sensor, the configuration requires a little more work:

Microlan

For a Microlan device, it is necessary to add an entry for each device that is connected via the coupler. The `w1find` application will display the devices on each branch, but it is necessary to add an entry for each device. For each of these entries, the device field is address of the DS2409 coupler, the abbrv1 field is set to 'MAIN' and the abbrv2 filed is set to 'AUX'. The name1 field is a device id on the main branch, the name2 field is the name of a device on the auxiliary branch. Such an example is shown above. While this is ugly, and an automated tool `w1sensors.rb` with `w1find` can create an initial w1sensors database table template. `w1sensors.rb` is included with w1retap 1.24 and later to address this. Alternately, each coupler may be defined once, and the devices on the main and aux branches listed as space separated lists in the abbrv1 and abbrv2 fields respectively. See the `daria.co.uk-single.sql` and `daria.co.uk-multi.sql` listings in the documentation directory.

MPX4115A

In order to convert the voltage readings from the MPX4115A's DS2438 sensor into pressure (which is assumed linear), values of the slope and offset in an equation:

```
pressure (hPa) = slope x Vout + offset
               where Vout is the sensed (output) voltage.
```

These values will depend on the components used and whether an OpAmp is included in the design. By default slope= `35.95` and offset = `751.08`. As these values probably don't work for any real device, "correct for your setup" values may be provided as a set of space separated numbers in the params field (as many as are necessary for any particular device; not limited to the MPX4115A / DS2438 combination).

HB-BARO

The HobbyBoards barometer works in a similar fashion to the MPX4115A 'Bray' device, in that a slope and offset are used to convert the Vad voltage from the DS2438 into a pressure reading. Please note that:

- The device calibration is defined by the vendor in terms of imperial units (inHg), rather than the SI units (hPa) used in the majority of the world, thus;
- The default slope and offset in w1retap are those from the HB-BARO documentation for sea-level (slope `0.6562`, offset `26.0827`). These, alas, give a value in inHg, which w1retap multiplies by `33.863886` to give hPa.
- Alternatively, you can specify the parameters in SI (for hPa) by multiplying the vendor supplied values by `33.863886`. This makes correcting the calibration by adjustment of offset (the usual case) simpler, as one merely adds or subtracts the hPa difference from the offset.
- The design of the HB-BARO assumes that the altitude correction is embodied in the slope / offset, and w1retap

does not compensate for altitude (it compensates for altitude and temperature for the other barometer devices);

- If you wish to use your own parameters, you must take into account the inHg to hPa conversion factor. For example, if your device, at sea level, consistently over-reads by 6hPa, then you would reduce the offset by (`6.0/33.863886`), giving an offset of `25.90552` compared to the default `26.0827`. If you have specified the calibration values as SI units, then you would just subtract 6 from the SI value (i.e. original offset `883.26`, modified offset `877.26`). In the author's experience, the device is delivered with an accurate calibration voltage, but the offset may need adjusting (by -3 hPa for the author's device);
- For purposes of calibration, by adding an additional configuration entry for the measured voltage (Vad), you can use w1retap to perform the manufacturer's documented calibration steps (see HobbyBoards' web site).

HB Solar

The Hobby Boards Solar sensor is a DS2438 that provides the output of the photo-diode as the 'Vsens' voltage. With the standard HB device, multiplying the millivolts from w1retap by `2.9682` gives W/m^2; albeit possibly an underestimate if you're using parasitic (vice external) power. From a long disappeared internet FAQ:

```
# The formula to get current thru the sense resistor (current
# register volts) / 390 ohms = (sensor current)
# Note: asumes a standard Hobby Boards solar sensor The formula to
# get solar energy is: (sensor current) * 1157598 = W/M^2 (solar
# energy)
# Thusly, w/m^2 = milliVolts * 2.9682
```

DS1921

If a mission is running, the last recorded mission value is returned, if no mission is running, then a forced temperature conversion is run on the device, giving the instantaneous temperature. If the params value for the device is set to a numeric value, then any extant mission is aborted and the instantaneous temperature returned.

DS1923

If the params value for the device is set to a numeric value, then any extant mission is aborted before the device is read.

DS2423, TAI8575

The params value may consist of two integers that are *added* to the readings (so if you want to subtract, make the param values negative). This is useful if you change the counter, but wish to maintain reported values.

HB-UV

The HobbyBoards UV sensor has a number of settings stored in non-volatile memory. These can be set using the `hbuvtest` tool, running `hbuvtest -h` describes how to view or alter the nv-ram settings.

HB-HT

The (new, (2015)) HobbyBoards Humidity and Temperature sensor has a number of settings stored in non-volatile memory. These can be set using the `hbhttest` tool, running `hbhttest -h` describes how to view or alter the nv-ram settings.

