Nutzen des Emulators mit autoISF auf dem Smartphone

Das Konzept des Emulators auf dem Smartphone

Die Benutzung auf dem Smartphone ist viel einfacher als auf dem PC, weil es kaum Eingabeparameter gibt und die wenigen über Pop-up Menüs abgefragt werden. So wird kein Logfile oder Zeitfenster abgefragt, sondern es wird nach jedem Loop der aktuelle benutzt. Dadurch wird AAPS quasi beschattet und man kann wie im Open Loop jeweils sofort sehen, ob die SMB und TBR anders gewesen wären.

Falls die SMB höher liegen als im Master, wird der User über die Sprachausgabe darüber informiert. Man sollte sich ja sowie immer von der sicheren Seite herantasten. So kann man das Phone versteckt tragen und bei Bedarf die Lage beurteilen und ggf. zusätzlich Insulin abgeben.

Als weitere Sicherheitsmaßnahme wird der Masterlogfile jedesmal daraufhin untersucht, ob beim aktuellen Loopdurchlauf „add'l carbs req“ von AAPS im Feld Reason angezeigt wurde. Falls das so war, wird der User wiederum per Sprachausgabe darauf hingewiesen.

Das Konzept von autoISF

- Die Methode wird nur benutzt wenn COB=0, weil sonst Einflüsse von IC stören können.
- Das zu beobachtende Verhalten des BZ muß mindestens 10 Minuten angedauert haben.
- Bei der Methode „range“ wird bestimmt, seit wann der BZ sich maximal um +/- 5% geändert hat und wie der zugehörige Mittelwert ist. Die beiden Werte werden im „*glucose_status*“ als „*dura05*“ und „*avg05*“ angehängt, damit sie für die Skalierungsformeln zur Verfügung stehen. Der Durchschnittswert muß oberhalb des Targets liegen, damit diese Methode aktiv wird.
- Bei der Methode „slope“ wird bestimmt, seit wann der BZ sich durch einen linearen Verlauf gut anähern läßt. Gut heißt dabei der Korrelationskoeffizient dieser Regression muß mindestens 70% sein. Die Werte für die Länge und das Delta des besten Fits werden im „*glucose_status*“ als „*dura70*“ und „*slope70*“ angehängt, damit sie für die Skalierungsformeln zur Verfügung stehen. Der BZ muß steigen, d.h. „*slope70*“ ist positiv, damit diese Methode aktiviert wird.

Dieses autoISF reagiert viel schneller als Autotune, das ich deshalb (und aus anderen Gründen) deaktiviert habe. Nur die Obergrenze von Autotune benutze ich als Obergrenze für die Änderung von ISF.

Installation

Für python und seine Scripte geht man so vor:

- Vom Playstore das qpthon3 laden und installieren („QPython 3L – Python for Android“ von QpythonLab).
- Dadurch wird der Ordner „qpython“ auf der obersten Ebene angelegt.
- Darin in den Unterordner „scripts3“ gehen.
- Die Scriptfiles (die python Programme) determine_basal.py“, „vary_ISF_batch.py“ und „vary_ISF_core.py“ in diesen Unterordner kopieren

Der Variationsdefinitionsfile (abgeküzt VDF)

