

Pourquoi ce projet ?

Il y a quelque temps, le diabète d'un de mes proches a pas mal grimpé. Les comprimés ne suffisaient plus, et son médecin l'a fait passer sous **insuline**. Du jour au lendemain, le quotidien change : une piqûre le matin, une le soir, et surtout une glycémie à mesurer tous les matins à jeun avec un petit lecteur, un **Accu-Chek Guide** de chez Roche.

Et ce n'est pas tout. Le médecin lui a donné un protocole écrit : "si la glycémie du matin est au-dessus de tel seuil plusieurs jours de suite, on augmente la dose du soir de tant d'unités, si elle passe sous tel autre seuil, on la baisse". Bref, il faut noter chaque mesure, regarder les derniers jours, faire un petit calcul, et ramener tout ça au médecin au prochain rendez-vous.

Au début, tout se faisait sur un carnet papier. Sauf qu'un carnet, ce n'est pas pratique, ça s'oublie, on oublie de noter une valeur. Le lecteur, lui, garde pourtant **toutes** les mesures en mémoire, avec la date, l'heure, et même un marqueur "à jeun" ou "après repas". Les données sont là, dans le boîtier. Il fallait juste trouver un moyen de les sortir.

Voici comment j'ai fait.

Les solutions "officielles"

Mon premier réflexe a été de regarder ce que propose Roche. De ce que j'ai trouvé, les solutions passent par une application (Gluci-chek) smartphone reliée à un compte en ligne, ou par des logiciels pensés pour Windows. Rien qui fonctionne simplement sur un PC Linux, sans compte, sans cloud, et sans envoyer des données de santé chez quelqu'un d'autre.

Il y a aussi **Tidepool**, un projet open source très sérieux qui sait lire énormément de lecteurs de glycémie. Mais son "uploader" est une application qui tournait dans Chrome, avec un compte Tidepool à la clé. Pas vraiment ce que je cherchais pour un proche pas fan de l'informatique qui veut juste brancher son lecteur et voir ses chiffres.

La trouvaille : accuchek

En fouillant sur GitHub, je suis tombé sur [emogenet/accuchek](https://github.com/emogenet/accuchek) : un petit programme en C++ qui parle directement au lecteur en USB grâce à **libusb**, et qui recrache toutes les mesures. Son auteur avait fait le plus dur : il avait **rétro-ingénieré** le protocole de communication en lisant le code JavaScript du pilote Roche de Tidepool, et l'avait réécrit en C++ "brut".



C'était la bonne base, mais pour un usage au quotidien, il manquait des choses :

- les **marqueurs repas** (à jeun, avant repas, après repas, coucher...) n'étaient pas lus, alors que c'est justement ce qui permet de retrouver la glycémie du matin ;
- les mesures hors échelle "HI" et "LO" étaient jetées ;
- les dates étaient décalées d'une heure en été ;
- il fallait être **root** pour accéder au lecteur ;
- un paquet USB un peu bizarre pouvait faire lire le programme en dehors de sa mémoire ;
- pas de moyen simple de savoir *pourquoi* une lecture avait échoué.

J'ai donc forké le projet ([Ronnarrdd/accuchek](https://github.com/Ronnarrdd/accuchek)) et je l'ai repris en profondeur. C'est ce code que j'ai ensuite importé dans mon application de suivi, **Glucofi**.

Les acteurs de la pièce

Pour bien comprendre la suite, voici qui fait quoi :

- **Le lecteur Accu-Chek Guide** : il fait les mesures et les stocke. Branché en micro-USB, il affiche "transfert de données" et attend qu'on lui parle.
- **Le PC sous Linux** (Mageia dans mon cas) : il voit le lecteur comme un périphérique USB.
- **accuchek** : le petit programme C++. Il trouve le lecteur, discute avec lui dans sa langue, récupère toutes les mesures et les écrit en **JSON**.
- **Glucofi** : l'application graphique (GTK4 / libadwaita) qui lance accuchek, range les mesures dans une base locale, fait les graphiques, propose l'ajustement de la dose du soir selon le protocole du médecin, et sort un PDF pour la consultation.

Dans ce livre, je vais surtout parler d'**accuchek** : comment le lecteur se présente à Linux, quelle "langue" il parle, comment se déroule la conversation octet par octet, et tout ce que j'ai dû faire pour rendre le programme fiable. Je garde Glucofi pour le dernier chapitre.

Ce que ça donne au final

On branche le lecteur, on lance une commande, et on obtient ça :

```
$ accuchek > mesures.json
$ jq '.readings[-1]' mesures.json
{"id": 637, "epoch": 1790880420, "timestamp": "2026/10/01 20:47", "mg/dL": 104, "mmol/L":
5.777778, "status": 0, "meal": "fasting"}
```

Pas de cloud, pas d'application du fabricant, pas de navigateur, pas de root. Juste un câble USB et un fichier JSON.

Ni accuchek ni Glucofi ne sont des dispositifs médicaux. Ils ne remplacent ni le lecteur, ni l'avis du médecin. Toutes les valeurs importantes se vérifient sur l'écran du lecteur, et toute décision sur l'insuline reste celle du médecin.

Au programme

1. Pourquoi ce projet ? (vous êtes ici)



2. Le lecteur vu par Linux
3. La langue du lecteur : ISO/IEEE 11073
4. La conversation, pas à pas
5. Identité, horloge et marqueurs repas
6. Blinder le code : tests, traces et fuzzing
7. Installer et utiliser accuchek
8. Et ensuite : Glucofi

Révision #3

Créé 2026-10-02 19:50:20 UTC par Renard

Mis à jour 2026-10-02 20:50:35 UTC par Renard