**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:**

* Pi 2 Model B
* UN/PW: pi/GlareBucket2
* Home: 192.168.1.132
* Work: 192.168.**XXX.XXX**

$ 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/repo

git remote add origin <https://mikestebbins@bitbucket.org/mikestebbins/openaps00.git>

git push -u origin --all # pushes up the repo and its refs for the first time

git push -u origin --tags # pushes up any tags

cd openapsdev/

git remote add origin <https://mikestebbins:BeRtensg@bitbucket.org/mikestebbins/openapsdev.git>

git push –u origin –all

git push –u origin --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

**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

**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 monitor-cgm-ns

$ openaps report invoke monitor/iob.json

$ openaps report invoke enact/suggested.json

rm -f settings/\* monitor/\* enact/\* && openaps gather-profile && openaps monitor-pump && openaps monitor-cgm-ns && 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

$ 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

Added create\_enacted.py and delete\_files.py to pi@raspberry:

Run:

$ python3 create\_enacted.py

Status:

The python code runs when run from the openaps directory. When run from the loop code, I get the following error:

pi@raspberrypi:~/openapsdev $ openaps loop

curl://text/shell/monitor/glucose-NS.json

reporting monitor/glucose-NS.json

pump://JSON/read\_clock/monitor/clock.json

reporting monitor/clock.json

PREFLIGHT OK

pump://JSON/read\_settings/settings/settings.json

reporting settings/settings.json

pump://JSON/read\_bg\_targets/settings/bg\_targets.json

reporting settings/bg\_targets.json

pump://JSON/read\_insulin\_sensitivities/settings/insulin\_sensitivities.json

reporting settings/insulin\_sensitivities.json

pump://JSON/read\_basal\_profile\_std/settings/basal\_profile.json

reporting settings/basal\_profile.json

get-profile://text/shell/settings/profile.json

Could not parse carbratio\_data. Optional feature Meal Assist disabled.

reporting settings/profile.json

Traceback (most recent call last):

File "/usr/local/bin/openaps", line 6, in <module>

exec(compile(open(\_\_file\_\_).read(), \_\_file\_\_, 'exec'))

File "/home/pi/src/openaps/bin/openaps", line 168, in <module>

app( )

File "/home/pi/src/openaps/openaps/cli/\_\_init\_\_.py", line 44, in \_\_call\_\_

self.run(self.args)

File "/home/pi/src/openaps/bin/openaps", line 154, in run

builtins.dispatch(args, self)

File "/home/pi/src/openaps/openaps/builtins.py", line 48, in dispatch

get\_alias(args.command, app)(args)

File "/home/pi/src/openaps/openaps/builtins.py", line 34, in \_\_call\_\_

exit(call(cmd + args.args))

File "/usr/lib/python2.7/subprocess.py", line 522, in call

return Popen(\*popenargs, \*\*kwargs).wait()

File "/usr/lib/python2.7/subprocess.py", line 710, in \_\_init\_\_

errread, errwrite)

File "/usr/lib/python2.7/subprocess.py", line 1335, in \_execute\_child

raise child\_exception

OSError: [Errno 2] No such file or directory

No CGM data.

3/31: Seemed to finally get openaps to call python script using python3.

Alias’ are now as follows:

enact = ! bash -c "./create\_enacted.py &> /dev/null || echo "error creating enacted""

loop = ! bash -c "openaps monitor-cgm-NS 2>/dev/null && ( openaps preflight && openaps gather-profile && ./create\_enacted.py) || echo No CGM data."

Ns-upload: added:

$ openaps alias add latest-ns-treatment-time '! bash -c "nightscout latest-openaps-treatment $NIGHTSCOUT\_HOST | json created\_at"' || die "Can't add latest-ns-treatment-time"

$ openaps alias add format-latest-nightscout-treatments '! bash -c "nightscout cull-latest-openaps-treatments monitor/pumphistory-zoned.json settings/model.json $(openaps latest-ns-treatment-time) > upload/latest-treatments.json"' || die "Can't add format-latest-nightscout-treatments"

$ openaps alias add upload-recent-treatments '! bash -c "openaps format-latest-nightscout-treatments && test $(json -f upload/latest-treatments.json -a created\_at eventType | wc -l ) -gt 0 && (ns-upload $NIGHTSCOUT\_HOST $API\_SECRET treatments.json upload/latest-treatments.json ) || echo \"No recent treatments to upload\""' || die "Can't add upload-recent-treatments"

Running: openaps report invoke monitor/upload-status.json returns:

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

Could not parse input data: [TypeError: Cannot read property 'duration' of undefined]

No JSON object could be decoded

Traceback (most recent call last):