An example complex configuration with an MPX4115A and DS2409 is:

```
INSERT INTO w1sensors (device, type, abbrv1, name1, units1, abbrv2, name2, units2,
params) VALUES ('106B89C4000800B9','DS18S20','DS1820 Temp',
'Temperature','°C',NULL,NULL,NULL,NULL);
INSERT INTO w1sensors (device, type, abbrv1, name1, units1, abbrv2, name2, units2,
params) VALUES ('264E1169000000B5','MPX4115A','Baro Press','Pressure','hPa', 'Baro
Temp','Temperature','°C','34.249672152 762.374681772');
INSERT INTO w1sensors (device, type, abbrv1, name1, units1, abbrv2, name2, units2,
params) VALUES ('01F8A3880E0000A2','SHT11','SHT11 RH','Humidity','%','SHT11
Temp','Temperature','°C',NULL);
INSERT INTO w1sensors (device, type, abbrv1, name1, units1, abbrv2, name2, units2,
params) VALUES
('1FCD2D020000007F','DS2409','MAIN','264E1169000000B5',NULL,NULL,NULL,NULL,NULL);
INSERT INTO w1sensors (device, type, abbrv1, name1, units1, abbrv2, name2, units2,
params) VALUES ('1FCD2D020000007F','DS2409',NULL,NULL,NULL,
'AUX','01F8A3880E0000A2',NULL,NULL);
```

In this example, the final sensor is the Microlan coupler, the name fields define a sensor on each branch. The MPX4115A “Bray” barometer uses specific slope and offset parameters from the params field (c.f. the HB_BARO).

The TAI8515 Weather Station from AAG provides temperature and wind speed and direction. These components are provided by three separate one wire devices in the TAI8515, e.g.:

```
201A1B01000000F8 2450:quad a/d converter-> wind direction
10EF161400080056 18S20:high precision digital thermometer -> air temp
1DA273010000005D 2423:4k ram with counter -> wind speed
```

These devices would be defined by three separate entries in the configuration file, for example:

```
INSERT INTO w1sensors VALUES ('10EF161400080056','DS18S20','DS1820 Temp',
'Temperature','°C', NULL,NULL,NULL,NULL);
INSERT INTO w1sensors VALUES ('201A1B01000000F8','TAI8515','WDIR','Wind Direction', '',
NULL,NULL,NULL,NULL);
INSERT INTO w1sensors VALUES ('1DA273010000005D','DS2423','WSPD','Wind Speed', '',
NULL,NULL,NULL,NULL);
```

The temperature will be returned in °C, the direction as an integer in the range 0-15, which you must interpret as a direction N,NNE,NE … NNW etc, and the speed as a counter value. The AAG web site FAQ once provided a formula for converting counts per time interval to wind speed… This Raspberry Pi project offers a calculation that may be a useful starting point.

Summary of device type naming

The following device 'type' keys are recognised in the w1sensors table or sensors configuration file (second parameter).

Type	Alternative	Deprecated	Function
DS1820	DS18S20	Temperature	DS1820,DS18S20, DS1920 temperature
TAI854		Humidity	TAI8540 (DS2438 based) Humidity sensor
TAI8570		Pressure	TAI8570 Pressure sensor (dual DS2406)
DS2423	Counter, TAI8575	RainGauge	DS2423 Counters
MPX4115A		Bray	Barometer based on MPX4415A (with DS2438)
SHT11			SHT11 based humidity sensor

Type	Alternative	Deprecated	Function
TAI8515	Windvane, Weather vane		AAG Weather station wind direction sensor (DS2450 based)
DS2490	Coupler		DS2409 Microlan Coupler
DS2438		Voltage	DS2438 as a voltage sensor
HB-BARO	HB_BARO		Hobby Boards Barometer
MS-TH	HWHIH		MS-TH Humidity Sensor (also works for Hobby Boards Honeywell HIH4000 based humidity sensor)
DS2760			DS2760 voltage / current / temperature
DS2450			Quad A / D Converter
MS-TC			iButton current and temperature
DS1923			DS1923 Hygrochron, instantaneous temperature and humidity only
HB-UV	HB_UV		Hobby Boards ultra-violet

The following device name keys are required in order to get data from multi-function sensors stored correctly in the database. The match is partial; the quoted text must occur somewhere in the 'nameN' field. For w1retap v1.24 and later, the quoted text may alternately be given precisely as the abbrvN field. The match is case independent in both instances.

