SwitchHub 0.42

Friday, March 11, 2016

Table of Contents

1 Introduction

1.1 SwitchHub

SwitchHub is a command line application for Linux (exclusively for the time being) that allows flexible control of the Telldus Tellstick[1]. The Telldus Tellstick is a device that can be controlled from a computer to control power switches remotely, refer to Illustration 1. SwitchHub can be adopted to control similar devices that can be controlled from the command line interface.

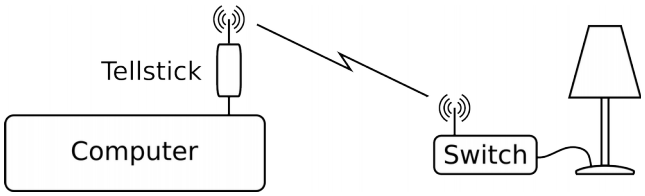

Illustration 1: Controlling an electrical device

The scheduling of events (i.e. scheduling switches to turn on or off, or dim) is made by creating Boolean expressions[2], one per switch. This is an example of a Boolean expression used as an event definition:

```
on = 06:15 <= t < 22:15 and weekday
```

When the Boolean expression becomes True (between 06:15 and 22:15 during a weekday), the switch is turned on. When the Boolean expression becomes False (outside the time interval or on a weekend), the switch is turned off. To write Boolean expressions, you will use:

- Variables, e.g. 't' and “weekday”
- Boolean operators, “and”, “or” and “not”
- Relational operators, e.g. '<' and “>=”.

1.2 SwitchHub characteristics

For you to know what to expect from SwitchHub, here follows a list of some of the important characteristics of the application:

- SwitchHub is lightweight and will not use much of your system's resources.
- SwitchHub can use plug-ins. The core functionality (based on Python standard libraries) is implemented in the SwitchHub software itself. You then have the possibility to add plug-ins to increase the number of variables that can be used in the event definitions. Some plug-ins are bundled with SwitchHub.
- SwitchHub can control both switches and dimmers.
- The scheduling of events is very flexible. This is because:
  - You have a high degree of freedom when you write the Boolean expressions.
  - The amount of meaningful variables.
  - You can add variables by writing your own plug-ins, or installing the plug-ins that

1 Tellstick is a product made by Telldus Technologies, www.telldus.se.
2 https://en.wikipedia.org/wiki/Boolean_expression

follow with SwitchHub.

- SwitchHub is made to run unattended in the background. Once configured and started, SwitchHub will be out of your way.
- SwitchHub does not have a GUI. You will edit a few text files to configure the program and the events.
- SwitchHub is written in Python 3. This is only of interest if you want to read the code or make changes to it.

1.3 The SwitchHub project

The program files and this document resides at GitHub at www.github.com/switchhub. You are encouraged to provide feedback in any form, bug reports, feature requests or code.

Besides maintaining SwitchHub for the intended usage with the characteristics as described in this document, the following goals are also considered important:

- Pythonic code (that is readable code, written the" Python way").
- Understandable and sufficient documentation.

1.4 This document

This document assumes that you have a general understanding of computers, especially computers that run a variant of the GNU/Linux operating system.

This document describes SwitchHub as it works in its current version. If you want to now what changes and new features to expect in the future, refer to https://github.com/thoelf/switchhub/issues.

1.5 Legal notes

SwitchHub is released as free/libre open source software under the GNU General Public License v3. For the license text, refer to the `LICENSE` file or to www.gnu.org/licenses/gpl.html.

You will use SwitchHub (the program) at your own risk. The program is not intended for mission critical tasks, where the condition of physical property or the life or health of humans or animals depends on how the program behaves.

2 Configuration files

2.1 Introduction

SwitchHub uses two configuration files which are installed in the directory /etc/switchhub:

- events, for the event definitions.
- switchhub, to control certain aspects of the program execution.

SwitchHub will read the configuration files when it starts. If you want SwitchHub to be aware of changes you make in a configuration file when SwitchHub is running, you must stop and start SwitchHub.

Depending on which plug-ins you use, there can be more configuration files to care for in the directory /etc/switchhub/plugins/.