File "/usr/local/bin/openaps-report", line 6, in <module>

exec(compile(open(\_\_file\_\_).read(), \_\_file\_\_, 'exec'))

File "/home/pi/src/openaps/bin/openaps-report", line 82, in <module>

app( )

File "/home/pi/src/openaps/openaps/cli/\_\_init\_\_.py", line 44, in \_\_call\_\_

self.run(self.args)

File "/home/pi/src/openaps/bin/openaps-report", line 75, in run

output = app(args, self)

File "/home/pi/src/openaps/openaps/cli/subcommand.py", line 52, in \_\_call\_\_

return self.method.main(args, app)

File "/home/pi/src/openaps/openaps/reports/invoke.py", line 48, in main

reporters.Reporter(report, device, task)(output)

File "/home/pi/src/openaps/openaps/reports/reporters/\_\_init\_\_.py", line 27, in \_\_call\_\_

self.blob = self.serialize(data)

File "/home/pi/src/openaps/openaps/reports/reporters/\_\_init\_\_.py", line 25, in serialize

return self.method.serialize(render(data), self)

File "/home/pi/src/openaps/openaps/vendors/process.py", line 56, in prerender\_json

return json.loads(data)

File "/usr/lib/python2.7/json/\_\_init\_\_.py", line 338, in loads

return \_default\_decoder.decode(s)

File "/usr/lib/python2.7/json/decoder.py", line 366, in decode

obj, end = self.raw\_decode(s, idx=\_w(s, 0).end())

File "/usr/lib/python2.7/json/decoder.py", line 384, in raw\_decode

raise ValueError("No JSON object could be decoded")

ValueError: No JSON object could be decoded

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

Deleted out the upload-status report definition from openaps.ini, then recreated the report with:

$ 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

Went into openaps.ini, and in that just-created report, I deleted the “[]” from the line “remainder = []”.

Now, running openaps report invoke monitor/upload-status.json seems to work.

Conversation history from gitter that got me fixed-up:

**mikestebbins**

[13:20](https://gitter.im/nightscout/intend-to-bolus?at=57017b038d2a72471b7ca33c)

@jasoncalabrese I tried first deleting the [], but that didn't change anything.  
I then deleted the upload-status report from my openaps.ini, then recreated the report using the code straight from the docs 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  
Invoking that report didn't work.  
I then went into openaps.ini, deleted the "[]" from the remainder line in the upload-status.json report, and after invoking that report, it worked.  
I must have messed up/corrupted my report definition somehow, but additionally, the brackets on the remainder line seem to have also been a problem. Thank you very much for the help!

**bewest**

[13:34](https://gitter.im/nightscout/intend-to-bolus?at=57017e4ebbffcc665fabc159)

the dev stuff for openaps has been updated recently, which fixes that issue

there's some new tools in oref0 topical/experimental branches to help create "recipes" and templates for dealing with lots of reports<https://github.com/openaps/oref0/pull/94#issuecomment-204616833>

**mikestebbins**

[13:46](https://gitter.im/nightscout/intend-to-bolus?at=57018135d9b73e635f68d26e)

@bewest I installed dev on the raspi using:

sudo easy\_install -ZU setuptools

sudo easy\_install -ZU openaps

sudo openaps-install-udev-rules

Do I just re-run sudo easy\_install -ZU openaps to update to the latest commit of dev with the changes you mentioned?

**bewest**

[14:15](https://gitter.im/nightscout/intend-to-bolus?at=570187f4d9b73e635f68d37c)

ah, easy\_install does not typicaly install dev

pip install git+git://github.com/openaps/openaps.git'#dev' might install dev

the usual workflow for using dev is to git clone the repo and install it from source using sudo python setup.py develop

**mikestebbins**

[14:19](https://gitter.im/nightscout/intend-to-bolus?at=570188f976b6f9de194dbb3d)

Dang. Thanks for the explanation @bewest. Can I update my existing openaps on the raspi from Master to Dev, or do I need to start from scratch using the workflow you describe?

I'm a bit confused. If I run sudo easy\_install -ZU openaps, it returns

Searching for openaps

Reading https://pypi.python.org/simple/openaps/

Best match: openaps 0.0.11.dev0

openaps 0.0.11.dev0 is already the active version in easy-install.pth

Using /home/pi/src/openaps

Processing dependencies for openaps

Finished processing dependencies for openaps

It says that openaps 0.0.11.dev0 is already the active version...but you're saying that I couldn't have dev if I installed with easy\_install?

**jasoncalabrese**

[14:27](https://gitter.im/nightscout/intend-to-bolus?at=57018ab3d478c81e2cbcaf4d)

the clone and setup above should replace the easy\_install version

**scottleibrand**

[14:29](https://gitter.im/nightscout/intend-to-bolus?at=57018b4876b6f9de194dbbac)

0.0.11 is not the latest dev. 0.1.0 is newer.

