UiT The Arctic University of Norway

Department of Computer Science

M-CDS: Mobile Carbohydrate Delivery System

Helping people with insulin-treated diabetes deal with “on the go”

Neethan Puvanendran

INF-3971 Master's Thesis in Health Technology – June 2023

Preface

When I was choosing the topic for my master's thesis, I saw this project and resonated with it immediately. As someone with two parents with T2D and at a high risk of developing T2D myself, I hoped to create something useful for those with diabetes and, if it were to come, something I could use as well. Throughout the year, I have learned more about diabetes than ever before. I also realized how much work it can be to help a group of people who know more about their disease than I do.

During my five years of being a student at UiT, I have accomplished many things that I am proud of; I worked part-time at the local student house, DRIV, as a technical director with a 50% position throughout my studies; renovated an apartment; got married; became a father; and I passed all my exams and even got as far as writing my master’s thesis and creating the physical prototype described in this project. I hope that someone else can take this proof-of-concept and create an even better product to help those with diabetes.

Acknowledgments

This thesis would not have been possible without Eirik Årsand. Thank you for dealing with me the past year and helping me tremendously with my last-minute questions and late-night emails. I am also grateful to my co-supervisors Gunnar Hartvigsen, André Henriksen, Pietro Randine, and Miriam Wolff for sticking with this project and giving me the feedback and ideas needed.

Many thanks to Ken-Arne Jensen for all your help in the IFI Workshop, teaching me how to use the 3D printer, and giving me a new Raspberry Pi when I bricked the first one.

Without Øystein Tveito's ideas for a mobile system, this thesis would not be nearly as successful. Thank you for your ideas which you happily explained in detail to me, and for helping plan the hardware aspect of the system.

Thank you to my fellow students Eric, Sigurd, and Sondre for helping me get through school when my life was at its busiest and countless hours of table tennis. I couldn't ask for a better group of friends during these years.

Thank you to my church friends and family for your prayers, my parents for their support, and my father and elder sister for proofreading.

Thank you to my wonderful wife, Sunniva, for your support and love throughout the past year. I could not have done this without you; thank you for always believing in me. Thank you to my son Caleb for all the smiles, laughter, and joy you bring to the world.

Neethan Puvanendran, June 2023.

Commit to the Lord whatever you do,

and he will establish your plans.

Proverbs 16:3

Abstract

When patients with type 1 diabetes (T1D) are physically active, they encounter an issue with keeping their blood glucose (BG) stable. Generally, their blood glucose level (BGL) will drop, causing hypoglycaemia which can have fatal consequences. The simple solution is to consume carbohydrates in the form of liquids or food, but during physical activities, it can be difficult to follow their BGL at the same time as they exercise.

This thesis presents the design and implementation of a mobile carbohydrate delivery system, M-CDS. Previous work has shown that it is possible to create a stationary carbohydrate delivery system that reads the user's BG data in real-time, gives feedback to the user when their BGL is nearing hypoglycaemia, and issues a dose of juice with 15 grams of carbohydrates. The proof-of-concept system in this thesis has the same functions but is contained within a modified CamelBak backpack. A Raspberry Pi, together with various sensors and a peristaltic pump, can transfer juice from a drinking reservoir to a drinking tube, which the user can easily drink from while physically active.

The results show that the backpack works as intended and was able to avoid a BGL under 3.9 mmol/L while testing the system with a user using physical activity, thus successfully avoiding a hypoglycaemic event.

As the system is a proof-of-concept, many things can be improved or modified to create a more robust, user-friendly, compact, and complex system. However, creating a prototype proved to be a time-costly project, whereas future work can use this project as a base to further improve it.

Table of Contents

i. List of Tables

ii. List of Figures

iii. List of Abbreviations and Acronyms

Abbreviation	**Explanation**
AI	Artificial Intelligence
AP	Artificial Pancreas
API	Application Programming Interface
BG	Blood Glucose
BGL	Blood Glucose Level
BLE	Bluetooth Low Energy

CGM	Continous Glucose Monitor
COB	Carbohydrates-on-board
DKA	Diabetic Ketoacidosis
GPIO	General Purpose Input Output
HIT	Health Informatics and -Technology
ID	Inner Diameter
IOB	Insulin-on-board
M-CDS	Mobile Carbohydrate Delivery System
ML	Machine Learning
OD	Outer Diameter
RPi	Raspberry Pi
T1D	Type 1 Diabetes
T2D	Type 2 Diabetes

1 Introduction

Physical activity and exercise are essential to maintaining a healthy lifestyle. Engaging in regular exercise has been shown to provide numerous health benefits, such as improved cardiovascular function, stress reduction, increased energy levels, and the prevention of many chronic diseases [1].