2.2 The event definitions file, events

2.2.1 Structure

The event defintion file (/etc/switchhub/events) can have for example this contents:

```
[kitchen_lamp]
id = 1
on = 06:00 < t <= 12:30 and monday

[bedroom_lamp]
id = 2
dim_10 = 06:00 < t <= 07:00 and weekday
dim_90 = 15:00 < t <= 22:00 and ping:sven
```

The event defintion file contains:

- The device to control, e.g. kitchen_lamp in square brackets.
- An device id number.
- The event type: on, only_on, only_off or dim_xx.
- An event expression with variables, Boolean operators and relational operators.

The device name and the device id number should match the name and id in /etc/tellstick.conf.

When using the event type on, the device will be powered on when the expression becomes True and turned off when the expression becomes False.

When using the event type only_on, SwitchHub will power on the device when the expression becomes True, but SwitchHub will not power off the device when the expression becomes False. The opposite is true for the event type only_off.

The event type dim_xx (e.g. dim_50) can be used together with dimmers. You can also control a dimmer with the other event types. You can mix event types for a device, but you cannot use identical event types for a device.

Different dimmers, with different loads and settings will behave differently. That means that when you use dimmers, you must be aware of the different dimlevels to send to the device. You can experiment with different dimlevels with this command (requires that you have installed telldusd):

```
tdtool --dimlevel <level> --dim <device id number>
```

2.2.2 The built-in variables used by the event definitions

For descriptions of the built-in variables you can use when you write event definitions, refer to Table 1.

Variable	Description
`t`	The time right now
`weekday`	The variable is True between Monday and Friday
`monday, tuesday, wednesday, thursday, friday, saturday, sunday`	You can use a day, or a combination of days, as variables. Each variable is True during the corresponding day
`january, february, march, april, may, june, july, august, september, october, november, december`	You can use a month, or a combination of months, as variables. Each variable is True during the corresponding month
`ping:host`	A host on the internet or most likely on your LAN. Configure the host IP or URL in the file /etc/switchhub/switchhub. Any number of hosts can be defined. You can use this variable to let an expression depend on whether you or a family member are home or not (by letting SwitchHub ping your cell phone(s)). This only works if: • You have a LAN • The IP addresses of the hosts are set dependent on the MAC addresses, so that the IP addresses are consistent over time

Table 1: Built-in variables

2.2.3 Boolean operators

For the Boolean operators that you can use when you define events, refer to Table 2.

Boolean operator	Description
and	This operator is used when you need to express that two variables or expressions must be True for the expression to be True. The expression A and B is True only if both A and B is True. (A = True) and (B = True) => True (A = True) and (B = False) => False (A = False) and (B = True) => False (A = False) and (B = False) => False

Boolean operator	Description
or	This operator is used when you need to express that any variable or expression must be True for the expression to be True. The expression A or B is True only if both A and B is True or if either A or B is True. (A = True) or (B = True) => True (A = True) or (B = False) => True (A = False) or (B = True) => True (A = False) or (B = False) => False
not	This operator is used when you want to express that a variable or expression must be False for the expression to be True. not A is only True is A is False. not (A = True) => False not (A = False) => True

Table 2: Boolean operators

2.2.4 Relational operators

For the relational operators that can be used when defining events, refer to Table 3.

Relational operator	Description
==	“Equal to” `5 == 5` is True `5 == 6` is False
!=	“Not equal to” `5 != 6` is True `5 != 5` is False
<	“Less than” `5 < 6` is True `6 < 5` is False
>	“Greater than” `6 > 5` is True `5 > 6` is False
>=	“Greater than or equal to” `6 >= 5` is True `5 >= 5` is True `5 >= 6` is False
<=	“Less than or equal to”

Relational operator	Description
	`5 <= 6` is True `5 <= 5` is True `6 <= 5` is False

Table 3: Relational operators

2.2.5 Example 1 – on and time

```
[aquarium]
id = 1
on =  14:15 <= t < 22:15
```

The device named ”aquarium” with id number '1' will be on between 14:15 and 22:15 (because that is when the expression is True). The device name ”aquarium” is a unique name that is intended for you to know which device it is. The id number, '1', is a unique number that is used in the calls to the Tellstick (or a similar device). If you are using Tellstick, the id number must match the id number for the device in /etc/tellstick.conf.