**bewest**

[14:30](https://gitter.im/nightscout/intend-to-bolus?at=57018b6abbffcc665fabc387)

that looks like you a have a checkout already

just need to fins it, issue git pull to bring it up to date

then sudo python setup.py develop

try cd ~/src/openaps

**mikestebbins**

[14:51](https://gitter.im/nightscout/intend-to-bolus?at=5701907a11ea211749c45fbc)

git pull returns Already up-to-date.. I'll try the clone and setup to replace my easy\_install version.

**bewest**

[14:52](https://gitter.im/nightscout/intend-to-bolus?at=570190878d2a72471b7ca6c5)

git status

your almost done

**mikestebbins**

[14:52](https://gitter.im/nightscout/intend-to-bolus?at=570190b211ea211749c45fc8)

git status returns

On branch master

Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: enact/enacted.json

modified: openaps.ini

modified: pump.ini

no changes added to commit (use "git add" and/or "git commit -a")

**bewest**

[14:53](https://gitter.im/nightscout/intend-to-bolus?at=570190bfd478c81e2cbcb043)

hmmm

git remote -v?

pwd

**mikestebbins**

[14:54](https://gitter.im/nightscout/intend-to-bolus?at=5701910411ea211749c45fd6)

origin https://mikestebbins:BeRtensg@bitbucket.org/mikestebbins/openapsdev.git (fetch)

origin https://mikestebbins:BeRtensg@bitbucket.org/mikestebbins/openapsdev.git (push)

and

/home/pi/openapsdev

**bewest**

[14:54](https://gitter.im/nightscout/intend-to-bolus?at=5701910cd478c81e2cbcb056)

ok:

cd ~/src/openaps

**mikestebbins**

[14:54](https://gitter.im/nightscout/intend-to-bolus?at=5701912311ea211749c45fdb)

ok.

**bewest**

[14:55](https://gitter.im/nightscout/intend-to-bolus?at=57019138e4a8384a1bbe7714)

git status

**mikestebbins**

[14:55](https://gitter.im/nightscout/intend-to-bolus?at=5701914b76b6f9de194dbc9d)

On branch dev

Your branch is up-to-date with 'origin/dev'.

nothing to commit, working directory clean

**bewest**

[14:55](https://gitter.im/nightscout/intend-to-bolus?at=570191528d2a72471b7ca6ed)

geat: git pull origin

**mikestebbins**

[14:56](https://gitter.im/nightscout/intend-to-bolus?at=5701919cd9b73e635f68d51f)

Nice, that shows 22 files changed!

**bewest**

[14:56](https://gitter.im/nightscout/intend-to-bolus?at=5701919e11ea211749c45fec)

sudo python setup.py develop

sudo easy\_install -ZU dexcom\_reader

**mikestebbins**

[14:59](https://gitter.im/nightscout/intend-to-bolus?at=57019225d39de41b496031c5)

after running setup.py develop I see Finished processing dependencies for openaps==0.1.0.dev0. And after running dexcom\_reader, I seeFinished processing dependencies for dexcom\_reader.

**bewest**

[14:59](https://gitter.im/nightscout/intend-to-bolus?at=5701922bd9b73e635f68d537)

cd ~/src/decoding-carelink; git pull origin; sudo python setup.py develop

very good

oops fixed typo

**mikestebbins**

[15:00](https://gitter.im/nightscout/intend-to-bolus?at=57019266d39de41b496031d3)

returns Finished processing dependencies for decocare==0.0.23.dev0

**bewest**

[15:00](https://gitter.im/nightscout/intend-to-bolus?at=5701926bd478c81e2cbcb098)

fantastic

cd ~/openapsdev to get back to your instance

**mikestebbins**

[15:03](https://gitter.im/nightscout/intend-to-bolus?at=57019329d39de41b496031ec)

awesome, thanks @bewest! So, for future updates, I follow that same procedure of cd'ing into the appropriate directories, git pull origin, thne run setup.py develop?

