W1RETAP

A One-wire sensor logging application

Jonathan Hudson
<jh+w1retap@daria.co.uk>

Introduction

Overview

W1RETAP is a system for logging data from 1-wire sensors to either a relational database or files (or combination thereof).

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;
- DS2490 USB adaptor;
- DS2480 Serial adaptor.

W1RETAP also supports a number of other sensors, typically "hobby/build your own":

- SHT11 based humidity sensor http://home.hetnet.nl/~thomas_7/1Wire/1-WireIOPort.html ;
- MPX4115A based pressure sensor ('fronted' by DS2438) http://home.comcast.net/~andrew.g.miller/barometer/ .

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:

- Sqlite (version 3);
- PostgreSQL;
- MySQL;
- ODBC;
- Text file;
- CSV file;
- XML file.

W1RETAP is designed to run on the Linux operating system and assumes that the

interface between the computer and the 1-wire system is either a DS2490 USB adaptor or a DS2480 RS232 serial adaptor.

Porting to any other operating system that supports the Dallas SDK, the gcc compiler and dynamically loadable modules should also be possible.

A modified Dallas public domain 1-wire SDK is included in its entirety, and may be built from `makefile.shrlib`. This will build all the Dallas sample applications, which may be useful for troubleshooting.

The standard **W1RETAP** installation includes a program w1`find` which detects devices on the 1-wire network and may be used as a basis for the sensor configuration table /file.

W1RETAP does not in itself offer any graphical user interface, however there is a `contrib` directory that contains scripts to build web pages, and a GNOME panel applet.

Unless otherwise indicated, the software is released under the GNU Public Licence.

Organisation of the w1retap release

The **W1RETAP** release is organised into a number of sub-directories:

This distribution is organised as:

src	Contains the **W1RETAP** software.
src/libusblinux300	The Dallas PD 1-wire SDK
doc	Documentation on configuring and using **W1RETAP**
contrib	Various scripts and applications, including a web page builder, a wunderground.com reporting script, RSS feeder, and GNOME panel applet.

Installation

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:

Type	Name	Module
Sqlite (version 3)	w1sqlite	`libw1sqlite.so`
PostgreSQL	w1pgsql	`libw1pgsql.so`
MySQL	w1mysql	`libw1mysql.so`
ODBC	w1odbc	`libw1odbc.so`
Text file	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 files (header files and libraries installed). The file based modules have no external dependencies, and you can always use `libw1file.so`, as this can provide fall back configuration data.

Build Process

W1RETAP uses `autoconf` 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

Issue the following commands:

```
$ ./configure
$ make
$ sudo make install  # (or run as root).
```

`make install` installs the **W1RETAP** application into `/usr/local/bin` and its plugins to `/usr/local/lib/w1retap/` with the default `autoconf prefix` setting. To change this, run ./configure with your preferred settings, for example:

```
$ ./configure –prefix=/usr
```

will install the application into `/usr/bin` and the plugin modules to `/usr/lib/w1retap/`.

You can force the installed programs to be stripped with `make install-strip`, however, as far as the author can ascertain, this does not strip the modules. You can force the plugin modules to be stripped with `STRIP_LIBS=yes`, e.g.:

```
$ sudo make install-strip  STRIP_LIBS=yes prefix=/usr
```

In addition to the RDBMS shared libraries, shared libraries are built for USB and RS232 device access.

Create configuration of the database (if used)

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** needs to create a set of configuration files in their home directory under `.config/w1retap`.

  ```
  $ mkdir -p ~/.config/w1retap
  ```

The `~/.config/w1retap` directory must contain the file `rc` which configures the application, and optionally `applet` which configures the GNOME applet (see `contrib/w1temp` for details) 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 (recommended), create the RDBMS from the `docs/mksens.sql` or `docs/mksenst.sql` (depending on how you which to store timestamps) files.

e.g.