2.2.6 Example 2 – only_on, time and ping:host

```
[kitchen_lamp]
id = 2
only_on = t == 06:15 and ping:t_s5
```

”kitchen_lamp” will be turned on at 06:15 if t_s5 is present on the LAN. Note the difference between ”on” in Example 1 and ”only_on” in Example 2. ”only_on” is used to turn a device on, but the device will not be turned off when the expression becomes False. ”only_on” is useful when you want to turn off the device manually.

2.2.7 Example 3 – only_off and time

```
[tv]
id = 3
only_off = t == 03:00
```

”tv” will be turned off at 03:00. The device can only be turned on manually.

2.2.8 Example 4 – multiple rows and multiple variables

```
[bedroom_lamp]
id = 5
on =  (06:00 <= t < 07:15 and weekday) or \
      (16:20 <= t < 22:30 and weekday) or \
      (15:00 <= t < 22:30 and not weekday)
```

This example shows how it is possible to build long expressions while keeping the readability. Each row contains an expression that, if True will make the whole expression True.

2.3 The program configuration file, switchhub

2.3.1 Introduction

Each of the following sub chapters corresponds to a section in the file /etc/switchhub/switchhub. The shown settings are the default settings.

2.3.2 The logging section

```
[logging]
log_level = WARNING
```

The log level defines what kind of log info you will get in the log file /var/log/switchhub.log. You can choose from the log levels that follow:

- DEBUG
- INFO
- WARNING
- ERROR
- CRITICAL.

The log level DEBUG will make all types of log messages to be written to the log file. If you choose for example the log level WARNING, only log messages of the type WARNING, ERROR and CRITICAL will be written to the log file.

The log file is rotated. Refer to /etc/switchhub_logrotate for information about the configuration of the log rotation.

You will find the log file in /var/log/switchhub.

The logging functionality is not fully implemented, so many fail states will not be logged.

2.3.3 The misc section

```
[misc]
event_config = /etc/switchhub/
```

The path to the event configuration file events. The file is read when SwitchHub starts, and only then.

```
repeats = 3
```

With this setting you can make SwitchHub repeat its calls to the devices you are controlling. You might want to do repeated calls if the connection between the Tellstick and the receiving switches is poor.

2.3.4 The ping_ip section

```
[ping_ip]
thomas = 192.168.1.110
marlene = 192.168.1.111
```

This is where you define the hosts that SwitchHub will ping, to know if they are on the LAN or not. You can also specify an URL here. The variable name for the ping to a host is on the form ping:host. ping:host is True when the devise is on the network, otherwise False.

2.3.5 The timer section

```
[timer]
ping_off_delay = 10
```

Time in minutes for SwitchHub to notice that the connection to a host (that is configured in the

[ping_ip] block) is lost. In other words, the time in minutes for ping:host to go from True to False when the connection to the host is lost. The off_delay is good to have if you for example want a lamp to go off some time after you leave your home. The off_delay also functions as a filter, so that if you loose the wifi-connection now and then, the devices that you control will not be so easily affected (no flickering).

```
turn_around = 60
```

The time in seconds for how often SwitchHub will check the value of the Boolean expressions and possibly change the states of the switches. 60 seconds is a good default value. If you use for example motion sensors, you might want to lower the turnaround time to a few seconds.

3 Description of the software components

3.1 Software components

3.1.1 Python 3

Python is a programming language and the software that will interpret and compile the code in the different SwitchHub modules.

3.1.2 telldusd

telldusd is the program that communicates with the Tellstick. SwitchHub calls telldusd when it is time to operate a switch (or dimmer).

3.1.3 SwitchHub

SwitchHub consists of 3 modules:

- switchhub.py is the main module. This is the program that you run when you want to start SwitchHub.
- get_plugin_data.py is the module that collects data from any plug-ins.
- operate_switch.py – This module tells the telldusd software to turn a switch on or off.