**bewest**

[15:08](https://gitter.im/nightscout/intend-to-bolus?at=5701946811ea211749c4605e)

right

in case of dexcom\_reader, that just converted you to the release version for that package

now in your instance:

openaps report show --cli

**bewest**

[15:10](https://gitter.im/nightscout/intend-to-bolus?at=570194ddd478c81e2cbcb0f8)

or if that errors, openaps report show --json monitor/upload-status.json

**mikestebbins**

[15:12](https://gitter.im/nightscout/intend-to-bolus?at=57019553d39de41b49603238)

openaps report show --cli did not error.

returns: openaps use cgm iter\_glucose 5 openaps use curl shell openaps use pump iter\_pump\_hours 5.0 openaps use cgm iter\_glucose\_hours 5.0 openaps use pump read\_settings openaps use pump read\_bg\_targets openaps use pump read\_insulin\_sensitivities openaps use pump read\_basal\_profile\_std openaps use get-profile shell --not-json-default settings/settings.json settings/bg\_targets.json settings/insulin\_sensitivities.json settings/basal\_profile.json max\_iob.json openaps use pump iter\_pump\_hours 5.0 openaps use pump read\_clock openaps use calculate-iob shell --not-json-default monitor/pumphistory.json settings/profile.json monitor/clock.json openaps use pump read\_temp\_basal openaps use determine-basal shell --not-json-default monitor/iob.json monitor/temp\_basal.json monitor/glucose-NS.json settings/profile.json openaps use tz clock --timezone "PDT" --adjust "missing" --date "None" --astimezone monitor/clock.json openaps use pump reservoir openaps use pump read\_battery\_status openaps use pump status openaps use ns-status shell --not-json-default monitor/clock-zoned.json monitor/iob.json enact/suggested.json enact/enacted.json monitor/battery.json monitor/reservoir.json monitor/status.json

**bewest**

[15:13](https://gitter.im/nightscout/intend-to-bolus?at=57019570e4a8384a1bbe77b3)

that's a way to see exactly how openaps is running the **use** commands when you invoke

so, eg: openaps report show --cli monitor/upload-status.json will show how it's being run

**mikestebbins**

[15:13](https://gitter.im/nightscout/intend-to-bolus?at=570195a08d2a72471b7ca7a2)

Cool. Thanks for the help @bewest, I need to run out the door now. I'll dig deeper into this tonight!

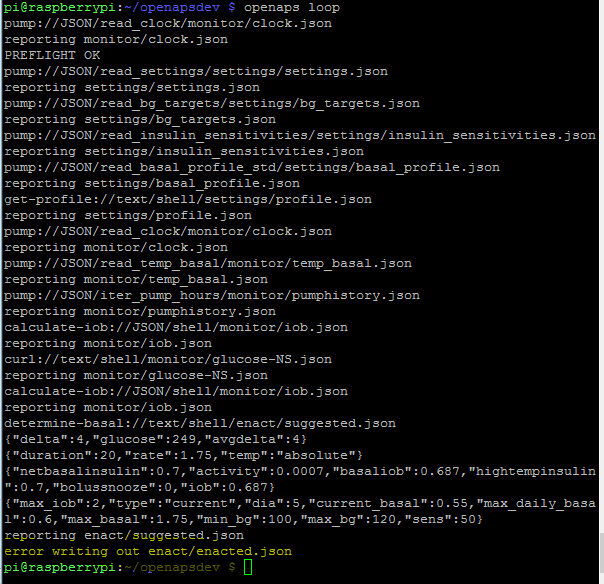
Ran:

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

Built test\_python\_read\_pump\_output.py, which enacts a temp basal from a json file, and then captures the output. Currently figuring out how to take output, and put in a json file (“enacted.json”)

Modified create\_enacted.py to

* read suggested.json
* if duration is found, then write out “duration”, “rate”, “temp” to a temp.json file
* try to set temp basal with the temp.json file
* read the output and then merge the returned dictionary with the suggested.json dictionary
* write out the new merged dictionary to enact/enacted.json



Current cron lines (**DEPRECATED, NOW CRON BACKS ITSELF UP TO OPENAPS DIRECTORY)**

$ crontab –e

\*/1 \* \* \* \* (cd /home/pi/openapsdev/; openaps loop && openaps git-push)

\*/1 \* \* \* \* killall -g --older-than 10m openaps

### Crontab test scripts:

\*/1 \* \* \* \* (cd /home/pi/openapsdev/; openaps crontest)

\*/1 \* \* \* \* echo Hello world >> ~/openapsdev/test\_log



### Get-bg alias that checks for “recency”

Ok, that should be fine, as long as you don't have ??? on the CGM. The inverse approach, which actually checks for recency, is openaps alias add get-bg '! bash -c "( openaps get-ns-glucose && cat monitor/ns-glucose.json | json -c \"minAgo=(new Date()-new Date(this.dateString))/60/1000; return minAgo < 10 && minAgo > -5 && this.glucose > 30\" | grep -q glucose && mv monitor/ns-glucose.json monitor/glucose.json ) || ( openaps monitor-cgm 2>/dev/null | tail -1 && grep -q glucose monitor/cgm-glucose.json && rsync -rtu monitor/cgm-glucose.json monitor/glucose.json )"'

Not sure if I have a version that tries cgm first and does the json date check. Would need to use a different field than dateString.