Der VDF ist der gleiche wie unter Windows und wird in den Ordner kopiert, in dem die Logfiles stehen. Im VDF wird definiert welcher Parameter wie geändert werden soll im Vergleich zur Standard Konfiguration. Für autoISF wird die Berechnung hier ausgeführt, sodaß jeder seine passenden Formeln erfinden und testen kann ohne das im python Script machen zu müssen.

```
new_parameter   autoISF_flat        True        ### additional parameter; AAPS is fix at False; enable autoISF_flat
new_parameter   autoISF_slope       True        ### additional parameter; AAPS is fix at False; enable autoISF_slope

new_parameter   maxISFAdaptation    1.7                                     ### later replace it by profile['autosense_max']
temp            dura05_weight       glucose_status['dura05'] / 30           ### 1.0 at 30 min duration
temp            avg05_weight        0.5/profile['target_bg']                ### target_bg may be modified in AAPS, e.g. by autosense
temp            deltaBG             abs(glucose_status['avg05']-profile['target_bg'])
new_parameter   prodISF_flat        temp['dura05_weight']*temp['avg05_weight']*pow(temp['deltaBG'],1)
new_parameter   liftISF_flat        min(new_parameter['maxISFAdaptation'], 1+new_parameter['prodISF_flat'])

temp            dura70_weight       glucose_status['dura70'] / 30           ### 1.0 at 30 min duration
temp            slope70_weight      max(0, 1+glucose_status['slope70']/120)  ### 0.0 at <=0, 1.1 at 12mg/dl/5min
new_parameter   prodISF_slope       temp['slope70_weight']*temp['dura70_weight']*temp['avg05_weight']*pow(temp['deltaBG'],1)
new_parameter   liftISF_slope       min(new_parameter['maxISFAdaptation'], 1+new_parameter['prodISF_slope'])
```

Es werden folgende Angaben gebraucht:

1. Die Obergrenze „*maxISFAdaptation*“ gibt an um welchen Faktor sich der ISF maximal verstärken darf. Das ist die gleiche Logik wie beim Autosene und seiner Obergrenze, d.h. beim Wert 1 bleibt der ISF wie er ist, bei 2 kann er sich maximal halbieren. Hier bitte wie gewohnt langsam erhöhen auch wenn es nur virtuell ist. Immerhin basieren die eventuellen Vorschläge für extra SMB darauf und sollen ja später im Ernstfall passen.

2. Mit „*autoISF_flat*“ als True oder False wird die Methode für konstant hohe BZ aktiviert bzw. deaktviert.
3. Die Variable „*prodISF_flat*“ enthält die prozentuale Verstärkung für den ISF.
4. Die Variable „*liftISF_flat*“ enthält den endgültigen Faktor unter Berücksichtigung der Obergrenze. Dieser sollte in einem Bereich liegen wie man z.B. in der Situation sonst das Profil erhöht hätte. Schwächung des ISF bei Unterschreiten des Targets werden ignoriert.

5. Mit „*autoISF_slope*“ als True oder False wird die Methode für lineare Anstiege des BZ aktiviert bzw. deaktviert. Der Effekt ist noch sehr experimentell und führte leicht zu Hypos. Wie oben abgebildet ist er für mich so schwach, daß er kaum durchschlägt.
6. Die Variable „*prodISF_slope*“ enthält die prozentuale Verstärkung für den ISF.
7. Die Variable „*liftISF_slope*“ enthält den endgültigen Faktor unter Berücksichtigung der Obergrenze. Schwächung des ISF bei fallendem BZ werden ignoriert.

8. Die unter „*temp*“ benutzten Variablen sind Zwischenwerte von beliebiger Anzahl und mit frei wählbaren Namen. Sie sollen die Formeln übersichtlicher und somit das Leben einfacher machen.

Nur der maximale Einfluß von entweder Autosense, liftISF_flat oder liftISF_slope wird benutzt um ISF zu verstärken.

Bei Änderungen in den Formeln empfehle ich dringend den Logfile des VDF durchzusehen, also den File „AndroidAPS.log.<*VDF_name*>.log“ im Ordner der AAPS Logfiles, um sicherzustellen, daß die Zuweisung zu den Variablen funktioniert hat wie beabsichtigt.