The body utilizes glucose in the bloodstream during exercise as an energy source. As physical activity progresses, insulin sensitivity increases, leading to the transfer of glucose from the blood into muscle cells [2]. This process generally results in lower blood sugar levels. However, the pancreas can counteract this effect under normal circumstances by releasing glucagon. Glucagon triggers the liver to convert glycogen into glucose, which is then released into the bloodstream and raises BGL [3]. This intricate balance between insulin and glucagon helps maintain stable blood sugar levels during exercise.

When individuals with T1D exercise, their body does not control their BGL in the same way as a person without T1D. Their pancreas cannot produce glucagon to counteract a hypoglycaemic event, causing their BGL to continue dropping [4]. They must instead actively monitor their BGL while exercising and consume carbohydrates to increase their BGL. The lack of a convenient solution for managing their BGL during exercise can pose significant challenges. These challenges may result in reduced motivation to engage in regular physical activities, as maintaining a balanced BGL during exercise requires additional effort and constant monitoring. This, in turn, can have a detrimental impact on their overall health and well-being, as exercise is crucial for maintaining a healthy lifestyle and effective diabetes management [5].

In this thesis, we propose and develop a mobile carbohydrate dispensing system specifically designed for patients with T1D. The primary objective of this system is to allow individuals with T1D to concentrate on exercising while the dispensing system effectively manages the risk of hypoglycemia by delivering carbohydrates when the user's BGL begins to drop before entering hypoglycemia. By shifting the focus away from BGL monitoring and towards the physical activity itself, this innovative solution aims to improve the exercise experience and empower those with T1D to lead healthier, more active lives.

1.1 Motivation

The research on carbohydrate delivery systems for individuals with T1D, or for delivering carbohydrates during exercise in general, is lacking. Few similar projects focus on this issue. Some attention has been given to dual-hormone artificial pancreases that can administer insulin and glucagon, but there are currently no commercially available products.

A system proposed in this thesis could greatly benefit individuals with T1D. T1D requires a delicate balance between insulin administration and carbohydrate intake, and during exercise, correct carbohydrate intake is especially important to avoid hypoglycaemia. Automating carbohydrate intake during exercise would allow individuals with T1D to reduce the stress and burden associated with maintaining a stable BGL while exercising. Additionally, it could help those with a fear of exercising due to a fear of hypoglycaemia to participate in physical activities.

On a personal note, I have two parents with insulin-dependent T2D who are physically active. This also puts me at a high risk of being diagnosed with T2D [6]. Having seen my parents struggle with diabetes throughout their lifetime, creating a system that can be of good use to them – and possibly myself – is a personal motivation for this project.

1.2 Goal and Research Problem

The goal of this master's project is to create a mobile carbohydrate delivery system for patients with T1D, which allows them to exercise and engage in physical activities without the need to focus on consuming enough carbohydrates and avoiding hypoglycaemia. The main research problem is as follows:

RP: How do we design a mobile carbohydrate delivery system?

1.2.1 Subproblems

The main problem can be broken down into several subproblems.

The main problem consists of making an existing carbohydrate delivery system, House of Carbs [7], into a mobile system. From the standalone system, two main changes need to be made:

The original carbohydrate delivery system uses an unsealed system, where the container of carbohydrates and the glass that the juice is pumped into are both unsealed containers. This allows the containers to avoid releasing pressure built up due to the transfer of liquids. In a sealed system, the containers are closed. The pressure in each container would change and cause damage if the system does not handle the change in pressure during pumping. This leads us to the first subproblem:

RP1: How can we use a sealed system to pump liquid from a container?

Taking the existing system and adding a battery to it may be a simple task, but containing the system inside a backpack presents additional challenges. The electronics must be protected from damage, and the system must be kept functional and reliable during transportation and use. In addition, in case of liquid spillage inside the backpack, the system should not short-circuit and harm the user. Thus, the second subproblem is defined as:

RP2: How can we safely encase the system in a portable container?

1.3 Assumptions and Limitations

Some assumptions and limitations have been considered throughout this master project. These factors are important to note, as they influence the overall quality and the outcome of the project.