3.1.4 Plug-ins

These plug-ins are included in the software package from GitHub (but they are not enabled by default):

- The calendar plug-in (cal.py) gives you variables for holidays.
- The sunrise/sunset plug-in (suntime.py) gives you variables for sunrise and sunset.
- The read files plugin (readfiles.py) lets SwitchHub read variable values from files that you define.
- Wunderground plug-in (wunderground.py) gives you variables with weather values, for example temperature or weather type. (Coming soon...)

For more information about these plug-ins and plug-ins in general, refer to chapter 6.

4 Installing SwitchHub

4.1 Introduction

This instruction is tested on Ubuntu Server 14.04 32-bit and Ubuntu Desktop 13.10 64-bit. It will probably work without major tweaks on other Ubuntu versions or other Linux distributions (especially those in the Debian/Ubuntu "family"). The procedure describes how to do the installation from the terminal. Most or all steps can be done from the GUI if you prefer that.

4.2 Install Python 3

1. Most Linux distributions already have Python 3 installed. To verify that you have Python 3 installed, enter this command:

   ```
   python3 --version
   ```

   The expected output is something like: `Python 3.3.2+`

2. If you are missing Python 3, install it with this command:

   ```
   sudo apt-get install python3
   ```

4.3 Install unzip

1. Install unzip (which will be used to unzip the downloaded SwitchHub software):

   ```
   sudo apt-get install unzip
   ```

4.4 Install telldusd

1. Install telldusd, refer to http://developer.telldus.com/wiki/TellStickInstallationUbuntu.
2. Configure telldusd, refer to http://developer.telldus.com/wiki/TellStick_conf.
3. Initialize a switch. Refer to the manual for the switch.
4. Test that you can turn a device on and off with these commands:

   ```
   tdtool --on <device id>
   tdtool --off <device id>
   ```

   If it is a dimmer, try this command:

   ```
   tdtool -dimlevel <level> --dim <device id>
   ```

4.5 Install SwitchHub

1. Make sure that you are in your home directory:

   ```
   cd
   ```

2. Download SwitchHub:

   ```
   wget https://github.com/thoelf/switchhub/archive/master.zip
   ```

3. Unpack the downloaded file:

   ```
   unzip master.zip
   ```

4. Go to the unpacked directory:

   ```
   cd switchhub-master
   ```

5. Run the installation script:

   ```
   sudo ./install.sh
   ```

6. Go to your home directory:

   ```
   cd
   ```

7. If you do not want to keep the downloaded file, remove it:

   ```
   rm master.zip
   ```

8. If you do not want to keep the install directory, remove it:

   ```
   rm -rf switchhub-master
   ```

4.6 Improving the robustness of SwitchHub

In the unlikely event of a program crash, SwitchHub will not restart automatically. To make SwitchHub restart automatically, you might want to add this command in your crontab:

```
/usr/bin/pgrep switchhub.py >/dev/null || (echo "Cron: Switchhub started"
$(date) >> /var/log/switchhub.log && /opt/switchhub/switchhub.py)
```

5 Managing SwitchHub

5.1 Starting SwitchHub

1. Make sure that you are in your home directory:
   ```
   cd
   ```
2. Start SwitchHub:
   ```
   ./switchhub.sh start
   ```
3. Detach from the screen session:
   ```
   Ctrl+A D
   ```

5.2 Stopping SwitchHub

1. Make sure that you are in your home directory:
   ```
   cd
   ```
2. Stop SwitchHub:
   ```
   ./switchhub.sh stop
   ```

5.3 Checking the status of SwitchHub

1. Make sure that you are in your home directory:
   ```
   cd
   ```
2. Check if SwitchHub is running:
   ```
   ./switchhub.sh status
   ```

6 Plug-ins

6.1 Introduction

By adding a plug-in you can increase the number of variables known by SwitchHub. You can then use these extra variables in the event definitions. You can write your own plug-ins.