Das Ergebnis von autoISF kann man im Emulator Logfile nachlesen, also in „AndroidAPS.log.<*VDF_name*>.txt“ im Ordner der AAPS Logfiles. Beispiele sind:

- Beitrag von Methode slope:
  gz keep ISF as glucose trend is unclear
- Beitrag von Methode flat:
  gz ISF 33.0 avg. glucose 143.7 for 10 minutes; go more aggressive at 30.8

Starten der Emulation

Auf dem Smartphone den Button „QPython3L“ drücken, der bei der Installation angelegt wurde. Danach auswählen:

- Programs
- vary_ISF_batch.py
- Run
-

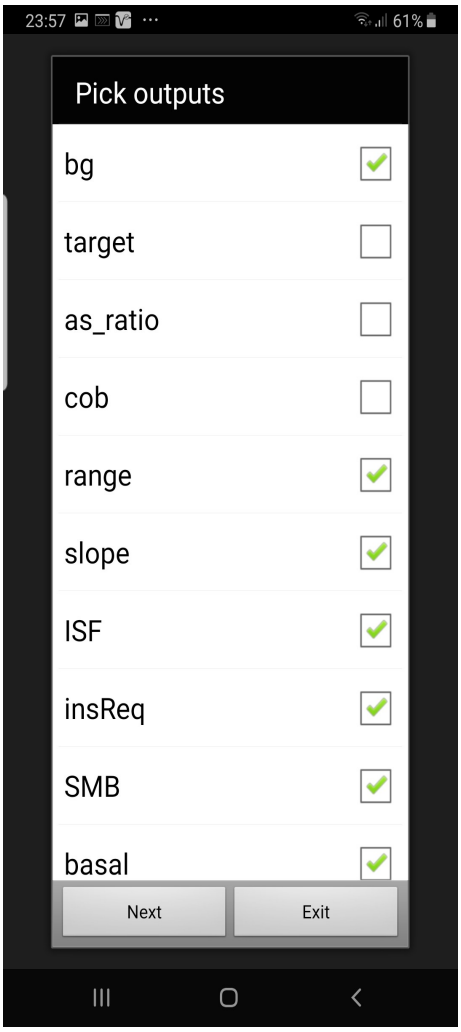
Als erstes Dialogfenster kommt die Auswahl der Sprache, die von der Sprecherin im Smartphone benutzt wird. Warum das? Auch als ich Deutsch fest vorgab, wollte die Lady manchmal partout alles Englisch ausprechen, manchmal wechselte sie sogar im Lauf der Sitzung die Sprache.

Dann musste ich das Script jedesmal neu starten. Wahrscheinlich wurde von Xdrip+ bei „Speak Readings“ da etwas umgeschaltet.

Mit „Test“ kann man sich eine entsprechende Sprechprobe anhören.

„Next“ drücken zum Fortfahren.

Das zweite Dialogfenster dient dazu alle die Spalten auszuwählen, die tabellarisch angezeigt werden. Die vorgewählte Standardauswahl braucht 93 Spalten und die sind auch im Querformat nicht auf jedem Smartphone verfügbar. Dementsprechend können weniger interessante Spalten abgewählt werden.