1.3.1 Time Constraints

The limited timeframe for the master's project meant that some aspects of the system had to be prioritized over others. Designing, assembling, and programming the hardware for the system, creating a mobile application, understanding the BLE protocol, and developing a physical end product took a substantial amount of time. This limitation may have led to sub-optimal design choices or oversights in each aspect of the system.

1.3.2 Focus on Prototyping

The end product of this master's project is a prototype rather than a polished, production-ready system. Therefore, the system may have limitations such as poorly optimized code and hardware performance, lack of scalability, and the lack of mass-production focused hardware, and may need to be improved before bringing it to a broader commercial platform. However, the primary goal of this master's project was to demonstrate the feasibility of a mobile carbohydrate delivery system rather than creating a finished product for mass production.

1.3.3 CGM and Mobile Device Compatibility

The implemented system relies on Nightscout to fetch BGL data from the user. It is assumed that the user has a CGM that is compatible with Nightscout and can send their BGL data in real-time or near real-time. Although the most popular CGMs in Norway can do this natively, there are two CGMs[1] that require modified applications running on the user's phone to access BGL in real-time [8]. Non-power users of these CGMs are therefore not able to use our system.

The mobile application designed in this thesis is created exclusively for Android devices, using the Xamarin framework to simplify the development process. Due to this choice, users with iOS devices, such as an iPhone or iPad, cannot use the system. The focus on Android compatibility is a limitation, potentially excluding a portion of the intended user base.

1.4 Thesis Structure

The rest of the thesis follows the following structure:

Chapter 2: Theoretical Framework – Overview of key concepts and literature relevant to this thesis. Covers diabetes, state-of-the-art, related work and technical details of hardware and software components used.

Chapter 3: Methodology – Description of the approach and process of this thesis. Includes literature reviews, a user interview, the implementation process

Chapter 4: Requirements Specification – Presentation of the requirements for the system

Chapter 5: Design – Presentation of the system design and required elements

Chapter 6: Implementation – Description of the system development and implementation process

Chapter 7: Evaluation and Results – Assessment of the system performance and a summary of the findings

---

[1] Eversense E3 and Guardian Connect

Chapter 8: Discussion – Discussion of the results, choices made during design and implementation, and areas of future work.

Chapter 9: Dissemination – Conferences and other platforms where the work has been shared

Chapter 10: Conclusion – Summary of the work and closing remarks

2 Theoretical Framework

2.1 Diabetes

Diabetes mellitus is a group of chronic medical diseases characterized by the body's inability to produce or use insulin effectively. When an individual has diabetes, they either have a pancreas that cannot produce enough insulin or cells in their body cannot respond properly to the produced insulin, causing an imbalance in blood sugar regulation [9]. Without insulin, their blood sugar will rise, causing hyperglycaemia, which can lead to toxic acids called ketones to build up in the body, also known as diabetic ketoacidosis (DKA). This can lead to nausea, frequent urination, blurred vision, vomiting, confusion, and loss of consciousness [10]. According to the International Diabetes Federation Diabetes Atlas, in 2021, an estimated 537 million adults live with diabetes, which was also attributed to over 6.7 million deaths annually [11]. The two most common types of diabetes are type 1 and type 2 diabetes.

2.1.1 Type 1 Diabetes

Type 1 diabetes is caused by an autoimmune process in the body. The cells that produce insulin in the pancreas are attacked by the body's immune system, resulting in a reduction or complete absence of insulin production. The cause of T1D is not fully understood but is likely attributable to a combination of genetic and environmental factors [11]. Individuals with T1D depend on administering insulin daily to maintain their BGL within a safe range, usually between 4 – 7 mmol/L before meals and below 10 mmol/L after meals [12]. Excessive insulin administration can cause BGLs to drop to an unsafe level, resulting in hypoglycaemia, which can lead to impaired concentration, headaches, blurred vision, weakness, and loss of consciousness [13]. Approximately 9 % of the Norwegian population with diabetes has T1D. [14]

2.1.2 Type 2 Diabetes

Type 2 diabetes is the most prevalent type of diabetes, affecting approximately 91 % of the Norwegian diabetic population [14]. Compared to T1D, which involves the immune system attacking insulin-producing cells, T2D primarily occurs when the body becomes less responsive to insulin or the pancreas cannot produce enough insulin. T2D has a multifactorial etiology involving genetic and lifestyle factors, such as obesity and physical inactivity [15]. Symptoms of T2D can be similar to T1D; however, they often appear gradually, and some individuals may not experience any symptoms at all, leading directly to complications such as DKA, impaired vision, heart disease, and strokes [11].