You will find the plug-ins that are included with SwitchHub in the directory /opt/switchhub/plugins. The included plug-ins are:

- cal.py
- suntime.py
- readfiles.py
- wunderground_weather.py (coming soon).

The plug-ins above are not activated by default. For you to be able to use the plug-ins, you must follow the install and or activation procedure for the plugin.

6.2 Activating a plug-in

To activate a plug-in you have to copy it to one of the directories that follow in /opt/switchhub/plugins:

- every_time
- 1_minute
- 15_minutes
- hour
- day.

The name of a directory reflects how often the plug-ins in the directory will be executed and its data be collected by SwitchHub. All plug-ins will be executed once when SwitchHub starts. The plug-ins in every_time/ will be run at each turnaround (refer to chapter 2.3.5). The plug-ins in 1_minute/ will then be executed every full minute. The plug-ins in 15_minutes/ will be executed every full quarter. The plug-ins in hour/ will be executed every full hour. The plug-ins in day/ will be executed every day at 00:00.

You must also set the file permissions for the plug-in so that the group switchhub have the rights to execute the file.

6.3 Writing your own plug-ins

The plug-ins communicates to SwitchHub through stdout. The message from the plug-in must be on this format:

```
<plug-in name>;<variable name>;<variable value>
```

Example (output on stdout):

```
temperature.sh;outdoor_temp;22
```

If the plug-in requires a configuration file, you are recommended to put the configuration file in the

directory /etc/switchhub/plugins. Give the configuration file the same name as the plug-in executable, but exclude the file extension.

To let the plug-in output more than one variable, output the strings on subsequent rows. Example:

```
outdoor_sensor.sh;outdoor_temp;22
outdoor_sensor.sh;outdoor_hum;47
```

The plug-in variables must not conflict with the names of already existing variables known to SwitchHub. SwitchHub will inform you about any name conflicts.

Make the plug-in executable for the group switchhub.

You should not make a plug-in run forever. A plug-in should just fetch its data, output the data on stdout and then exit.

7 The Read Files plug-in, readfiles.py

The Read Files plug-in reads a variable value from the first line of a file. You define the file and the corresponding variable name in the file /etc/switchhub/plugins/readfile. In the configuration file, you can also define a default value for the variable/file that will be used if the file does not exist.

This is an example of what the contents of the file can look like:

```
[files]
party = /run/shm/data/party
weather_type = /run/shm/data/weather_type

[default]
party = False
weather_type = Clear
```

8 The calendar plug-in, cal.py

8.1 Variables provided by the calendar plug-in

The calendar plug-in provides the variables according to Table 4.

Variable	Description
`holiday_today`	The variable value depends on: • The day in the week. The variable is True on Saturdays and Sundays. • If it's holiday or not according to the downloaded calendar. Because the calendar can contain non work-free days, the program will check the file calendar_holidays to see if you have acknowledged the day as a free day. • The dates in calendar_free_days in which you can define your own free days
`holiday_tomorrow`	The variable is True if it is holiday tomorrow. The check will be done in the same way as for the variable holiday, but the check is made using tomorrow's date
`holiday_yesterday`	The variable is True if it was holiday yesterday. The check will be done in the same way as for the variable holiday, but the check is made using yesterday's date
`workday`	The variable is True between Monday and Friday when there is no holiday (refer to the holiday_today variable)

Table 4: Variables provided by the calendar plug-in

8.2 Configuration files for the calendar plug-in

8.2.1 The holidays configuration file, calendar_holidays

The file /etc/switchhub/plugins/calendar_holidays is a list of holidays that you have acknowledged as free days. When SwitchHub have found out, by reading the automatically downloaded holiday calendar from www.webcal.fi, that it might be a holiday, it will check the file calendar_holidays to see if you have acknowledged the day to be a free day. This procedure is necessary because not all days in the calendar are work free days. This is what a few rows in the file can look like (in Sweden):

```
Nyårsdagen
Trettondagsafton
Långfredag
…
```

To know what days to have in calendar_holidays, refer to www.webcal.fi. The name of the days must be written exactly as at www.webcal.fi, but the names are not case sensitive.