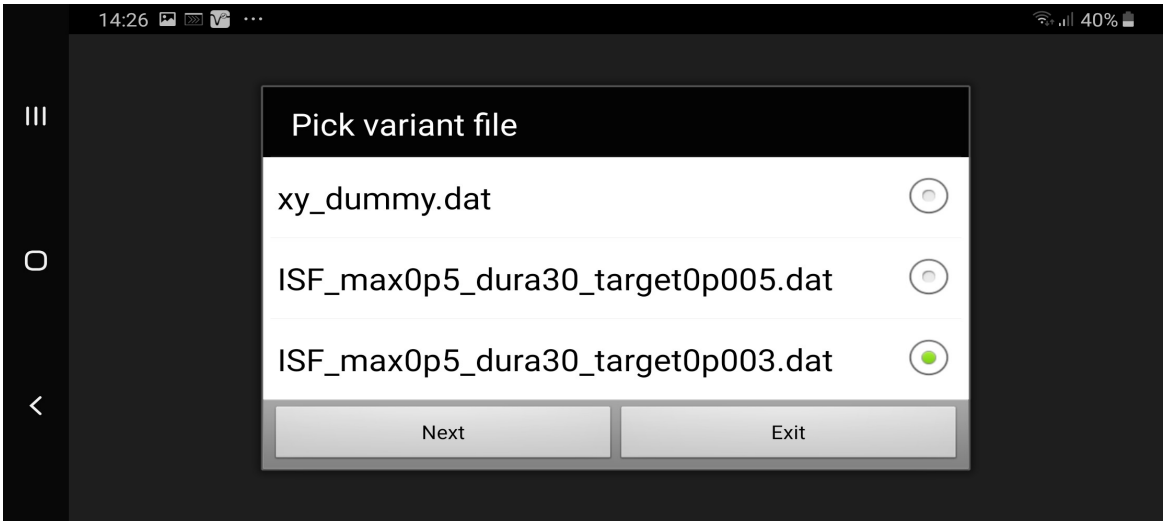


Für die autoISF Studie können folgende 7 Spalten angezeigt werden:

- „range“
  - wieviel Minuten der BZ sich kaum änderte
  - durchschnittlicher BZ innerhalb dieses Intervalls von +/- 5%
- „slope“
  - wieviel Minuten der BZ annähernd linear stieg oder fiel
  - das zugehörige Delta, also die lineare Änderung je 5 Minuten
- „ISF“
  - orig ISF wie im aktiven Profil definiert
  - auto ISF nachdem er ggf. durch Autosense modifiziert wurde
  - emul ISF wie von diesem neuen Algorithmus vorgeschlagen

„Next“ drücken zum Fortfahren.

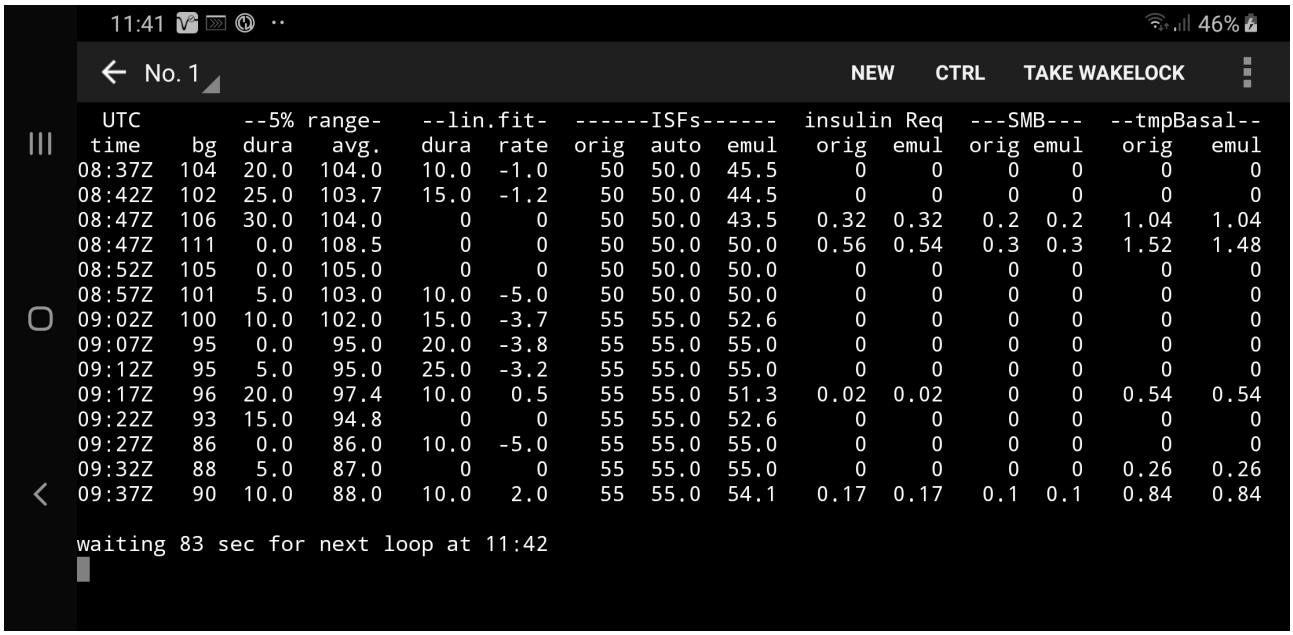
Das letzte Dialogfenster dient zur Auswahl des gewünschten VDF. Dies ist ein geeigneter Zeitpunkt um das Smartphone auf Querformat zu drehen. So werden längere Filenamen besser lesbar und im darauf folgenden Bildschirm wird ja sowieso schon die Ergebnistabelle im Querformat gezeigt.