2.1.3 Hypoglycaemia

Hypoglycaemia is a medical condition that occurs when an individual has an abnormally low BGL, usually below four mmol/L. It can lead to symptoms such as shakiness, dizziness, sweating, confusion, and in severe cases, loss of consciousness and seizures [16]. Hypoglycaemia occurs when there is an imbalance between the body's glucose production, glucose utilization, and insulin levels, often resulting from excessive insulin administration or insufficient carbohydrate intake [13].

Hypoglycaemia is more likely to occur during exercise as the body's demand for glucose increases. Muscles require glucose for energy, leading to rapid glucose utilization and a drop in blood glucose levels. Insulin sensitivity is also enhanced during exercise, causing more glucose to be moved out of the blood and into muscle cells, further lowering blood sugar levels [2].

Hypoglycaemia can be particularly detrimental during exercise, leading to decreased cognitive function, impaired coordination, and reduced physical performance. Additionally, experiencing hypoglycaemia during exercise may cause individuals to become fearful of engaging in physical activities, leading to a sedentary lifestyle, which can further exacerbate diabetes-related complications [5]. Severe cases of hypoglycaemia can be life-threatening, especially during exercise, as it may result in unconsciousness or seizures, increasing the risk of accidents or injury [16].

2.1.4 Continuous Glucose Monitor

A Continuous Glucose Monitor (CGM) is a device that continuously measures BGLs in the interstitial fluid, which is the fluid that surrounds the cells of the body’s tissues. A CGM usually consists of a small wearable sensor that detects the BGL under the skin every 5 to 15 minutes, a transmitter for sending the reading from the sensor, and a receiver that displays the readings to the end user. The receiver may be a physical hardware device or a mobile phone [17].

Since a CGM provides a continuous stream of BGL data, individuals with diabetes can make better decisions about insulin administration, carbohydrate intake and other activities that may impact their BGL. They can also view historical data to assess the impact of their past choices. A CGM can help them to manage their diabetes better, reduce the risk of hypo- and hyperglycaemia, and minimize the complications that arise with these conditions [18].

A CGM can also send an alert when their BGL is trending downwards, giving them enough time to take corrective action by eating a snack or adjusting insulin doses. This is especially helpful for those who do not get symptoms during a BGL fall and those with hypoglycaemic unawareness [19]. A CGM can help predict a hypoglycaemic event before it happens and allow the individual to avoid entering hypoglycaemia [20]. Four companies have CGMs approved in Norway: Abbott with the Freestyle Libre 2 and 3, Medtronic’s Guardian 4 and Guardian Connect, Dexcom G6 and G7, and Senseonics with the Evensense E3 [21].

2.2 State-of-the-art

The current market for technology related to diabetes and blood glucose monitoring, with a particular focus on physical activity, mainly consists of mobile applications that use commercially available CGMs to help users discover what affects their BGLs. With this information, they can then make informed decisions on specific diets or activities that positively impact their BGLs. This section provides an overview of some of the most popular mobile applications in this field: Supersapiens, Veri, Levels, and January AI. Although each app has its unique features, they all share a common goal of improving metabolic health, and they all rely on some form of CGM to collect and interpret the user's blood glucose data.

2.2.1 Supersapiens

Supersapiens is a sports technology company that produces the Supersapiens mobile application which allows athletes to gain insights into their glucose data in real-time. The mobile application uses the Abbott Libre Sense Sport Biosensor to provide live blood glucose data minute-by-minute, track historical BGL data, and receive real-time BGL alerts. This allows users to analyze the impact of exercise routines, carbohydrate intake, and other factors on their BGL to increase their metabolic efficiency. Additionally, the sensor can connect directly to the Supersapiens Energy Band, a smartwatch specifically designed for use with Supersapiens, freeing the user from their mobile device [22]. The Supersapiens mobile application can be seen in the Figure below.

Figure 1: Supersapiens mobile application [22]

The sensor streams data to the user's mobile phone via Bluetooth and is designed for sports use only. Supersapiens states on their website in multiple sections that the sensor and application is not intended for use in diagnosis, treatment, or management of diabetes. They also indicate that the technology used in the Libre Sense is identical to that in the Libre 1 and 2 [22].

