ORS SPECIFICATION

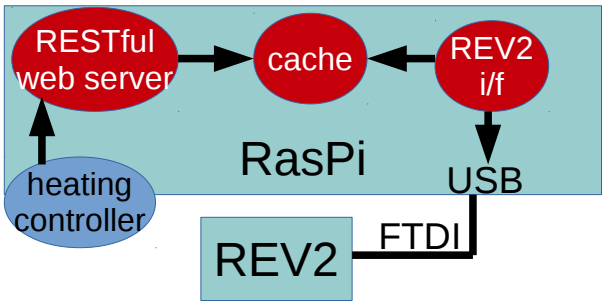
Modification history

Name	Ver	Date	Description
Mark Hill	1.0	29 June 2015	Initial version
Mark Hill	1.1	30 June 2015	Modifications per conversation with Damon (security, architecture, dialout group)
Mark Hill	1.2	30 June 2015	Added in timescales
Mark Hill	1.3	1 July 2015	Ready for publication

Summary

The work to be done is detailed in this document. Please refer to opentrv@opentrv.org.uk for clarification in case of any doubt. The objective is to produce a set of tested scripts and programs to run on a Raspberry Pi, to provide a REST interface to an OpenTRV REV2 controller, connected via an FTDI cable.

Possible architecture (other components may be added regarding data capture):



Details

1. The name, unless a better one arrives, is ORS (OpenTRV Rest Server).
2. Licence: all code and documentation will use Apache 2.0. Copyright is to be recorded and belongs to the author. No copyright assignment or contributor licence agreement required.
3. Written in a mainstream language. OpenTRV have a slight preference for Java, however, that is not required. Some of the mechanism and parsing to talk to the REV2 (REV2 i/f in diagram) has already been written in Java, and this can be reused.
4. With tests. Preferably using test driven development.
5. Really with tests. Essential. Not to be skipped over. 'Nuff said.
6. API can be in the code (a header file?), as long as it is well documented. For example, if ORS responded to http://ors:8080/api with a documented API in HTML format, that's good.
7. Built in webserver, that starts up in response to the run.sh startup script.
8. REST API to include version number, diagnostics, configuration (output only, not required to dynamically configure).
9. All logs to be written to $LOG_DIR which is by default $ORS_DIR/logs, max size of a log file, log rotation, max log retention all in the config file.
10. The SD card will be read only at runtime.

11. Dog fooding: ensure the deployment to your local Pi is slick so that you run it constantly, testing the scripts.
12. Store the project in the OpenTRV github repo. Commit often. Use meaningful commit messages.
13. Runtime is non privileged, implying a port greater than 1024.
14. Security TBD: TLS client server security or equivalent, plus other requirements may come to light depending on integrations with other systems.
15. Stateless/sessionless as far as possible, if not completely.
16. Non blocking.
17. Serial interface specification will be subject to change, so ensure flexibility in design to cater for this eventuality.
18. Scripts etc.:
    1. Install: shell script to unpack and install the system. Ensure that, if root privileges are required, that the script asks for them, but only if necessary. Will need to create users etc. Consider using ansible. Sensible return value and messages. Idempotent. Dialout group to gain access to serial port.
    2. Run-tests: shell script to run tests, either local (simulated REV2), or integration (connected REV2). Sensible return value and messages.
    3. Config file: self documenting, overridden by command line options
    4. Run: shell script, arguments include [--version] [--verbose] [--config <file>] [--port <portnumber>] [--tty <n>] Sensible return value and messages.
    5. Upgrade: shell script to unpack an upgrade file and run an upgrade. Note that this could be a blitz and reinstall, preserving (upgrading?) the config file, or it could be a very carefully crafted upgrade of certain components. Sensible return value and messages. Idempotent.
19. Config file to contain sensible defaults, so all command line parameters can be optional. Also include directory environment variables, such as $ORS_DIR, so that installation can be in any directory.
20. Obviously, stuff has been missed off this sheet, some things are subject to change, etc. Flexibility a must.
21. Timescales
    1. 08/07/2015 – API for heating controller/REST server interface
    2. 13/07/2015 – test harness (stub) for the REST server, with a on screen simulation of a REV2 board that can be run by pressing buttons on a web page (served up by the REST server, on a different URL)
    3. 17/07/2015 – delivery of first release
    4. 17/07/2015 – start integration testing with real hardware (REV2 i/f)
    5. 22-27/07/2015 – 100 unit test? (one Pi radio, 30m radius) (hopefully this be simulated)

6. 24/07/2015 – delivery of UAT release
7. 27/07/2015 – first installation
8. 10/08/2015 – ongoing installation: 10 units/day for the rest of the month