Now have alias of status-upload (must be run after loop, invokes monitor/clock-zoned and then invokes monitor/upload-status and uploads it to NS).

In treatment uploads to Nightscout, got errors trying to run alias:

openaps format-latest-nightscout-treatments

ended up running mkdir upload to create the directory

also had to change report for “model” to go to settings, not “monitor”, which I think I got from the docs somewhere.

now run:

openaps loop

openaps status-upload

openaps report invoke monitor/pump-history-zoned.json

openaps report invoke settings/model.json

openaps format-latest-nightscout-treatments

openaps upload-recent-treatments

openaps loop && openaps status-upload && openaps report invoke monitor/pump-history-zoned.json && openaps report invoke settings/model.json && openaps upload-recent-treatments

## Linux aliases I’ve created: (<https://community.linuxmint.com/tutorial/view/891>)

## <https://www.raspberrypi.org/documentation/linux/usage/bashrc.md>

nano ~/.bashrc then source ~/.bashrc

cron-stop = sudo /etc/init.d/crond stop

cron-start = sudo /etc/init.d/crond start

cron-log = tail -f /var/log/syslog | grep CRON

log = 'tail -f /var/log/syslog | grep openaps'

## Important LINKS:

<http://explainshell.com/>

<http://openaps.readthedocs.org/en/dev/docs/>

add new wifi networks: <http://www.algissalys.com/how-to/how-to-raspberry-pi-multiple-wifi-setup-through-the-command-line>

## Shut-down button

GPIO pinouts: <https://pinout.xyz/>

http://www.raspberry-pi-geek.com/howto/GPIO-Pinout-Rasp-Pi-1-Model-B-Rasp-Pi-2-Model-B

Instructions from: <https://www.element14.com/community/docs/DOC-78055/l/adding-a-shutdown-button-to-the-raspberry-pi-b>

# Adding a Shutdown Button to the Raspberry Pi B+

Version 1

Created by [ipv1](https://www.element14.com/community/people/ipv1) on Aug 4, 2015 3:05 AM. Last modified by [ipv1](https://www.element14.com/community/people/ipv1) on Aug 18, 2015 9:52 AM.

# Introduction

For a beginner to the world of raspberry pi, there are a number of projects that can become the start of something big. In this article, I discuss such a simple project which is adding a button that can be used to shutdown the raspberry pi using a bit of software tinkering. I wrote a similar article in 2013 at my blog “embeddedcode.wordpress.com” and its got its share of attention since a lot of people starting out with a single board computer, kept looking for a power button. Additionally, those who wanted a headless system, needed a way to shutdown the computer without the mess of connecting to it over the network or attaching a monitor and keyboard to it. In this article, I revisit the tutorial on how to add a shutdown button while trying to explain the workings and perhaps beginners will find it an amusing to add find more things to do with this little recipe.

# What do you need?

Here is a basic bill of materials required for this exercise.

- Raspberry Pi B+(Product Link)

- A push button

- Connecting wires

- Breadboard OR...

- Nipper & Soldering Iron and related stuff(not necessary)

The Raspberry Pi B+ can be replaced by another Pi model but for this exercise, I will be explaining things with the RPi B+ as a reference. The push button also depends on availability and preference. Do you prefer a BIG RED BUTTON? No problem… use that. The Connecting wires are required since we need to connect the button to the Raspberry Pi GPIOs. I will start with using the breadboard and upgrade to a soldered button. In case you have something like a button break out which does not need soldering, thats fine too. We will make these decisions at the end. Right now lets get prototyping.

# Step 1. Setup the RPi

We need a raspberry Pi fully setup with the latest version of Raspbian OS. Since the you probably have that done, I won’t go into the details but for those who have not done it yet, I have a few links you might like to visit.

[https://embeddedcode.wordpress.com/2013/07/10/the-geek-getting-started-guide-to-the-raspberry-pi/](https://www.element14.com/community/external-link.jspa?url=https%3A%2F%2Fembeddedcode.wordpress.com%2F2013%2F07%2F10%2Fthe-geek-getting-started-guide-to-the-raspberry-pi%2F)