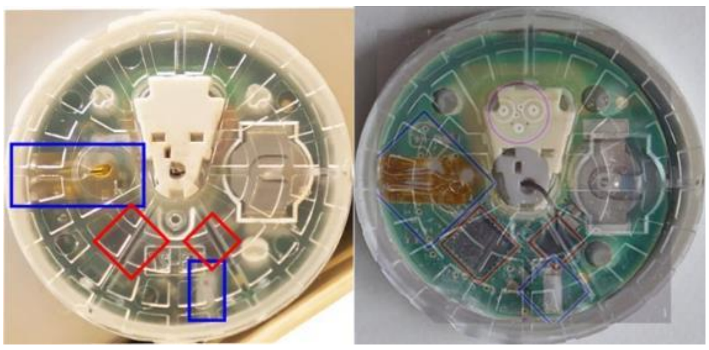


Figure 2: Comparison of the Libre Sense (left) and the Libre 2 (right). The Libre 2 image has the PCB overlayed on top of the sensor to show the similarities in hardware parts [23, 24]

There is strong evidence that the Libre Sense Sport Biosensor is a repackaged Libre 2 sensor with real-time streaming capabilities enabled. Libre 2 sensors can be hacked to retrieve BGL data, as the sensors can transmit this data but do not use Abbott's official mobile applications [25]. Figure 2 compares the two sensors, with colored boxes highlighting similarities in various hardware components.

2.2.2 Veri

Veri is a company that produces a mobile application to help individuals improve their metabolic health. The app reads their BGL via a Libre CGM and gives feedback on how different foods affect them, allowing users to find foods that help them lose weight, and reverse insulin resistance [26]. The Veri mobile application can be seen in the figure below.

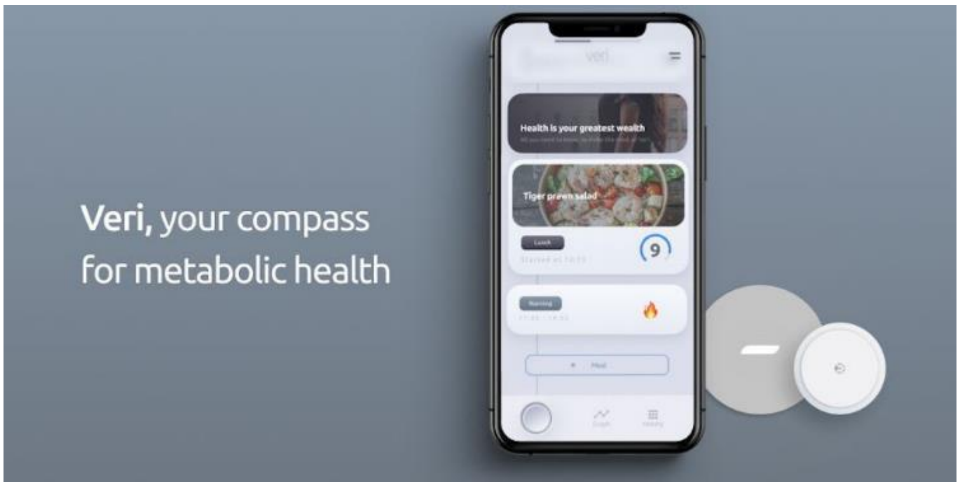


Figure 3: Veri's mobile application and the Libre 2 CGM sensor [26]

Veri uses the Libre 1 and Libre 2 sensors to access the user's BGL data. Their data is transferred using the LibreLink application, as Veri accesses it via a Practice ID, usually reserved for general practitioners to see their patient's BGL data. The user must manually scan their sensor to receive updates in the Veri application, as they are not using a sensor that streams data automatically as Supersapiens with the Libre Sense Sport Biosensor [27].

Veri also specifies that it is not intended for those with diabetes or anyone seeking medical advice [28].

2.2.3 Levels

Levels Health is responsible for creating the Levels fitness tracking program to help individuals optimize their metabolic health. The app connects to the user's CGM and reads their BGL to give real-time feedback on how food, exercise and other variables influence them and allows them to control how their body produces and uses energy [29]. The Levels mobile application and sensor patch can be seen in the figure below.