Device	Name	Usage
MPX4115A / Bray or HB-BARO or TAI8570	Pres	Pressure Value
MPX4115A / Bray or HB-BARO or TAI8570	Temp	Temperature Value
SHT11 or TAI8540 or MS-TH / HWHIH	Humidity	Humidity Value
	Temp	Temperature Value
DS2438 (or TAI8540 or other DS2438 based sensor (Bray, HB-BARO etc))	Vdd	Supply voltage
	Vad	Output voltage
	Vsens	Sensed voltage
	Temp	Temperature Value
DS2760	Voltage	Voltage (V)
	Current	Current (I)
	Accumulator	Amp hour
	Temp	Temperature Value
MS-TC (DS2438 based iButton current sensor) (Note 1)	current	Measured current
	Vdd	Supply voltage
	Vad	Output voltage
	vsens	Sensed voltage
	Temp	Temperature Value

Device	Name	Usage
DS1921	Temp	Temperature Value
DS1923	Temp	Temperature Value
	Humidity	Humidity Value
HB-UV	Temp	Temperature Value
	uv	UV value
	ultra	UV value
	violet	UV value
DS2450 (Note 2)	ADx	i.e. ADA, ADB, ADC, ADD.

Notes

1. MS-TC : Only the current and temperature are useful outputs
2. DS2450 : Two entries are necessary to define all the A/D converters. This is only necessary for a standalone DS2450. The device included in the AGG Wind-vane is read directly by the TAI8515 item.

Database and storage strategies

The default database schema writes a record for each reading (a tuple of date, sensor name and the sensor value). From w1retap 1.41, it is also possible to use a document oriented strategy with PostgreSQL and SQLite databases.

Database storage and performance configurations

The author initially ran w1retap on a record orientated PostgreSQL implementation, then for six months on mongodb (w1retap 1.4.0, now discontinued), and is now back to PostgreSQL, with a document orientated storage strategy. There are a number of advantages and disadvantages to each scheme, a document oriented strategy greatly reduces the number of records (I have 10 sensors, so for sensor pass, that's one record in the document oriented database vice ten in the record orientated store. Likewise, for display, retrieving all records within a set time period requires fewer records to be returned; as a modern scripting languages handle JSON efficiently, this can result in a simpler application. The only downside is perhaps in retrieving a particular record on a none-timestamped criteria.

For example the coldest ever outside temperature reading (OTMP0 here), using the record schema:

```
select * from readings where name='OTMP0' order by value asc limit 1;
```

It is pretty straight forward, whereas the document search is (perhaps) somewhat more convoluted in an RDBMS with a document storage.

```
wx=# select * from readings order by (wxdata->'OTMP0')::float limit 1;
 2010-12-26 07:16:00+00 | {"GHT": -6.0625, "OHUM": 93.447586, "OPRS": 1030.93689, "RGC0":
19793, "RGC1": 16219, "OTMP0": -8.0625, "OTMP1": 16.65625, "OTMP2": -3.8125, "SOLAR": 0,
"STMP1": 3.0625, "CFRAME1": -5.1875}
```

Configuring the RDBMS record type

For a record storage schema, the readings table requires the following:

```
CREATE TABLE readings (
 date timestamp with time zone NOT NULL,
 name text NOT NULL,
 value double precision ,
 id serial
);
```

For the document oriented schema:

```
CREATE TABLE readings (
 date timestamp with time zone NOT NULL,
 wxdata json
);
```

And optionally

```
CREATE INDEX readings_date ON readings USING btree (date);
ALTER TABLE ONLY readings ADD CONSTRAINT reading_sanity UNIQUE (date, name); -- single
record
ALTER TABLE ONLY readings ADD CONSTRAINT readings_pkey PRIMARY KEY (date); -- document
orientated
```

For SQLite, the json data type may be replaced by a type of text.

NOTE The w1retap application introspects the readings table at start-up to determine the storage strategy; it is therefore advisable to use the default table and column names as above.

Using per-sensor “readings” tables

The default database configuration creates a single table `readings` with columns of `date`, name (i.e. the `abbrv1` and `abbrv2` values from the `w1sensors` table) and `value`, the actual data which is coerced to a double precision value.

Some users may prefer to have a per sensor data (readings) table, which is possible if you use the PostgreSQL database backend (patches for other database backends are welcome). In order to use per-sensor readings tables, it is necessary to:

- The `abbrv1/2` field is defined with a leading > character. The text after the > is taken as the table name;
- The table is created with fields of `date` and `value`. For sensors returning integer data (WindVane and Counters), the value field may be an integer type, otherwise it should be double precision.

It is possible to mix the 'one monolithic table' and 'one table per sensor' modes, by definition of the `abbrv1`/`abbrv2` fields.