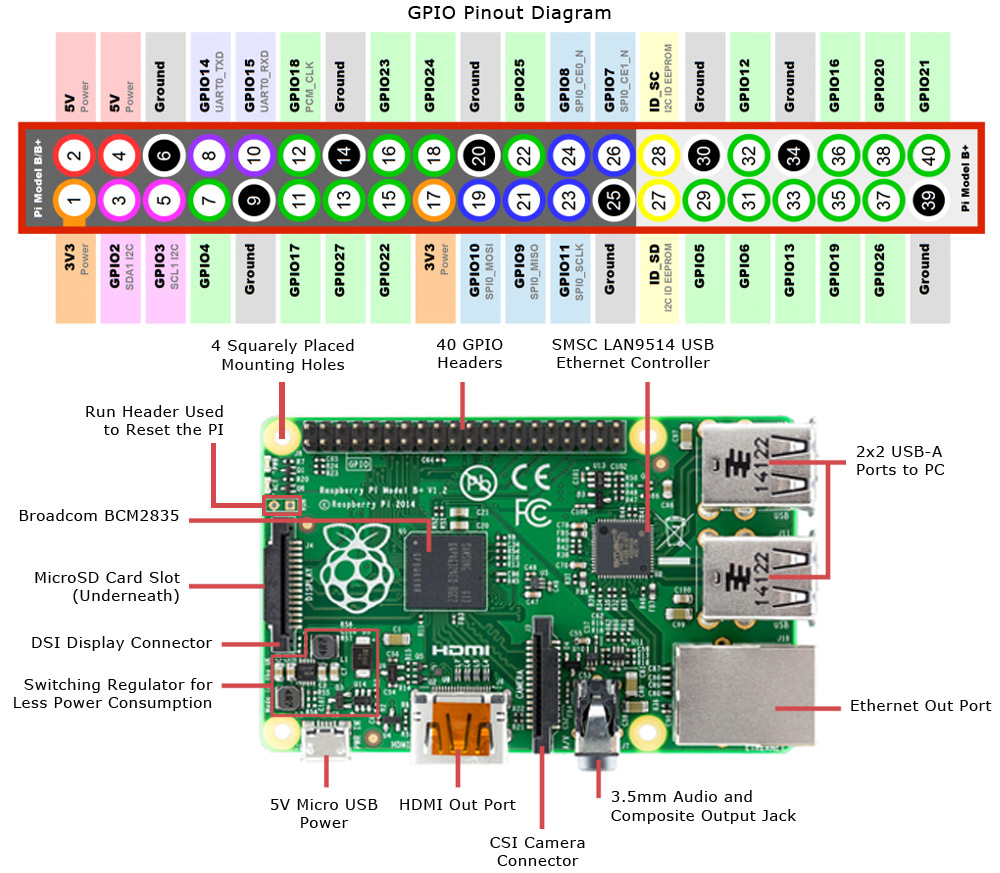
[https://www.raspberrypi.org/help/noobs-setup/](https://www.element14.com/community/external-link.jspa?url=https%3A%2F%2Fwww.raspberrypi.org%2Fhelp%2Fnoobs-setup%2F)

[https://www.raspberrypi.org/help/quick-start-guide/](https://www.element14.com/community/external-link.jspa?url=https%3A%2F%2Fwww.raspberrypi.org%2Fhelp%2Fquick-start-guide%2F)

Connect up a monitor, keyboard, mouse, ethernet cable(not necessary) and the power adaptor. Boot up the RPi and login.