Figure 4: The Levels mobile application and sensor patch [30]

Both the Dexcom G6 and the Libre 2 may be used with the Levels application, which they differentiate as a real-time streaming sensor and a manual scan CGM. Libre 2 sensors are connected to Levels using the same method as Veri, via LibreLink. Dexcom has an official API that allows for direct integration with applications. This allows users to log in to their Dexcom Cloud account and authorizes the Levels application to read their BGL data [31].

Levels also states that their product is not intended for use with a Type 1 or Type 2 diabetes diagnosis.

2.2.4 January AI

January AI is a health application that combines CGMs with AI to provide personalized health insights and recommendations to help users align their lifestyle habits and routines to suit their body's needs. It achieves this by tracking the user's BGLs, physical activity, and food intake to provide personalized recommendations [32]. The January AI mobile application can be seen below.

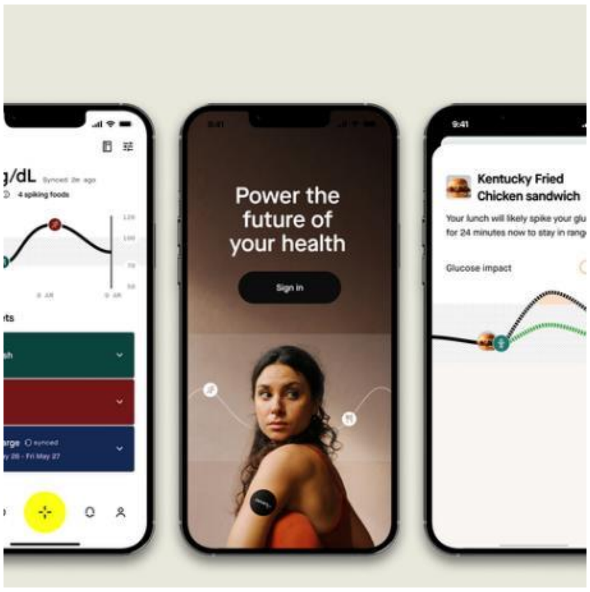

Power the future of your health
Sign in
Kentucky Fried Chicken sandwich
Glucose impact

Figure 5: January AI's mobile application [32]

By utilizing AI, January AI can provide continuous glucose estimates, predictions, and insights. The user wears a Libre CGM for 14 days, during which their BG data is gathered and sent to January AI. This information is then used to train the AI to understand the user's BGL patterns. After 14 days, the user does not need to use a CGM. The AI can predict the user's BGL solely based on the user's carbohydrate intake and physical activity data. January AI's white paper suggests a percent error of 13.0 %, which shows their AI-driven predictions' accuracy. The user may update the model by reapplying a CGM, which January AI recommends doing every three months [33].

An activity tracker must also use January AI, such as a Fitbit or Apple Watch. This provides insights into physical activity patterns, including heart rate data.

They also state that their application can be used by anyone who is not dependent on insulin, so those with type 2 diabetes, pre-diabetes, or at risk for diabetes can use the January application. This is different than the previous applications, which explicitly warns against use by individuals with diabetes [34].

2.3 Related Work

Although no other mobile carbohydrate delivery systems exist today, there are both stationary systems that are proven to work and clinical trials of dual hormone AP systems.

2.3.1 House of Carbs

House of Carbs is a stationary carbohydrate dispensing system that dispenses juice to help patients with T1D avoid hypoglycaemic events. The system consists of a stationary dosing system and a messaging bot accessible through the Telegram messaging application. A Raspberry Pi controls the dosing system and communicates with Nightscout to retrieve the user's BG data. In the case of an oncoming hypoglycaemic event, the RPi will notify the user via the Telegram bot, requesting that they get ready for a dose of juice. The RPi then pumps liquid from a container into a glass, equivalent to 15 grams of carbohydrates, using a peristaltic pump before sending a new message to the user requesting them to pick up the glass from the station and drink the juice. The system then waits 15 minutes before checking the user's BGL to see if the dose successfully helped avoid a hypoglycaemic event. [7]

