Full Closed Loop (FCL) using autoISF 3.0 .. V20 w DIY cockpit

0. Introduction

readme.md in github/ bernie4375 V.2.0

Exploring Full Closed Loop potential of-autoISF-3.0

Disclaimer – Important to read and understand

Authors are no medical professionals but T1 diabetics (or parents of a T1D child) who report their - limited - understanding and experience, in an effort to contribute to a growing body of knowledge, and to facilitate development of patient centered solutions.

Nothing in this site is medical advice, but meant to stimulate patient-driven self-responsible research, and is meant also to stimulate product developments by the medical industry. Anything you try to conclude for yourself you do on own risk. This is by no means a medical product but what is offered is a toolset for participating in development.

Never copy what others report to use, but investigate and adjust to your data. Neglecting safety instructions, and just using the “buttons” that are made available in a supposed “learning by doing” mode, would be very dangerous with the early development stage tools this research paper is about.

In case you choose to get deeper involved, run the system disconnected, parallel to your current glucose management, to learn its behavior before eventually considering (on own risk) to go any further. Please stay connected and share experiences, too.

Introduction

Full Closed Loop using Automations is represented in AAPS Master and in the related readthedocs since autumn 2023. (https://androidaps.readthedocs.io/en/latest/Usage/Full-ClosedLoop.html.).

Pre-requisites and the principal function of a Full Closed Loop, *without the user ever giving a bolus and without entering any carb info* are explained, also in a couple of other languages, there.

The essential points are summarized also below, in section 1.

autoISF is being developed as a much more sophisticated alternative for FCL, allowing better %TIR performance at higher degree of daily „freedom“ than simpler approaches to FCL could.

However, this demands much higher degree of involvement by the user (- as you shall see, following us through this paper. Of note, parts of this paper *marked in brown color, notably sections 5.3 and 6.3 describing functions of the "FCL cockpit"* are not implemented at launch because development focus had to be on core functions. For most of these “missing elements”, work arounds are described, often involving a similarly ease to use (but requiring some extra work in your set-up) DIY FCL cockpit (see section 6.2 and case studies 6.2 and 5.2)

With autoISF, and especially with the intention to use it for Full Closed Loop, you are in the early development area. It is therefore important to observe the disclaimer given above, and the warnings given below, as well as the hints given by the developers in the respective manuals (readme files on their Github pages. For autoISF with AAPS the main ones are https://github.com/T-o-b-i-as/AndroidAPS/ and https://github.com/ga-zelle/autoISF/).

autoISF has also been ported into an early development branch of iAPS (oref(1) for i-Phone) (https://github.com/mountrcg/iAPS).

First of all, a tip: If the following looks too complicated for you - and it's not just about understanding, but also about time requirements and discipline during implementation - you would be well advised to first try the Full Closed Loop in a simpler form with Automations (reference see above, and section 13.1): Depending on the quality of their HCL tuning they are starting from, their expectations for %TIR, and on rapid carb contents of their diet, an increasing number of people succeed in making a respectable start the first time they try using AAPS in Full Closed Loop mode.

See also the first published medical study that included 16 patients using AAPS, who found, on average, comparable %TIR performance when using a basic Full Closed Loop mode: https://pubmed.ncbi.nlm.nih.gov/36826996/

Alternatively you can use some techniques used in hybrid closed loop, such as using a pre-bolus with autoISF, or explore other early DEV variants mentioned in section 13.3, which also undergo permanent further development (Boost, AIMI, EatingNow, Tsunami).