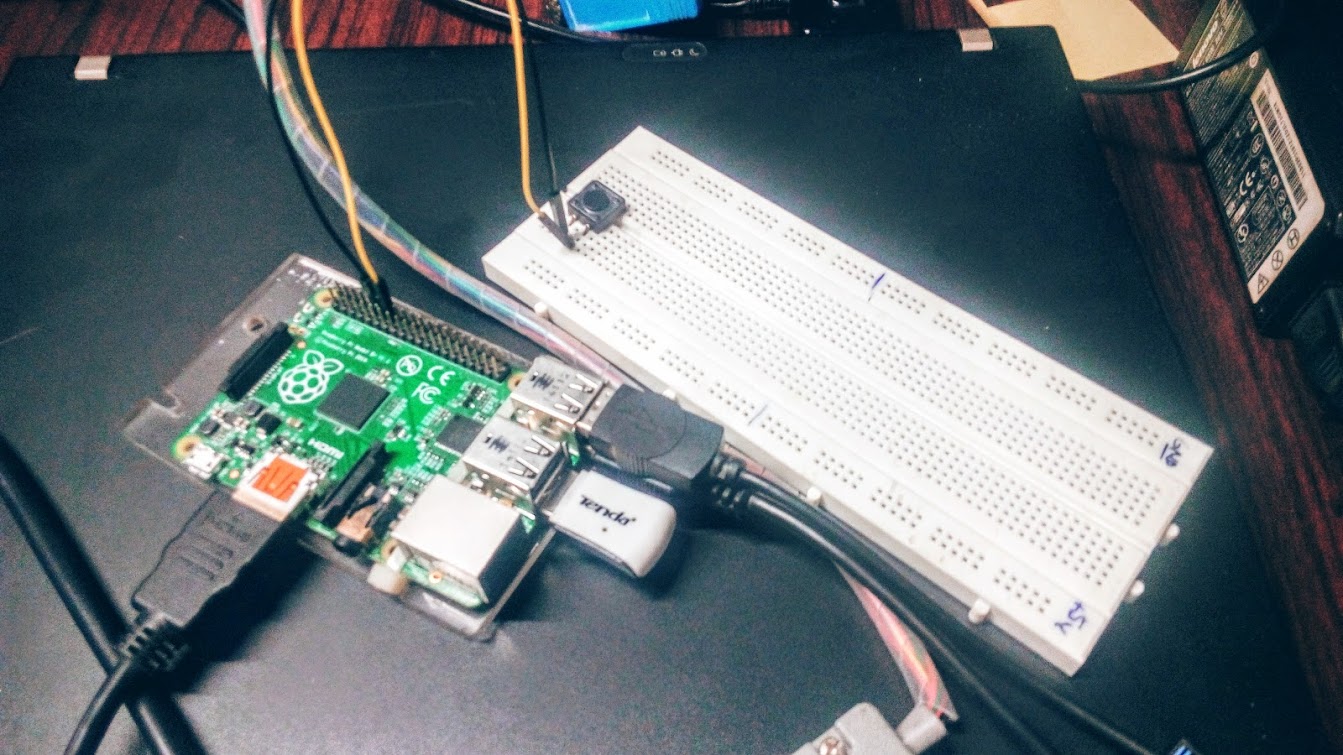
# Step 2. Connecting the button

Next we need to add the button. I added a push button to the breadboard and added the connecting wires. This part will vary with the button availability. Now where do we connect it to the RPi? Well lets take a look at the diagram below.

[](https://www.element14.com/community/servlet/JiveServlet/showImage/102-78055-1-227625/1.jpg)

From the diagram its clear that some of the GPIOs have dual roles as well such as UART, I2C, SPI etc. From these I chose GPIO18 because its near the GND or ground pin but you can choose one that fits your need. I suggest staying away from the dual purpose pins since you never know when you might want to add an SPI, I2C or serial peripheral in you next project.

The image below shows my setup and you should make sure you connected the pins right.

[](https://lh3.googleusercontent.com/7Pt-YEAgHs53P_ClRovFCyZTdPm5qdBG2hpdMm-d3WE=w1343-h755-no)

# Step 3. Writing a Python Script

There are other ways to do a shutdown script but over time, I have come to the understanding that since python has gained so much popularity, its better go this route and if you want to extend functionality to say starting or stopping a service, we can do that as well. In order to keep things organised, we create a folder called ‘Scripts’ and then an empty file called shutdown\_pi.py

[+ expand sourceview plain](https://www.element14.com/community/docs/DOC-78055/l/adding-a-shutdown-button-to-the-raspberry-pi-b)

1. mkdir Scripts
2. cd Scripts
3. touch shutdown\_pi.py

Next we need to edit the file and add the contents. Execute the command

[+ expand sourceview plain](https://www.element14.com/community/docs/DOC-78055/l/adding-a-shutdown-button-to-the-raspberry-pi-b)

1. nano shutdown\_pi.py

and then type the following into it.

[+ expand sourceview plain](https://www.element14.com/community/docs/DOC-78055/l/adding-a-shutdown-button-to-the-raspberry-pi-b)

1. #!/bin/python
2. # Simple script for shutting down the raspberry Pi at the press of a button.
3. # by Inderpreet Singh
5. import RPi.GPIO as GPIO
6. import time
7. import os
9. # Use the Broadcom SOC Pin numbers
10. # Setup the Pin with Internal pullups enabled and PIN in reading mode.
11. GPIO.setmode(GPIO.BCM)
12. GPIO.setup(18, GPIO.IN, pull\_up\_down = GPIO.PUD\_UP)
14. # Our function on what to do when the button is pressed
15. def Shutdown(channel):
16. os.system("sudo shutdown -h now")
18. # Add our function to execute when the button pressed event happens
19. GPIO.add\_event\_detect(18, GPIO.FALLING, callback = Shutdown, bouncetime = 2000)
21. # Now wait!
22. while 1:
23. time.sleep(1)

Save the file by pressing CTRL+X and then y, Enter

Note that in Python, you do not need parentheses({}) to create a sub-section. The tabbed or indented format tells the interpreter that its a sub block so please make sure that the spacing matches the above code.

Now since our python script is trying to access some GPIOs and trying to shutdown the system this scripts needs to be run as root. To test its working, enter the following command

[+ expand sourceview plain](https://www.element14.com/community/docs/DOC-78055/l/adding-a-shutdown-button-to-the-raspberry-pi-b)

1. sudo python shutdown\_pi.py

The ‘sudo’ ensures that python is given all the privileges for this run. If at this point you press the button, the RPi should shutdown. If it doesn’t then you need to check the script as well as the connections. No Pull up resistances are needed since we enable the internal pull-ups on the Pi itself.