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

or

```
$ psql -U USERNAME template1
template1=# create database w1retap;
CREATE DATABASE
template1=# \c w1retap;
You are now connected to database "w1retap".
w1retap=# \i mksenst.sql
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
w1retap=# \q
```

or

```
mysql -u USERNAME -p
mysql> create database w1retap;
Query OK, 1 row affected (0.02 sec)
mysql> use w1retap
Database changed
mysql> source mksens.sql
Query OK, 0 rows affected (0.01 sec)
Query OK, 0 rows affected (0.13 sec)
Query OK, 0 rows affected (0.02 sec)
Query OK, 0 rows affected (0.09 sec)
mysql> quit
```

Note that MySQL uses a somewhat strange SQL syntax, and you may need to modify the template files.

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);
- DS2409 Microlan coupler;

- SHT11 Humidity sensor;
- MPX4115A based pressure sensor ('fronted' by DS2438).

Unlike some advanced 1-wire applications, your sensors are not 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, from the table:

```
CREATE TABLE w1sensors
(
      device text,
      type text,
      abbrv1 text,
      name1 text,
      units1 text;
      abbrv2 text,
      name2 text,
      units2 text,
      params text
);
```

For each sensor, we need:

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 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).

In previous versions a generic name was allowed (“Temperature”, “Pressure”, “Humidity” etc.). These generic names are deprecated and support for generic names will be removed in some future version.

Name A arbitrary name of a function of the device.

Abbrv An 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 only required for MPX4115A based pressure sensors.

It is 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 that the device records in. The presence of the abbreviation field determines if that specific function is logged.

So, for example: I have a 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.

So my configuration for this device is:

```
device      = 12FC6B34000000A9
type        = Pressure
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):

e.g.: **SQL:**

```
INSERT INTO "w1sensors" VALUES('12FC6B34000000A9', 'Pressure', 'OPRS', 'Pressure',
'hPa', 'OTMP1', 'Temperature', '°C',NULL);
```

or, **.config/w1retap/sensors:**

```
12FC6B34000000A9:Pressure:OPRS:Pressure:hPa:OTMP1:Temperature:°C
```

For my complete station:

```
INSERT INTO w1sensors (device, "type", abbrv1, name1, units1, abbrv2, name2,
units2, params) VALUES ('286DA467000000AD', 'DS1820', 'GHT', 'Greenhouse
Temperature', '°C', NULL, NULL, NULL, NULL);
INSERT INTO w1sensors (device, "type", abbrv1, name1, units1, abbrv2, name2,
units2, params) VALUES ('10A942C10008009B', 'DS1820', 'OTMP0', 'Outside
Temperatue', '°C', NULL, NULL, NULL, NULL);
INSERT INTO w1sensors (device, "type", abbrv1, name1, units1, abbrv2, name2,
units2, params) VALUES ('1093AEC100080042', 'DS1820', 'XTMP2', 'Garage
Temperature', '°C', NULL, NULL, NULL, NULL);
INSERT INTO w1sensors (device, "type", abbrv1, name1, units1, abbrv2, name2,
units2, params) VALUES ('26378851000000AB', 'TAI8540', 'OHUM', 'Humidity', '%',
'OTMP2', 'Garage Temperature', '°C', NULL);
INSERT INTO w1sensors (device, "type", abbrv1, name1, units1, abbrv2, name2,
units2, params) VALUES ('12FC6B34000000A9', 'TAI8570', 'OPRS', 'Pressure', 'hPa',
'OTMP1', 'Temperature', '°C', NULL);
INSERT INTO w1sensors (device, "type", abbrv1, name1, units1, abbrv2, name2,
units2, params) VALUES ('1D9BB10500000089', 'TAI8575', 'RGC0', 'Counter0', ' tips',
'RGC1', 'Counter1', 'tips', NULL);
```

or (lines starting # are comments, old set of sensors, deprecated device names)