Using w1find to scan the 1-wire bus

In order to create the sensor configuration table, `w1sensors`, (or a text file), it is necessary to know the devices on the 1-wire bus. The `w1find` program will find this information. It does not create the configuration table or file, as a particular 1-wire sensor may be employed by a number of different devices; rather it provides a template that the user may edit to her requirement.

e.g. For these sensors:

```
w1find DS2490-1

(1) 26378851000000AB    2438:smart battery monitor
(2) 817E84240000008B    :Serial ID Button
(3) 1F1CE7030000009C    2409:microlan coupler
    (Main.1) 265066B5000000D2    2438:smart battery monitor
    ( Aux.1) 105EE02301080039    18S20:high precision digital thermometer
    ( Aux.2) 286DA467000000AD    18B20:programmable resolution digital thermometer
(4) 1FF9E60300000093    2409:microlan coupler
    (Main.1) 109F7724010800F1    18S20:high precision digital thermometer
    ( Aux.1) 10823BCB010800FB    18S20:high precision digital thermometer
    ( Aux.2) 26FD57E700000052    2438:smart battery monitor
(5) 1FEDCB0300000000    2409:microlan coupler
    (Main.1) 1D4A1B0B000000E1    2423:4k ram with counter
    ( Aux.1) 10A942C10008009B    18S20:high precision digital thermometer
```

Using `w1sensors.rb` with `w1find` to create an initial w1sensors database table

Configuring the `w1sensors` table requires some care , particularly if you have a large number of sensors, one or more DS2409 couplers, or you are new to one wire devices. This is made more complex by the fact that devices may supply multiple functions, or may serve a function other than the primary function of the device (a DS2438 voltage sensor may serve as a pressure or humidity sensor, a DS2423 counter may serve as the wind speed indicator).

The flexibility of one wire devices also means it is very difficult to automatically probe the device chain and ascertain precisely what the function of every device might be, and manual confirmation and final configuration of device functions will be required.

It is possible to build an initial configuration for the w1sensors database table (as a set of SQL INSERT statements (or a '|' delimited file with --file-based-config)) using the `w1find` program in conjunction with the `w1sensors.rb` script. The output is written to a file or STDOUT (which may be in turn piped to an RDBMS). Any unrecognised sensors are listed to STDERR.

```
$ w1sensors.rb -?
w1sensors.rb [options] [file|stdin]
e.g. w1find DS2490-1 | w1sensors.rb -o /tmp/w1_sensors-setup.sql
w1find DS2490-1 | w1sensors.rb | sqlite3 sensors.db
-f, --file-based-config
-o, --output FILE
-?, --help
Show this message
```

For some old sensors:

```
$ w1find DS2490-1
(1) 105EE02301080039 18S20:high precision digital thermometer
(2) 10A942C10008009B 18S20:high precision digital thermometer
(3) 286DA467000000AD 18B20:programmable resolution digital thermometer
(4) 12FC6B34000000A9 2406:dual addressable switch plus 1k memory
(5) 121B4A3400000030 2406:dual addressable switch plus 1k memory
(6) 26378851000000AB 2438:smart battery monitor
(7) 817E84240000008B :Serial ID Button
(8) 1D9BB10500000089 2423:4k ram with counter
```

Piping the results into w1sensors.rb gives the following SQL statements:

```
INSERT into w1sensors values ('105EE02301080039','DS1820','TMP_1','Temperature #1','°C',
 NULL,NULL,NULL,NULL);
INSERT into w1sensors values ('10A942C10008009B','DS1820','TMP_2','Temperature #2','°C',
 NULL,NULL,NULL,NULL);
INSERT into w1sensors values ('286DA467000000AD','DS1820','TMP_3','Temperature #3','°C',
 NULL,NULL,NULL,NULL);
```

```
INSERT into w1sensors values ('12FC6B34000000A9','TAI8570','Pressure_4', 'Pressure
4','hPa',
 'TMP_4','Temperature #4','°C',NULL);
INSERT into w1sensors values ('26378851000000AB','DS2438','VDD_6', 'VDD 6','V',
 'TMP_6','Temperature #6','°C',NULL);
INSERT into w1sensors values ('26378851000000AB','DS2438','VAD_6', 'VAD 6','V',
 'Vsens_6','Vsens #6','mV',NULL);
INSERT into w1sensors values ('1D9BB10500000089','TAI8575','CountA_7','CounterA
#7','pulses',
 'CountB_7','CounterB #7','pulses',NULL);
```

Some editing provides the required `w1sensors` table, noting that the DS2438 is the front end for a TAI8540 humidity sensor.

```
INSERT INTO w1sensors VALUES('105EE02301080039','DS1820','STMP1','Soil Temperature','°C',
 NULL,NULL,NULL,NULL);
INSERT INTO w1sensors VALUES('286DA467000000AD','DS1820','GHT','Greenhouse
Temperature','°C',
 NULL,NULL,NULL,NULL);
INSERT INTO w1sensors VALUES('12FC6B34000000A9','TAI8570','OPRS','Pressure','hPa',
 'OTMP1','Inside Temperature','°C',NULL);
INSERT INTO w1sensors VALUES('26378851000000AB','TAI8540','OHUM','Humidity','%',
 'OTMP0','Outside Temperature','°C',NULL);
INSERT INTO w1sensors VALUES('1D9BB10500000089','TAI8575','RGC0','Counter0','tips',
 'RGC1','Counter1','tips',NULL);
INSERT INTO w1sensors VALUES('10A942C10008009B','DS1820','OTMP2','Garage
Temperature','°C',
 NULL,NULL,NULL,NULL);
INSERT INTO w1sensors VALUES('1093AEC100080042','__DS1820','ITMP1','Propagator1
Temperature','°C',
 NULL,NULL,NULL,NULL);
INSERT INTO w1sensors VALUES('10E3EA23010800C9','__DS1820','ITMP2','Propagator2
Temperature','°C',
 NULL,NULL,NULL,NULL);
```

(Note also two seasonal sensors are “commented out” by prefixing the device type with two underscores).

