**TO-DOs:**

* From: <https://github.com/openaps/docs/blob/master/docs/getting-started/rpi.md>
  + change user login password or disable, only use SSH keys
  + figure out accessing Git on the device (necessary if any changes are being made, probably also to store me-specific files
* Read this: <https://github.com/openaps/openaps/blob/master/README.md>
* ~~From:~~ [~~https://github.com/openaps/docs/blob/master/docs/Log-clean-analyze-with-openaps-tools/using.md~~](https://github.com/openaps/docs/blob/master/docs/Log-clean-analyze-with-openaps-tools/using.md)
  + ~~configure OPENAPS on the Pi~~
* From: <https://github.com/openaps/docs/blob/master/docs/Build-manual-system/Using-oref0-tools.md>
  + re-invoke and review the reports with insulin pump settings, after I get old pump setup
  + starting doing steps after here: $ openaps report add monitor/glucose.json JSON cgm iter\_glucose 5
* draw out flow chart of reports, checks, and enacts for me to follow

**NOTES:**

* OPENAPS raspi:
* 192.168.1.132
* pi, raspberry

$ openaps use pump -h





$ openaps use cgm –h:



git push to Bitbucket:

Top of Form

Bottom of Form

Repository setup

Your repository is empty — let's put some bits in your bucket.

[I have an existing project](https://bitbucket.org/mikestebbins/openaps00#command-line-existing)

Already have a Git repository on your computer? Let's push it up to Bitbucket.

cd /path/to/my/repogit remote add origin https://mikestebbins@bitbucket.org/mikestebbins/openaps00.gitgit push -u origin --all # pushes up the repo and its refs for the first timegit push -u origin --tags # pushes up any tags

In order to eliminate needing to enter the Bitbucket password every time I “git push”, use the following instead of above, “… add origin https://mikestebbins: [BeRtensg@bitbucket.org](mailto:BeRtensg@bitbucket.org)...”

Crontab

Examples: <https://www.raspberrypi.org/documentation/linux/usage/cron.md>

<https://www.raspberrypi.org/forums/viewtopic.php?f=91&t=39344>

Access via:

$ crontab –e

Enter in Nano:

\*/5 \* \* \* \* (cd ~/openaps00/ && git push)



Nightscout integration

Followed these instructions: <https://github.com/openaps/docs/blob/dev/docs/Automate-system/vizualization.md>

Set-up clock zoned from line in script here: <https://github.com/openaps/oref0/blob/master/bin/ns-uploader-setup.sh#L51>

**More CRON to be set-up:**

live4sw 02:10

I think I may need to add something to retry-loop to deal with this scenario. My retry-loop is currently the default, which is retry-loop = ! bash -c "until( ! mm-stick warmup || openaps loop); do sleep 5; done". Has anyone here encountered a similar issue?

Basically, if I reboot, or reset the USB, it gets unstuck, but otherwise it just throws off that error every 5 minutes and doesn't even get to the preflight, even though everything is connected and i'm a few feet away from the stick

scottleibrand 03:48

openaps alias add preflight '! bash -c "rm -f monitor/clock.json && echo -n \"PREFLIGHT \" && openaps report invoke monitor/clock.json 2>/dev/null >/dev/null && grep -q T monitor/clock.json && echo OK || ( ( mm-stick warmup 2>&1 || sudo oref0-reset-usb ); echo FAIL; sleep 120; exit 1 )"'

openaps alias add retry-loop '! bash -c "openaps wait-loop || until( ! mm-stick warmup 2>&1 | egrep -v \"^ \" || ! openaps preflight || openaps loop); do sleep 10; done"'

live4sw 04:47

What do you use for your wait-loop alias? I didn't see that discussed in the docs.

scottleibrand 08:48

openaps alias add wait-loop '! bash -c "openaps preflight && openaps gather && openaps enact && openaps report invoke monitor/temp\_basal.json 2>/dev/null >/dev/null && openaps upload && openaps get-settings 2>/dev/null >/dev/null && openaps wait-for-bg && openaps enact && openaps upload-ns-status >/dev/null"'

live4sw 09:28

Thanks Scott, unfortunately this brings up wait-for-bg and several aliases not discussed in the docs. I see Jason uses these too. What is the purpose of the "wait" aliases? I think I may need to spend some time this weekend really trying to understand how these aliases do things differently.