```
$ cat ~/.config/w1retap/sensors
# Device:Type:Abbrv1:Name1:Units1:[Abbrv2:Name2:Units2]
286DA467000000AD:Temperature:GHT:Greenhouse Temperature:°C:::
2692354D00000095:Humidity:OHUM:Humidity:%:OTMP0:Temperature:°C
12FC6B34000000A9:Pressure:OPRS:Pressure:hPa:OTMP1:Temperature:°C
1D9BB10500000089:RainGauge:RGC0:Counter0:tips:RGC1:Counter1:tips
```

Note that the Greenhouse temperature sensor only has one function, so the abbrv2, name2 and unit2 fields are not defined (or NULL). None of the devices require a params field.

Complex sensors (Coupler / Parameters)

If you have DS2409 Microlan couplers or a MPX4115A based pressure sensor, your configuration requires a little more work:

Microlan For a Microlan device, the abbrv1 field is set to 'MAIN' and the abbrv2 filed is set to 'AUX'. The name1 field is a comma separated list of the device ids on the main branch, the name2 field is a list of the devices on the auxiliary branch.

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 (as many as are necessary for any particular device; not limited to the MPX4115A / DS2438 combination).

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','Coupler','MAIN','264E1169000000B5',NULL,'AUX',
'01F8A3880E0000A2',NULL,NULL);
```

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

TAI8518 Weather Station

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 provides a formula for converting counts per time interval to wind speed <http://www.aagelectronica.com/aag/en-us/pg_10.html#Q16>.

Summary of device type naming

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

Type	Alternatives	Deprecated	Device
DS1820	DS18S20	Temperature	DS1820,DS18S20 temperature sensors.
TAI8540		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.
TAI8515	Windvane, Weathervane		AAG Weather station wind direction sensor (DS2450 based).
DS2490	Coupler		DS2409 Microlan Coupler.

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 a double precision floating point.

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.

e.g. For my sensors:

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

And for Mihail Peltekov's sensors:

```
$ w1find /dev/ttyS0
(1) 106B89C4000800B9    18S20:high precision digital thermometer
(2) 01E3D68A0E0000B9    2401:silicon serial number
(3) 1FCD2D020000007F    2409:microlan coupler
        (Main.1) 264E1169000000B5       2438:smart battery monitor
        ( Aux.1) 01F8A3880E0000A2       2401:silicon serial number
```

The two w1sensors tables described previously relate to these configurations. Note that I am using a DS2490 USB adaptor, while Mihail has a DS2480 serial adaptor.

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.

The file contains a set of key / value pairs, blank lines and unrecognised line (e.g. #...) ignored.

e.g.

```
#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 = 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 section 8).

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.

delay The delay between successive reading of the the 1-wire bus. All sensors are read in one hit.

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”.

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 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.

```
e.g.

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):

```
2005-07-29T18:11:28+0100 GHT 20.312500
2005-07-29T18:11:28+0100 OHUM 74.050064
2005-07-29T18:11:28+0100 OTMP0 17.687500
2005-07-29T18:11:28+0100 OPRS 1009.950562
2005-07-29T18:11:28+0100 OTMP1 18.510059
2005-07-29T18:11:28+0100 RGC0 3496.000000
2005-07-29T18:11:28+0100 RGC1 3460.000000
```

w1xml

This 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
```

The data output is in the format:

```
<?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>
```

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"
```

w1sqlite

This provides database system access for configuration and logging using an Sqlite v.3 <http://www.sqlite.org> RDBMS. If used as a 'init' parameter, it reads the w1sensors table for sensor information as described for RDBMS based sensor configuration.

e.g.

```
init = w1sqlite=/var/tmp/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/tmp/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 7;
1122735840|GHT|25.625
1122735840|OHUM|87.6851425170898
1122735840|OTMP0|18.34375
1122735840|OPRS|1009.77972412109
1122735840|OTMP1|19.1231441497803
1122735840|RGC0|3498
1122735840|RGC1|3462
```

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

w1pqsql

This provides database system access for configuration and logging using an PostgreSQL <http://www.postgresql.org> RDBMS.