# Step 4. Adding it to startup

We need our python script to run automatically every time the RPi starts. For this we need to do a little manual editing of the RPI generated files. Run the following command

[+ expand sourceview plain](https://www.element14.com/community/docs/DOC-78055/l/adding-a-shutdown-button-to-the-raspberry-pi-b)

1. sudo nano /etc/rc.local

This file is what gets executed everytime your RPi boots up. We need to add our python command before the last line which closes the if loop. Therefore, add the following line before the #fi at the end of the file.

[+ expand sourceview plain](https://www.element14.com/community/docs/DOC-78055/l/adding-a-shutdown-button-to-the-raspberry-pi-b)

1. sudo python /home/pi/Scripts/shutdown\_pi.py &

Please note that everything is case sensitive so Scripts is not the same as scripts. The & at the end of the command tells it to run the process in the background. If you omit it, then your login prompt probably will not appear.

Now reboot the RPi. Thats it! You are done and once you get everything working, you can start making modifications to the script for adding more buttons to start a process such as a GUI etc or kill the process, send a command over the network etc. The possibilities are endless and I would love feedback on how you made your own shutdown button.

Jason Calabrese @jasoncalabrese 13:23

I run my loop every minute, with a check to prevent it from running if it's already in progress

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

But if you do that you also need need to kill runaway processes

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

# Unit Test output

pi@raspberrypi:~/src/oref0 $ mocha -c tests/determine-basal.test.js 2>&1 | less -r

determine-basal

✓ should cancel any temp when in range w/o IOB (39ms)

✓ should temp to 0 when low w/o IOB

✓ should not extend temp to 0 when >20m left

✓ should do nothing when low and rising w/o IOB

✓ should do nothing when low and rising w/ negative IOB

✓ should do nothing on large uptick even if avgdelta is still negative

✓ should temp to 0 when rising slower than BGI

✓ should temp to 0 when low and falling, regardless of BGI

✓ should cancel high-temp when low and rising faster than BGI

✓ should cancel low-temp when eventualBG is higher then max\_bg

✓ should high-temp when > 80-ish and rising w/ lots of negative IOB

✓ should high-temp when > 180-ish and rising but not more then maxSafeBasal

✓ should reduce high-temp when schedule would be above max

✓ should continue high-temp when required ~= temp running

✓ should set high-temp when required running temp is low

✓ should stop high-temp when iob is near max\_iob.

✓ should temp to 0 when LOW w/ positive IOB

✓ should temp to 0 when LOW w/ negative IOB

✓ should temp to 0 when LOW w/ no IOB

✓ should low-temp when eventualBG < min\_bg

✓ should low-temp when eventualBG < min\_bg with delta > exp. delta

✓ should low-temp when eventualBG < min\_bg with delta > exp. delta

✓ should low-temp much less when eventualBG < min\_bg with delta barely negative

✓ should do nothing when eventualBG < min\_bg but appropriate low temp in progress

✓ should cancel low-temp when lowish and avg.delta rising faster than BGI

✓ should cancel low-temp when lowish and delta rising faster than BGI

✓ should set current basal as temp when lowish and delta rising faster than BGI

✓ should low-temp when low and rising slower than BGI

✓ should high-temp when eventualBG > max\_bg

✓ should cancel high-temp when high and avg. delta falling faster than BGI

✓ should cancel high-temp when high and delta falling faster than BGI

✓ should do nothing when no temp and high and delta falling faster than BGI

✓ should high-temp when high and falling slower than BGI

✓ should high-temp when high and falling slowly with low insulin activity

✓ should set lower high-temp when high and falling almost fast enough with low insulin activity

✓ should reduce high-temp when high and falling almost fast enough with low insulin activity

✓ should profile.current\_basal be undefined return error

✓ should bg be < 30 (Dexcom is in ???) return error

✓ profile should contain min\_bg,max\_bg or target\_bg

✓ iob\_data should not be undefined

✓ iob\_data should contain activity, iob, bolussnooze

✓ should set current basal as temp when low and rising after meal bolus

**4-17-16**: removed the section of enact.py that checked for existing temp basal and cancelled it if need be. I think that this was causing the on/off/on/off seen below. I think (yet to see) that enact/suggested has that smarts baked into it.



# Up next:

* read through unit tests
* add wi-fi management to pi
* backup sd card
* add text message for low battery and reservoir
* get wait-for-bg figured out
  + add ability to pull local cgm value first, then go to NS
* Finish drawing out system flow map of reports and alias’
* Compare my flows to Calabrese’, incorporate that which I’m missing