Configuring the w1retap software.

The main configuration of the application is done via the `~/.config/w1retap/rc` file (or `/etc/default/w1retap` if the user’s file doesn’t exist).

Some of the options may also be specified on the command line when w1retap is invoked. This defines how w1retap obtains the sensor configuration and how it performs the data logging, for example:

```
#Init file
#init = w1sqlite=/var/tmp/sensors.db
#log = w1sqlite=/var/tmp/sensors.db
#init = w1odbc=DSN=w1retap
init = w1pgsql=dbname=w1retap user=postgres
#init = w1file
#log = w1xml=/tmp/xmllog.txt
#log = w1csv=/tmp/csvlog.txt
log = w1pgsql=dbname=w1retap user=postgres
log = w1file = |/usr/local/bin/pert-log.rb
```

```
#log = w1mysql=dbname=w1retap user=jrh host=kanga password=ohsososecret
#timestamp = 1
altitude = 19
device = DS2490-1
#device = /dev/ttyS0
```

Where the keys are:

init

The initialisation data for the sensors, e.g. a database with a `w1sensors` table, or a file. The value part is the name of a plugin and optionally, parameters (see below).

log

The log database to the readings table, or a file for a file data sink. The value part is the name of a plugin and optionally, parameters (see the section Log and Init options).

device

The name of the interface device. For Linux, using the standard USB interface, this defaults to `DS2490-1`, for a serial device `/dev/ttySn` where “n” represents a digit (or `/dev/ttyUSB0` for a USB/Serial adaptor.

delay

The delay between successive reading of the the 1-wire bus. All sensors are read in one hit unless Per-sensor polling frequency is defined.

daemonise

If set to 1, w1retap detaches and runs in the background

altitude

If the altitude is defined (in *metres*), above mean sea level (MSL), then pressure readings are normalised to MSL, otherwise you get the raw, uncorrected value.

timestamp

If timestamp is set to a non-zero integer value, then the time of the observation in the database `readings` table (field name `date`) will be stored as SQL TIMESTAMP data. The default is that the `date` field is stored as an integer number of seconds since 01 January 1970, UTC (Unix epoch values, also known as `time_t`). You must ensure that your database table is configured appropriately. If you choose TIMESTAMPs, then you need to consider if this will cause you any time-zone / summer time / daylight saving issues.

logtemp

By default, w1retap writes the latest sensor values to a file `/tmp/.w1retap.dat`. If you set logtemp to 0, this file is not updated. The file contains, for each sensor, the abbreviation and value and a time stamp (Unix epoch `time_t` and ISO date format):

```
GHT=23.75 °C
OHUM=75.95 %
OTMP0=19.50 °C
OPRS=1012.97 hPa
OTMP1=20.23 °C
RGC0=3517.00 tips
RGC1=3481.00 tips
udate=1122818880
date=2005-07-31T15:08:00+0100
```

Just an overcast Sunday afternoon in July

log_delimter

The delimiter for `w1file` logs is by default a space. You can override this here, '\' escapes are recognised, so '\t' is `TAB`.

log_time_t

If set to yes, the final field for `w1file` logs is the time_t (numeric seconds since the epoch) log time.

temp_scan

The time in milliseconds to allow DS1820 type sensors to acquire the temperature before the device is read. By default, `w1retap` uses a conservative value of 1000 (1 second). The data sheet claims the conversion should not exceed 750ms, which is the value the author has used with success.

pressure_reduction_temp

By default, `w1retap` uses a QFF pressure reduction model to reduce the pressure to mean sea level (if the **altitude** key (as above) is defined. If you also set the **pressure_reduction_temp** key to a value in degrees Celsius (°C), then this fixed temperature is used (the QNH model). A value of 15 is the ISA (International Standard Atmosphere) value used for some aviation reports.

force_utc

If set to true, then database and other timestamps are as UTC rather than local time. This setting may be useful if your database is less adept than PostgreSQL in storing time stamps with time zones.

Only the first `init` entry is used; multiple `log` entries may be given and are all logged to in the order defined.

Log and Init options

For the `log` and `init` options, the information supplied has two parts, separated by an equals sign. The name of the plugin handling that information and any additional information. For a file based plugin, this will be the file name and for a database, the name of the database and any access control parameters.