If used as a 'init' parameter, it reads the w1sensors table for sensor information as described for RDBMS based sensor configuration.

e.g.

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

The name of the database is a mandatory parameter, followed by optional parameters in the format described for PostgreSQL client programs e.g. See http://www.postgresql.org/docs/8.0/interactive/libpq.html section 27.1 describing the 'conninfo' format.

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

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

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

w1mysql

This provides database system access for configuration and logging using a MySQL < http://dev.mysql.com/> RDBMS.

If used as a 'init' parameter, it reads the w1sensors table for sensor information as described for RDBMS based sensor configuration.

e.g.

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

The name of the database is a mandatory parameter, followed by optional parameters of:

```
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 provides database system access for configuration and logging using ODBC <http://www.unixodbc.org/> 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.

e.g.

```
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].

Running w1retap

W1RETAP is started from the command line (shell script, @reboot cron job 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.

It accepts the following command options:

```
$ w1retap [-T] [-d] [-i interface] [-N] [-v] [-t sec] [-w] [-r repfile]
where:
-T disables /tmp/.w1retap.dat logging (e.g. logtemp = 0)
-d  causes w1retap to daemonise (run in backgound) (e.g. daemonise = 1)
-i interface specified the interface device. Defaults to
-i DS2490-1, (e.g. interface = DS2490-1)
-N Do not open the device or read the sensors (for debugging).
-v verbose (display configuration information).
-t secs Specify delay loop so -t 120 reads the sensors every 120s [2 minutes] (e.g. Delay = 120)
-w Wait at after startup to the next “Delay” time before reading the sensors (vice immediately).
-r repfile Report to file “repfile” when a sensor reports out of range data (see “Rate Limiting” below.
```

Equivalent configuration options are show as (e.g. .....).

The author runs **W1RETAP** as:

```
$ w1retap -d -t 120 -w -r /tmp/w1.limits
```

$ w1retap -Nv will dump out the configuration,

e.g: with `~/.config/w1retap/rc`:

```
#Init file
init = w1pgsql=dbname=sensors user=w1retap
log = w1pgsql=dbname=sensors user=w1retap
rep = w1pgsql=dbname=sensors user=w1retap
altitude = 19
# End of file
```

result:

```
$ ./w1retap -Nv
w1retap v0.0.20-rc1 (c) 2005,2006 Jonathan Hudson
Sensors:
286DA467000000AD DS1820
      1: GHT Greenhouse Temperature °C, 2.5000 /min
10A942C10008009B DS1820
      1: OTMP0 Outside Temperatue °C, 2.5000 /min
1093AEC100080042 DS1820
      1: XTMP2 Garage Temperature °C
26378851000000AB TAI8540
      1: OHUM Humidity %, 7.0000 /min, min=0.00, max=100.04
      1: OTMP2 Garage Temperature °C, 2.5000 /min, min=-10.00, max=50.00
12FC6B34000000A9 TAI8570
      1: OPRS Pressure hPa, 100.0000 /min, min=800.00, max=1200.00
      1: OTMP1 Temperature °C, 2.5000 /min
```

```
1D9BB10500000089 TAI8575
        1: RGC0 Counter0  tips, 50.0000 /min
        1: RGC1 Counter1 tips, 50.0000 /min
Plugins:
 0: c [0x8079490] /usr/lib/w1retap/libw1pgsql.so => dbname=sensors user=w1retap
 1: l [0x8079490] /usr/lib/w1retap/libw1pgsql.so => dbname=sensors user=w1retap
 2: r [0x8079490] /usr/lib/w1retap/libw1pgsql.so => dbname=sensors user=w1retap
Normalising pressure for 19m
```