jasoncalabrese 09:31

the wait-for-bg causes the loop to wait until BG changes, and just keeps polling

you then need a cron to killall openaps processes older than 10 minutes

\* \* \* \* \* killall -g --older-than 10m openaps

live4sw 09:40

Oh I see, so is the logic here that by using the wait, you can make sure that your loop runs immediately when there is a new bg? I assume your cron still runs retry-loop every 5 minutes, you're just killing old processes to make sure they don't snowball?

jasoncalabrese 09:49

the cron runs every minute, but checks if it's already running first

I use \* \* \* \* \* cd /home/edison/indy && ( ps aux | grep -v grep | grep -q 'openaps retry-loop' && echo OpenAPS already running || openaps retry-loop ) 2>&1 | logger -t openaps-loop

**Must go back and fix report monitor/upload-status.json, as I removed the enacted items just to test communication with Nightscout for CGM upload.**

**Python file in Linux**

Create new file from terminal:

* touch /path/to/file for an empty file

Just put this in the first line of your script :

~~#!/usr/bin/env python~~

Actually, change it to this:

#!/usr/bin/python

Then make the file executable with

chmod +x myfile.py

Execute with

./myfile.py

Use subprocess to run it in the shell:

<https://docs.python.org/2/library/subprocess.html>

<http://stackoverflow.com/questions/89228/calling-an-external-command-in-python>

**Round two, 2016-03-26. Install Dev into a new folder and start fresh with the Dev docs.**

**Names of things:**

* pump name = pump
* Dexcom cgm = cgm

Changed password to “GlareBucket2” using:

sudo raspi-config

Ran:

sudo bash -c 'echo "options 8192cu rtw\_power\_mgnt=0 rtw\_enusbss=0" >>/etc/modprobe.d/8192cu.conf'

Installed watchdog via:

Enable the built-in hardware watchdog chip on the Raspberry Pi:

sudo modprobe bcm2708\_wdog

sudo bash -c 'echo "bcm2708\_wdog" >> /etc/modules'

Install the watchdog package, which controls the conditions under which the hardware watchdog restarts the Pi:

sudo apt-get install watchdog

Next, add watchdog to startup applications:

sudo update-rc.d watchdog defaults

Edit the config file by opening up nano text editor

sudo nano /etc/watchdog.conf

Uncomment the following: (remove the # from the following lines, scroll down as needed to find them):

max-load-1

watchdog-device

Finally, start watchdog by entering:

sudo service watchdog start

Update the RPi2.

sudo apt-get update && sudo apt-get -y upgrade

Install dev via package manager:

curl -s https://raw.githubusercontent.com/openaps/docs/dev/scripts/quick-src.sh | bash -

Install openaps via:

sudo easy\_install -ZU setuptools sudo easy\_install -ZU openaps

Run

sudo openaps-install-udev-rules

Run

sudo activate-global-python-argcomplete

log session via:

script logfile20160326

Initialize new Openaps environment

$ cd

$ openaps init openapsdev

$ cd openapsdev

Add devices:

$ openaps device add pump medtronic 012842

$ openaps device add cgm dexcom

$ openaps device show

$ cat openaps.ini

$ cat pump.ini

$ openaps use <my\_pump\_name> model

$ openaps use <my\_pump\_name> -h

$ openaps use <my\_dexcom\_name> iter\_glucose 1

Pull nightscout values:

[device "curl"] <br>

fields = <br>

cmd = bash <br>

vendor = openaps.vendors.process <br>

args = -c "curl -s https:// mikestebbinscgmsite.azurewebsites.net/api/v1/entries.json | json -e 'this.glucose = this.sgv'" <br>

[report "monitor/glucose.json"] <br>

device = curl <br>

use = shell <br>

reporter = text <br>

reports

$ openaps report add last\_five\_pump\_hours.json JSON pump iter\_pump\_hours 5

$ openaps report invoke last\_five\_pump\_hours.json

$ openaps report add last\_five\_cgm\_hours.json JSON cgm iter\_glucose\_hours 5

aliases:

$ openaps report invoke last\_five\_pump\_hours.json

$ openaps report invoke last\_five\_cgm\_hours.json

$ openaps alias add last\_five\_hours "report invoke last\_five\_pump\_hours.json last\_five\_cgm\_hours.json"

$ openaps last\_five\_hours

Might need more reports here

**Phase 2: build a manual system**

$ openaps device add oref0 process oref0

$ openaps device add get-profile process --require "settings bg\_targets insulin\_sensitivities basal\_profile max\_iob" oref0 get-profile

$ openaps device add calculate-iob process --require "pumphistory profile clock" oref0 calculate-iob

$ openaps device add determine-basal process --require "iob temp\_basal glucose profile" oref0 determine-basal

$ mkdir -p settings monitor enact

$ openaps report add settings/settings.json JSON pump read\_settings

$ openaps report add settings/bg\_targets.json JSON pump read\_bg\_targets

$ openaps report add settings/insulin\_sensitivities.json JSON pump read\_insulin\_sensitivies

$ openaps report add settings/basal\_profile.json JSON pump read\_basal\_profile\_std

Create max\_iob file in main directory:

From main openapsdev directory:

touch max\_iob.json

nano max\_iob.json

input:

{"max\_iob":2}

$ openaps report add settings/profile.json text get-profile shell settings/settings.json settings/bg\_targets.json settings/insulin\_sensitivities.json settings/basal\_profile.json max\_iob.json

$ openaps alias add gather-profile "report invoke settings/settings.json settings/bg\_targets.json settings/insulin\_sensitivities.json settings/basal\_profile.json settings/profile.json"

$ openaps report add monitor/pumphistory.json JSON pump iter\_pump\_hours 5

$ openaps report add monitor/clock.json JSON pump read\_clock

$ openaps report add monitor/iob.json JSON calculate-iob shell monitor/pumphistory.json settings/profile.json monitor/clock.json

$ openaps report add monitor/temp\_basal.json JSON pump read\_temp\_basal

$ openaps report add monitor/glucose.json JSON cgm iter\_glucose 5

$ openaps report add enact/suggested.json text determine-basal shell monitor/iob.json monitor/temp\_basal.json monitor/glucose.json settings/profile.json

$ openaps alias add monitor-pump "report invoke monitor/clock.json monitor/temp\_basal.json monitor/pumphistory.json monitor/iob.json"

$ openaps alias add monitor-cgm "report invoke monitor/glucose.json"

$ openaps report show

$ openaps alias show

test the full sequence of aliases and the that which depend on them:

$ rm -f settings/\* monitor/\* enact/\*

$ openaps gather-profile

$ openaps monitor-pump

$ openaps monitor-cgm

$ openaps report invoke monitor/iob.json

$ openaps report invoke enact/suggested.json

**3.3.9**

$ openaps alias add preflight '! bash -c "rm -f monitor/clock.json && openaps report invoke monitor/clock.json 2>/dev/null && grep -q T monitor/clock.json && echo PREFLIGHT OK || (mm-stick warmup || (sudo oref0-reset-usb && echo PREFLIGHT SLEEP && sleep 120); echo PREFLIGHT FAIL; exit 1)"'

Added new Bitbucket repo for dev, pushed all to it using my password trick (earlier in this document)

Create loop

Perhaps re-check this: ktomy Jan 23 20:26: Also I have a very different "loop" command and maybe it would be good to add it to the documentation somewhere (without enact):

Loop:

openaps alias add loop '! bash -c "openaps monitor-cgm 2>/dev/null && ( openapspreflight && openaps gather && openaps enact) || echo No CGM data."'