For each plugin, the usage and parameters are:

w1file

This module provides basic file system access for configuration and logging. If used as a `init` parameter, it reads `~/.config/w1retap/sensors` (or supplied filename) for sensor information as described for file based sensor configuration.

```
init = w1file
```

```
init = w1file=/etc/w1sensors.dat
```

The first case assumes `~/.config/w1retap/sensors` contains the configuration data, the second explicitly reads `/etc/w1sensors.dat`.

If used as a `log` parameter, it writes one entry per line to STDOUT or a supplied file name:

```
log = w1file
log = w1file=/tmp/w1file.log
```

The data output is in the format (date abbreviation value units):

```
2005-07-29T18:11:28+0100 GHT 20.312500 ºC
2005-07-29T18:11:28+0100 OHUM 74.050064 %
2005-07-29T18:11:28+0100 OTMP0 17.687500 ºC
```

```
2005-07-29T18:11:28+0100 OPRS 1009.950562 hPa
2005-07-29T18:11:28+0100 OTMP1 18.510059 °C
2005-07-29T18:11:28+0100 RGC0 3496.000000 tips
2005-07-29T18:11:28+0100 RGC1 3460.000000 tips
```

If the file name begins with a pipe symbol (|), then it is taken as the name of an application that accepts the data on standard input. This might be used to update the database with derived (calculated) values, or drive an additional display device (e.g. via `pert-log.rb`, which drives a Pertelian LCD display via the `pertd2` program).

```
log = w1pgsql=dbname=w1retap user=postgres
log = w1file=|/usr/local/bin/pert-log.rb
```

The file `wetbulb-snow.rb` shows how a piped script can be used to update the database with derived values from the current set of readings (wet bulb temperature and snow height).

It should be noted that *piped scripts are run synchronously by the w1retap application*. This means that the scripts (in total) should not take longer to execute that the w1retap cycle period, and *care must be taken to ensure that the scripts cannot hang or block* for indefinite periods, as this would cause w1retap also to block and subsequent readings would be lost. As an example, the `pert-log.rb` script takes great care to use non-blocking I/O to ensure that any hang writing to the `pertd` FIFO cannot cause the main w1retap application to hang.

See also the configuration file `log_delimiter` and `log_time_t` entries, as these can affect the format of this log.

w1xml

This module provides basic file system access for logging only. It writes an XML file to STDOUT or a supplied file name:

```
log = w1xml
log = w1xml=/tmp/w1xml.log
```

```
<?xml version="1.0" encoding="utf-8"?>
<report timestamp="2005-08-01T19:51:45+0100" unixepoch="1122922305">
 <sensor name="GHT" value="17.0625" units="°C"></sensor>
 <sensor name="OHUM" value="96.4636" units="%"></sensor>
 <sensor name="OTMP0" value="16.2500" units="°C"></sensor>
 <sensor name="OPRS" value="1017.8268" units="hPa"></sensor>
 <sensor name="OTMP1" value="16.9916" units="°C"></sensor>
 <sensor name="RGC0" value="3544.0000" units="tips"></sensor>
 <sensor name="RGC1" value="3508.0000" units="tips"></sensor>
</report>
```

w1retap uses `libxml2` to write out syntactically correct XML. This also means that (a) there is a dependency on `libxml2` and (b) the input, including any values (such as names and units) in the database must be valid UTF-8. The file name may be prefixed with a pipe symbol to pipe the data to another program (as for `w1file`).

w1csv

This module provides basic file system access for logging only. It writes an CSV file to STDOUT or a supplied file name:

```
log = w1csv
log = w1csv=/tmp/w1data.csv
```

The data output is in the format of a timestamp followed by abbreviations, values and units (all on one line):

```
"2005-08-01T19:51:45+0100", "GHT", 17.062500, "°C", "OHUM", 96.463608, "%", "OTMP0",
16.250000, "°C", "OPRS", 1017.826843, "hPa", "OTMP1", 16.991602, "°C", "RGC0",
3544.000000, "tips", "RGC1", 3508.000000, "tips"
```

The file name may be prefixed with a pipe symbol to pipe the data to another program (as for `w1file`).

w1sqlite

This module provides database access for configuration and logging using a Sqlite3 RDBMS. If used as a `init` parameter, it reads the `w1sensors` table for sensor information as described for RDBMS based sensor configuration.

```
init = w1sqlite=/var/db/sensors.db
```

The name of the database is a mandatory parameter.