„Next“ drücken zum Fortfahren.

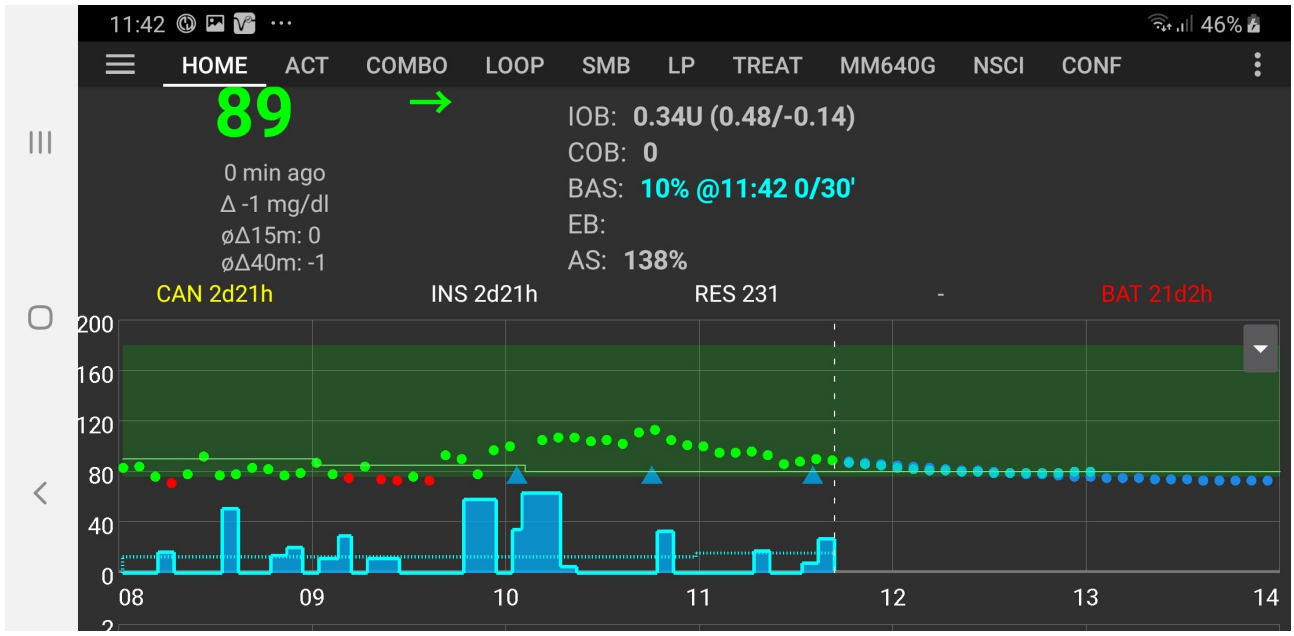
Die Ergebnistabelle

Die vom Emulator erzeugten Ergebnisfiles werden im Order der Logfiles abgespeichert. Ich bitte zu beachten, daß auf dem Smartphone kein Grafikfile erzeugt wird, weil ich für Android keine passende Bibliothek wie etwa sonst matplotlib fand.