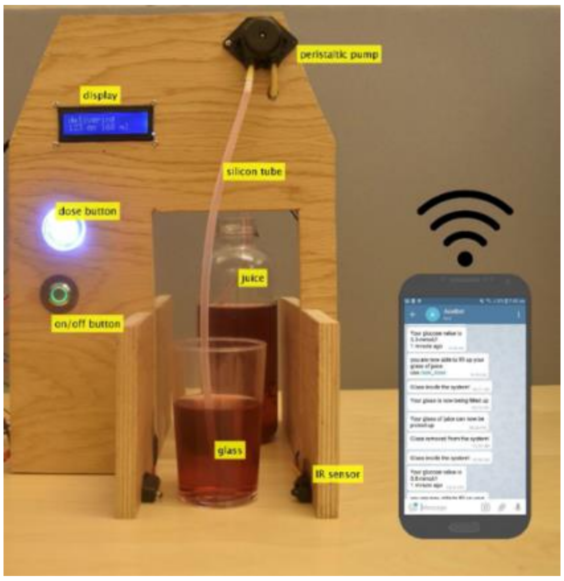


Figure 6: The House of Carbs system [7]

The user may also adjust settings by sending commands to the Telegram bot.

2.3.2 Push-button-get-candy

Push-button-get-candy is an open-source stationary carbohydrate dispensing system that uses Skittles to help patients with T1D avoid hypoglycaemic events. It also utilizes Nightscout to retrieve the user's BGL. By pressing the button on the outside of the system, the system calculates how much candy the user needs by reading their current BGL and dispenses the candy. The number of Skittles dispensed depends on the target BGL as defined in the application itself or from Nightscout, Loop or OpenAPS. The user must also input their insulin sensitivity factor to calculate their carbohydrate sensitivity factor, as some users may be more sensitive to changes in BGLs than others. The system does not automatically give the user a dose; the user must manually request it by pressing the machine's button. [35]

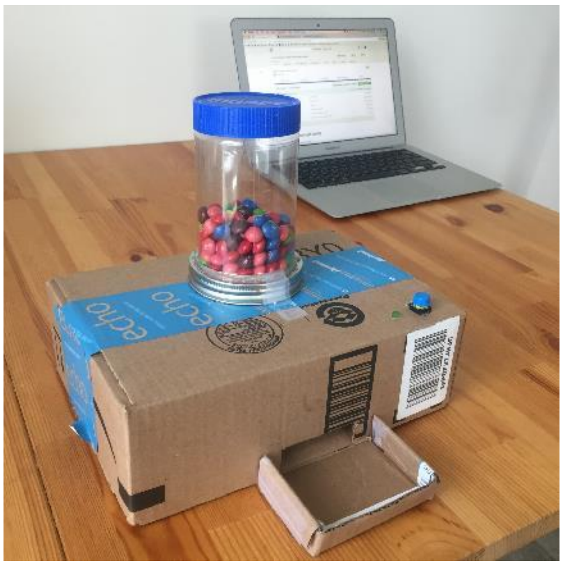


Figure 7: Push-button-get-candy [35]

2.3.3 Dual-hormone AP systems

Dual-hormone AP systems are advanced artificial pancreas systems that can administer two hormones; insulin to lower BGL, and glucagon to increase BGL. These closed-loop systems can mimic the biological pancreas more realistically than single hormone systems, as they can increase the user's BGL automatically when needed, like a normal pancreas. Automating a pancreas can improve a patient's quality of life and, due to the improved glycemic control, reduce long-term complications associated with diabetes. Incidents of hypo- and hyperglycaemia are also significantly reduced. [36]

These systems are not as widely adopted for multiple reasons. Firstly, there is a lack of widespread research and adoption of these systems. No commercially available systems are approved by the United States Food and Drug Administration, but there are many ongoing clinical trials with dual-hormone AP systems.[2]

---

[2] https://clinicaltrials.gov/ct2/results?cond=diabetes&term=dual+hormone

Secondly, tests of dual-hormone systems have been shown to increase nausea in patients due to the glucagon component of the systems [37, 38]. This could lead to patients discontinuing or not wanting to use the system. As these systems require two infusion sites for insulin and glucagon, it could lead to more pain or bleeding when applying or removing the patches. [37, 38]

2.4 Technical Background

To understand the software, hardware and technology used in the produced prototype from this thesis, knowledge of the following is required:

- **Nightscout**, a cloud-based CGM allowing for remote access of an individual's BGL, IOB and COB.
- **Raspberry Pi**: a programmable microcontroller running Linux that can communicate via WiFi, Bluetooth and be connected to various sensors
- **I$_2$C**, a serial interface protocol for communicating with devices

2.4.1 Nightscout

Nightscout is a cloud-based CGM used by those with diabetes and caretakers of those with diabetes to visualize and share data from their CGMs in real time. Nightscout is compatible with most CGMs devices, including Dexcom, Medtronic, and Abbott. The data from these CGMs are sent to Nightscout using an uploader. The uploader can be an application that runs on the user's mobile device or a script that fetches data directly from a cloud service from Dexcom, Abbott, or other CGM providers. Nightscout can also track a user's IOB and COB from their insulin pump [39].