If used as a `log` parameter, it writes data to the readings table.

```
log = w1sqlite=/var/db/sensors.db
```

The data is logged as (date, abbreviation, value):

```
$ sqlite3 /var/tmp/sensors.db
SQLite version 3.2.2
Enter ".help" for instructions
sqlite> select * from readings order by date desc limit 1;
1122735840|GHT|25.6251122735840|OHUM|87.68514251708981122735840|OTMP0|18.3437511227358
40|OPRS|1009.779724121091122735840|OTMP1|19.12314414978031122735840|RGC0|3498112273584
0|RGC1|3462
```

Where `date` is the unix epoch time (`time_t`), seconds since 00:00:00 1 Jan 1970 UTC.

w1pqsql

This module provides database system access for configuration and logging using a PostgreSQL RDBMS. If used as a `init` parameter, it reads the `w1sensors` table for sensor information as described for RDBMS based sensor configuration.

```
init = w1pgsql=dbname=w1retap user=postgres
```

The name of the database is a mandatory parameter, followed by optional parameters (dbname, `host`, `user`, `password`).

If used as a `log` parameter, it writes data to the `readings` table.

```
log = w1pgsql=dbname=w1retap user=postgres
```

By default, the data is logged as (`date abbreviation value`) [see `sqlite` example]. The Postgresql driver also offers document database and "one table per sensor" support, see Configuring the RDBMS record type for details.

w1mysql

This module provides database system access for configuration and logging using a Maria (or MySQL) RDBMS. If used as a `init` parameter, it reads the `w1sensors` table for sensor information as described for RDBMS based sensor configuration.

```
init = w1mysql=dbname=w1retap user=w1retap host=kanga
```

The name of the database is a mandatory parameter, followed by optional parameters dbname - name of the database, `user` – username, `password` - user’s password, `host` - database server.

If used as a 'log' parameter, it writes data to the `readings` table.

```
log = w1mysql=dbname=w1retap user=jrh host=kanga`
```

The data is logged as (`date abbreviation value`) (see sqlite example).

w1odbc

This module provides database system access for configuration and logging using ODBC RDBMS. It may thus be used for any database for which there is no specific w1retap module, but you have an ODBC driver. If used as a 'init' parameter, it reads the w1sensors table for sensor information as described for RDBMS based sensor configuration.

```
init = w1odbc=DSN=W1RETAP
```

The DSN of the database is a mandatory parameter.

If used as a 'log' parameter, it writes data to the readings table.

```
log = w1odbc=DSN=W1RETAP
```

The data is logged as (`date abbreviation value`) [see sqlite example].

NOTE Most operating system libraries will do the right thing such that `w1sqlite` will be tried as `libw1sqlite.so`. Due to the deprecation of the g_module_find_path`()` function with no documented replacement, this should not be relied upon.

Running w1retap

w1retap is started from the command line (shell script) or as a boot task (e.g. `systemd` etc). Assuming the permissions of the device (usb/serial) allow non-privileged access, it requires no special privileges and may be run from a normal user account.

If you are using a USB adaptor, it may be necessary to ensure that your user can access (has read and write access) to the USB device. Please see the udev `45-w1retap.rules` file in the documentation directory (this file may be added to `/etc/udev/rules.d` if desired).

w1retap accepts the following command options:

```
$ w1retap --help
Usage:
  w1retap [OPTION...] - w1retap

Help Options:
  -h, --help                  Show help options

Application Options:
  -w, --wait                  At startup, wait until next interval
  -1, --once-only             Read once and exit
  -R, --release-interface     Release the 1Wire interface between reads
  -d, --daemonise              Daemonise (background) application
  -T, --no-tmp-log             Disables /tmp/.w1retap.dat logging
  -l, --tmp-log-name=FILE      Names logging file (/tmp/.w1retap.dat)
  -i, --interface=DEVICE       Interface device
  -t, --cycle-time=SECS        Time (secs) between device readings
  -N, --dont-read              Don't read sensors (for debugging)
  -v, --verbose                Verbose messages
  -o, --vane-offset=VAL        Value for N for weather vane (0-15)
  -V, --version                Display version number (and exit)
  -s, --simulate               Simulate readings (for testing, not yet implemented)
  -u, --use-utc                Store dates as UTC (vice localtime)
  -r, --report-log=FILE        Report log file
```

Note `w1retap -Nv` will dump out the configuration.

Other Database elements

Rate Limiting

Occasionally a sensors may give a wildly inaccurate reading. In order to prevent these from polluting the database, a concept of rating limiting is implemented. This requires a table `ratelimit` exists, and contains the sensor abbreviation and the maximum acceptable rate in `units/minute`,`min` and `max` values. For example:

```
CREATE TABLE ratelimit ( name text, value real, rmin real, rmax real );
INSERT INTO ratelimit (name, value, rmin, rmax) VALUES ('GHT', 2.5, NULL, NULL);
INSERT INTO ratelimit (name, value, rmin, rmax) VALUES ('OTMP0', 2.5, NULL, NULL);
INSERT INTO ratelimit (name, value, rmin, rmax) VALUES ('OTMP1', 2.5, NULL, NULL);
INSERT INTO ratelimit (name, value, rmin, rmax) VALUES ('OPRS', 100, 800, 1200);
INSERT INTO ratelimit (name, value, rmin, rmax) VALUES ('RGC0', 50, NULL, NULL);
INSERT INTO ratelimit (name, value, rmin, rmax) VALUES ('RGC1', 50, NULL, NULL);
INSERT INTO ratelimit (name, value, rmin, rmax) VALUES ('OTMP2', 2.5, -10, 50);
INSERT INTO ratelimit (name, value, rmin, rmax) VALUES ('OHUM', 7, 0, 100.04);
```

The values are such that they would not normally be seen, but are less than the obviously bizarre rogue value seen very rarely.