UTC time	bg	--5% range- dura	avg.	--lin.fit- dura	rate	------ISFs------ orig	auto	emul	insulin Req orig	emul	---SMB--- orig	emul	--tmpBasal-- orig	emul
08:37Z	104	20.0	104.0	10.0	-1.0	50	50.0	45.5	0	0	0	0	0	0
08:42Z	102	25.0	103.7	15.0	-1.2	50	50.0	44.5	0	0	0	0	0	0
08:47Z	106	30.0	104.0	0	0	50	50.0	43.5	0.32	0.32	0.2	0.2	1.04	1.04
08:47Z	111	0.0	108.5	0	0	50	50.0	50.0	0.56	0.54	0.3	0.3	1.52	1.48
08:52Z	105	0.0	105.0	0	0	50	50.0	50.0	0	0	0	0	0	0
08:57Z	101	5.0	103.0	10.0	-5.0	50	50.0	50.0	0	0	0	0	0	0
09:02Z	100	10.0	102.0	15.0	-3.7	55	55.0	52.6	0	0	0	0	0	0
09:07Z	95	0.0	95.0	20.0	-3.8	55	55.0	55.0	0	0	0	0	0	0
09:12Z	95	5.0	95.0	25.0	-3.2	55	55.0	55.0	0	0	0	0	0	0
09:17Z	96	20.0	97.4	10.0	0.5	55	55.0	51.3	0.02	0.02	0	0	0.54	0.54
09:22Z	93	15.0	94.8	0	0	55	55.0	52.6	0	0	0	0	0	0
09:27Z	86	0.0	86.0	10.0	-5.0	55	55.0	55.0	0	0	0	0	0	0
09:32Z	88	5.0	87.0	0	0	55	55.0	55.0	0	0	0	0	0.26	0.26
09:37Z	90	10.0	88.0	10.0	2.0	55	55.0	54.1	0.17	0.17	0.1	0.1	0.84	0.84

waiting 83 sec for next loop at 11:42



Die Hauptanzeige der Android Verson ist die dynamisch sich ändernde Tabelle. Die Einträge decken ca. die letzte Stunde ab, ältere Einträge werden gelöscht und neue angehängt. Das funktioniert auch über den Wechsel zum neuen Logfile hinweg.

In der Windows Version wird diese Tabelle auch angezeigt, allerdings länger, weil ja pro Loop, der in den Logfiles gefunden wird, eine Zeile angezeigt wird. Diese Tabelle wird nicht als File gespeichert.

Beenden

Zum Beenden zunächst den „Zurück“-Button des Smartphones drücken und bei der folgenden Auswahl „No“, also nicht im Hintergrund weiterhin laufen lassen.

Tipps und Tricks

- Falls die Bildschirmtastatur die Ergebnistabelle überlagert oder verdrängt, einmal den „Zurück“-Button des Smartphones drücken
- Wenn man das Smartphone dreht zwischen quer und Hochkant, können sich die Spalten der Tabelle verzerren. Das wird beim nächsten Loopdurchlauf wieder neu erzeugt und sieht wieder sauber aus.
- Wenn die Tabelle anfangs nur wenige Zeilen lang ist, wurde dieses Script gestartet kurz nachdem ein neuer Logfile angelegt wurde, der ja zunächst noch weniger als 1 Stunde abdeckt.
- Die erste Spalte mit den Zeiten der Loopausführung ist in UTC Format, also z.B. in MEZ Sommerzeit um 2 Stunden hinter der normalen Zeit – und der des Smartphones.
- Manchmal verschwindet die App nach längerer Laufzeit im Nirwana. Dann habe ich sie gestoppt (Android Einstellungen > Apps > Qpython 3L > Stopp erzwingen) und neu gestartet.
- Die Meldung „add'l carbs req“ kommt auch dann, wenn eigentlich COB>“add'tl carbs req“ ist, wenn man im Bolusrechner gerade die Mahlzeit eingegeben und den Bolus abgegeben hat. Das ist so in AAPS implementiert. Ich sehe mir dann die Predictions an und beurteile, ob ich eventuell schnelle KH brauche oder auf die Resorption der bereits gegessenen KH warte. In der kommenden AAPS 2.7 wird der entsprecende Alarm vom System abgegeben und ich bin gespannt, wie diese Diskrepanz dort gelöst wird.

Status: 09-Sep-2020 @ 12:15