Retry loop:

openaps alias add retry-loop '! bash -c "openaps preflight && until( ! mm-stick warmup || openaps loop); do sleep 5; done"'

3.3.16. Configuring Automatic Sensitivity Mode

Add nightscout upload functionality, Follow 3.4.4.1.2. Environment Variables for OpenAPS Access to Nightscout:

echo -n " thisismynewapisecret" | shasum

nano ~/.profile

NIGHTSCOUT\_HOST=https://mikestebbinscgmtest/azurewebsites.net/api/v1/entries.json; export $

API\_SECRET=ad28f5c7f3caa6679b83e8e7a2ed32c7aea3d962; export API\_SECRET

Save and exit, the run:

source /etc/profile

add a new ns-status device:

[device “ns-status”]   
fields = clock iob suggested enacted battery reservoir status  
cmd = ns-status  
vendor = openaps.vendors.process  
args =

$ openaps report add monitor/upload-status.json JSON ns-status shell monitor/clock-zoned.json monitor/iob.json enact/suggested.json enact/enacted.json monitor/battery.json monitor/reservoir.json monitor/status.json

$ openaps vendor add openapscontrib.timezones

$ openaps device add tz timezones

$ git add tz.ini

$ openaps report add monitor/clock-zoned.json JSON tz clock monitor/clock.json

When I run:

$ openaps report invoke monitor/upload-status.json

I get:

ns-status://JSON/shell/monitor/upload-status.json

Could not require: /home/pi/openapsdev/enact/enacted.json { [Error: Cannot find module '/home/pi/openapsdev/enact/enacted.json'] code: 'MODULE\_NOT\_FOUND' }

Could not require: /home/pi/openapsdev/monitor/battery.json { [Error: Cannot find module '/home/pi/openapsdev/monitor/battery.json'] code: 'MODULE\_NOT\_FOUND' }

Could not require: /home/pi/openapsdev/monitor/reservoir.json { [Error: Cannot find module '/home/pi/openapsdev/monitor/reservoir.json'] code: 'MODULE\_NOT\_FOUND' }

Could not require: /home/pi/openapsdev/monitor/status.json { [Error: Cannot find module '/home/pi/openapsdev/monitor/status.json'] code: 'MODULE\_NOT\_FOUND' }

Could not require: /home/pi/openapsdev/[ { [Error: Cannot find module '/home/pi/openapsdev/['] code: 'MODULE\_NOT\_FOUND' }

reporting monitor/upload-status.json

$ openaps report add monitor/reservoir.json JSON pump reservoir

$ openaps report invoke monitor/reservoir.json

$ openaps report add monitor/battery.json JSON pump read\_battery\_status

$ openaps report invoke monitor/battery.json

$ openaps report add monitor/status.json JSON pump status

$ openaps report invoke monitor/status.json

Now, when I run:

$ openaps report invoke monitor/upload-status.json

I get:

ns-status://JSON/shell/monitor/upload-status.json

Could not require: /home/pi/openapsdev/enact/enacted.json { [Error: Cannot find module '/home/pi/openapsdev/enact/enacted.json'] code: 'MODULE\_NOT\_FOUND' }

Could not require: /home/pi/openapsdev/[ { [Error: Cannot find module '/home/pi/openapsdev/['] code: 'MODULE\_NOT\_FOUND' }

reporting monitor/upload-status.json

searching for proper enact code:

from AAybob Jan 31 20:03:

enact ! bash -c "rm -f enact/suggested.json; openaps report invoke enact/suggested.json && cat enact/suggested.json && grep -q duration enact/suggested.json || echo No action required"

[report "oref0-predict/enacted.json"]

device = pump

input = oref0-predict/oref0.json

use = set\_temp\_basal

reporter = JSON

from docs:

# 3.3.9. Enacting the suggested action

Based on suggested.json, which is the output of the determine-basal oref0 process, the next step is to enact the suggested action, i.e. to send a new temp rate to the pump, to cancel the current temp rate, or do nothing. The approach one may follow is to setup an enacted.jsonreport, and a corresponding enact alias. Thinking about how to setup the enact report and alias, you may consider the following questions:

* Which pump command could be used to enact a new basal temp, if necessary, and what inputs should that command take? Where should these inputs come from?
* How could a decision be made whether a new basal temp should be sent to the pump or not? What should enact do in the cases when no new temp basal is suggested?

Once you setup your enact alias, you should plan to experiment by running the required sequence of reports and by executing the enact alias using $ openaps enact. Plan to test and correct your setup until you are ceratin that enact works correctly in different situations, including recommendations to update the temp basal, cancel the temp basal, or do nothing.

Up next:

$ ns-upload $NIGHTSCOUT\_HOST $API\_SECRET devicestatus.json monitor/upload-status.json