Note that the plugins are assumed to be located in `/usr/lib/w1retap/`, (or where prefix was set at build time, e.g. `–prefix=/usr/local ---> /usr/local/lib/w1retap`) unless the name starts with '/' or '.'; in which case the actual path is used. If you don't give a path, you can name the module without 'lib' and '.so'. The loading mechanism (GLib/gmodule) should work on any platform where dynamically loadable libraries are supported (most Unix, Microsoft Windows etc.), but is only tested on Linux.

e.g.

```
log = w1csv
log = ./libw1xml.so
log = /tmp/testme-harder/libw1b0rken.so
```

And for Mihail Peltekov's sensors:

```
$ w1retap -Nv
w1retap v0.0.20-rc1 (c) 2005,2006 Jonathan Hudson
Sensors:
106B89C4000800B9 DS18S20
        1: DS1820 Temp Temperature °C
264E1169000000B5 MPX4115A
        Microlan: 1FCD2D020000007F, main
        Parameters: 34.249672 762.374682
        1: Baro Press Pressure hPa
        2: Baro Temp Temperature °C
01F8A3880E0000A2 SHT11
        Microlan: 1FCD2D020000007F, aux
        1: SHT11 RH Humidity %
        2: SHT11 Temp Temperature °C
1FCD2D020000007F Coupler
        1: MAIN 264E1169000000B5
        2: AUX 01F8A3880E0000A2
Plugins:
 0: c [0x8076130] /home/w1user/lib/w1retap/libw1mysql.so => dbname=sensors
user=w1user password= SomethingSecretAndBulgarian
 1: l [0x8076130] /home/w1user/lib/w1retap/libw1mysql.so => dbname=sensors
user=w1user password=SomethingSecretAndBulgarian
Normalising pressure for 440m
```

Rate Limiting

Very occasionally one of the author's sensors will 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. The following SQL commands created the author's ratelimit table.

```
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);
```

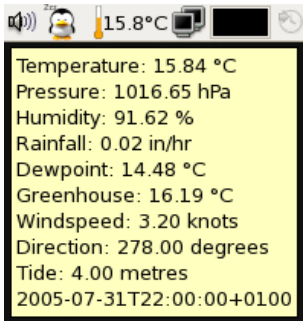
The values are such that they would not normally be seen, but are less than the obviously bizarre rogue value seen very rarely.

Viewing the data.

The `wplot.rb` (and other) scripts in the `contrib` directory illustrates techniques to access the data and:

- Build a web page e.g. <http://www.daria.co.uk/wx/>;
- Send data to Wunderground.com (e.g. See <http://www.wunderground.com/global/stations/03865.html >);
- Provide an RSS feed (e.g. <http://www.zen35309.zen.co.uk/wx/wx.rss2.xml>);
- Provide a static XML document of current conditions, (e.g. <http://www.zen35309.zen.co.uk/wx/wx_static.xml>). The latter format is read by the **W1RETAP** GNOME applet (`contrib/w1temp`) --- you can provide your own location as I'm sure you don't want to know what it's like here in Netley Marsh.

The `wplot.pl` and other scripts also require that the station table is populated. See `contrib/README` for details.



The `contrib/w1temp` directory contains a GNOME applet that can display a single temperature in the GNOME panel, and a set of data defined by the static XML file in a tooltip.

Credits

Thanks to:

Mihail Peltekov <http://zlatograd.com> for providing ssh access to zlatograd.com, which allowed me to develop the DS2480, DS2409, SHT11 and MPX4115A device support;

William R Sowerbutts <http://sowerbutts.com> provided a patch to allow field order independent PgSQL logging, the TAI8515 code and the 'one table per sensor' PgSQL logging code;

Joe Carter and Steven Searby provided bug reports;

Daria Hudson started this by requiring a temperature sensor in her greenhouse and has graciously allowed me pursue my interest in 1-wire weather stations since then.

Author / contact

W1RETAP is (c) Jonathan Hudson <jh+w1retap@daria.co.uk>. It is released (mainly) under the GNU Public licence.