The contents of calendar_holidays will effect the variables holiday_yesterday, holiday_today and holiday_tomorrow.

8.2.2 The free days configuration file, calendar_free_days

In the file /etc/opt/switchhub/plugins/calendar_free_days you can add the dates when you are planning to be free from you ordinary occupation. If you for example are planning a vacation or a day off in the middle of the week, you can put the corresponding dates in calendar_free_days. This is what the dates in the file can look like:

```
2014-10-22
2014-11-03:2014-11-07
```

...

The dates in the file free_days are written in the format YYYY-MM-DD. In the example, the first row contains a single date. The second row contains a range of dates.

The contents of the file calendar_free_days will affect the variables holiday_yesterday, holiday_today and holiday_tomorrow.

8.3 Settings in the configuration file calendar

In the file /etc/switchhub/plugins/calendar you must define the web address to the calendar for your country (the url setting in /etc/switchhub/plugins/calendar). Edit the existing setting if you do not live in Sweden. At www.webcal.fi, look for and define a calendar (Calendar → Other file formats) for your country and time zone in JSON-format. Click **Show download URL**, then copy/paste the url to etc/switchhub/plugins/calendar. Note that you will have to duplicate the '%'-character.

8.4 Installing the calendar plug-in

1. Go to the plug-ins directory:
   ```
   cd /opt/switchhub/plugins
   ```
2. Move the plug-in to the day/ plug-in directory:
   ```
   mv calendar.py day
   ```
3. Restart SwitchHub:
   ```
   cd ..
   ./switchhub.sh stop
   ./switchhub.sh start
   ```

9 The sunrise and sunset plugin, suntime.py

9.1 Variables provided by the suntime plug-in

This plugin provides the variables according to Table 5.

Variable	Description
`sundown`	The variable is True between 00:00 to sunrise and between sunset to 23:59, e.g.: “00:00 <= t < 07:26 or 18:01 <= t < 23:59”
`sundown_plus_10`, `sundown_plus_20`, `sundown_plus_30`, `sundown_plus_40`, `sundown_plus_50`, `sundown_plus_60`	The time for sundown with offset plus 10..60 minutes. With the sundown value above, sundown_plus_10 equals: “00:00 <= t < 07:36 or 18:11 <= t < 23:59”
`sundown_minus_10`, `sundown_minus_20`, `sundown_minus_30`, `sundown_minus_40`, `sundown_minus_50`, `sundown_minus_60`	The time for sundown with offset minus 10..60 minutes. With the sundown value above, sundown_minus_10 equals: “00:00 <= t < 07:16 or 17:51 <= t < 23:59”
`sunup`	The variable is True between sunrise and sunset, e.g: “07:26 <= t < 18:01”
`sunup_plus_10`, `sunup_plus_20`, `sunup_plus_30`, `sunup_plus_40`, `sunup_plus_50`, `sunup_plus_60`	The time for sunup with offset plus 10..60 minutes. With the sunup value above, sunup_plus_10 equals: “07:36 <= t < 18:11”
`sunup_minus_10`, `sunup_minus_20`, `sunup_minus_30`, `sunup_minus_40`, `sunup_minus_50`, `sunup_minus_60`	The time for sunup with offset minus 10..60 minutes. With the sunup value above, sunup_minus_10 equals: “07:16 <= t < 17:51”

Table 5: Variables provided by the suntime plug-in

9.2 Settings in the configuration file suntime

In the suntime configuration file, you must enter the applicable location code (the default code is for Sweden). To find your location code (i.e. WOEID = Where On Earth ID), refer to this url http://woeid.rosselliot.co.nz/

9.3 Installing the suntime plug-in

4. Go to the plug-ins directory:
   ```
   cd /opt/switchhub/plugins
   ```
5. Move the plug-in to the day/ plug-in directory:
   ```
   mv suntime.py day
   ```
6. Restart SwitchHub:
   ```
   cd ..
   ./switchhub.sh stop
   ./switchhub.sh start
   ```

10 The Wunderground plug-in, wunderground.py

This plug-in is planned, but not yet implemented.