Per-sensor polling frequency

w1retap polls all the sensors at the same frequency, by default 120 seconds, or the delay value from the configuration file or the `-t` command line argument. A per-sensor value may be applied, by creating a column in the `w1sensors` table called `interval`.

```
alter table w1sensors add column interval integer; -- postgres, sqlite3
alter table w1sensors add column `interval` int; -- mysql
```

This column should contain the polling interval for that sensor in seconds, where a value other than the default is required, e.g. `update w1sensors set interval=10 where device='286DA467000000AD';`.

A few arbitrary rules apply, largely to simplify the implementation:

- The minimum interval is 10 seconds (the 'wake up' time)
- The maximum interval is the delay / `-t SECS` parameter (default 120s) — the 'cycle' time
- If the column is omitted, or the value is NULL or zero, then the default delay value is used
- If the /tmp log file is used, it is only written when all sensors are read (the 'cycle' value)

- w1retap will calculate its wake up value and cycle value, using the highest common factor and lowest common multiple of the individual polling intervals. So if sensors had polling intervals of 10,20,30,40 and 60 seconds, the wake up value would be 10s and the cycle value would be 120s
- The algorithms may not be robust in the face of “unreasonable” (by my definition) values, but should work for the majority of reasonable cases.

Summary of configuration

While the configuration may seem, at first reading, to be complex or confusing, it is a number of logical steps:

- Decide on where you want to store the sensor definition and logged data, a relational database is recommended
- Create `~/.config/w1retap/rc` (or `/etc/defaults/w1retap`) defining the sensors (`init=xxxx`), and data logging (`log=xxxx`) configuration
- Create any necessary RDBMS tables, using the supplied scripts as a template
- Populate the `init=xxxx` definitions, using `w1find`, maybe in conjunction with `w1sensors.rb`
- If you are using the USB one wire device, please see the USB configurations in the documentation directory. It will be necessary to install a udev rule to manage access to the USB device and add the user to the `w1retap` group, unless you run `w1retap` as root, which is neither necessary nor recommended on Linux.

w1retap dependencies

w1retap has a few dependencies on other libraries in order to build the software, in particular `glib-2.0` and `libusb-1.0`. If you’re using Debian or derivative system, then the following will get you started:

```
apt-get install build-essential libglib2.0-dev libusb-1.0-0-dev libc-dev
```

In addition, you will need the development packages for any database you require, e.g. `libsqlite3-dev`, `libxml2-dev`.

Viewing the data

`wplot.rb` (and other) scripts, the Gnome applet and other visualisation tools are no longer distributed by default, these tools demonstrate:

- Building a web page
- Send data to Wunderground.com and other aggregators
- Provided a static JSON document of current conditions. The latter format is read by the w1retap GNOME applet.
- Provided a Gnome shell applet.

Please ping the author if you’re interested in any additional artefacts mentioned in this document.

Credits

Thanks to:

- Mihail Peltekov provided `ssh` access to zlatograd.com, which allowed me to develop the DS2480, DS2409, SHT11 and MPX4115A device support.
- William R Sowerbutts provided a patch to allow field order independent PgSQL logging, the TAI8515 code and the 'one table per sensor' PgSQL logging code, as well as other patches, including robust DS2409 handling.
- Hans van den Boogert kindly donated a Hobby Boards solar sensor.
- Leland Helgerson kindly donated a pair of DS1921 ibutton sensors.

- Peter Parsons contributed the basis of the MS-TC code.
- Andrew Ford lent me a LinkUSB adaptor, which worked out of the box for me (Andrew was not so fortunate, but this is likely a host hardware problem).
- Dave Johns lent me a Hobby Boards UV sensor for the development of that interface.
- Graeme Gemmill lent me the new Hobby Boards HT sensor for the development of that interface.
- Thomas Stewart kindly maintains a Debian package.

Other users have provided bug reports, requests for new sensors and other inspiration.

Daria Hudson started this by requiring a temperature sensor be fitted in her greenhouse and has graciously allowed me pursue my interest in 1-wire weather stations since then. She also allows me to (ab)use her vanity domain.

Author / contact

w1retap is (c) Jonathan Hudson. It is released (mainly) under the GNU Public licence.