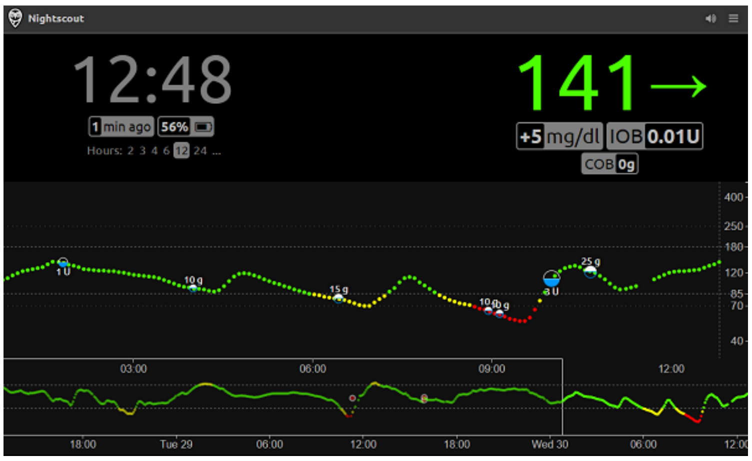


Figure 8: The Nightscout graphical interface [39]

Data from Nightscout can be fetched from other applications using its REST API. This allows for applications to integrate Nightscout's functionality in their application easily. In addition, the developer only needs to implement a single API to support multiple CGMs rather than creating a separate implementation for each CGM [39].

2.4.2 Raspberry Pi

The Raspberry Pi is a series of single-board computers created by the Raspberry Pi Foundation. They also produce smaller microcontrollers and are used for everything from robotics to carbohydrate delivery systems. Their newest flagship computer, the Raspberry Pi 4B, has a quad-core processor and supports USB 3.0, HDMI, Wi-Fi, and Bluetooth 5.0 [40].

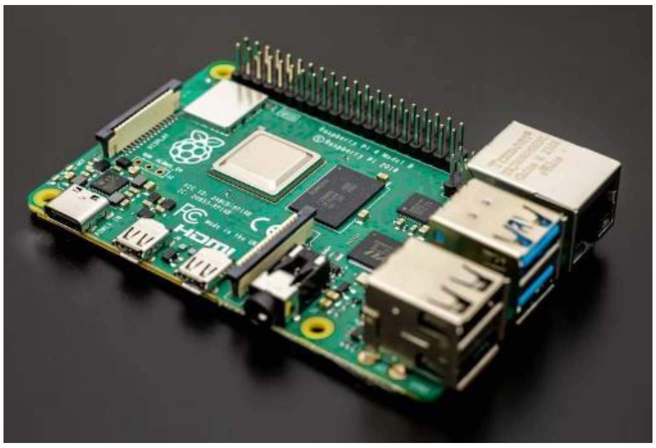


Figure 9: Raspberry Pi 4B[3]

One of the key aspects of the Raspberry Pi is its ease of use. Raspberry Pi devices typically run Raspberry Pi OS, which is Debian-based. This makes the development and setting up of the system very similar to a standard computer running a Linux distribution. With a large user base, multiple tutorials, and software, it is easy to start working with Raspberry Pis [40].

Raspberry Pis use "HATs", or "Hardware Attached on Top" add-on boards that add specialized functionality, such as sensors, motor controllers, and LED controllers. The boards are

[3] https://unsplash.com/photos/jvHymbpto1E, licensed under the Unsplash license

connected to the 40-pin header on the Raspberry Pi, which gives access to various protocols, power, and GPIO pins [40].

2.4.3 I_2C

Inter-Integrated Circuit (I2C) is a communication protocol used for transmitting data between devices, initially developed to allow simple communication between components on the same circuit board. It is a serial protocol and uses two wires to communicate: the data line, SDA, and the clock line, SCL. Multiple I2C devices can be chained together and only use these two pins for communication. Each device has its own address, which is used by the master device to communicate directly with a specific device. [41]

I2C is commonly used in Raspberry Pi HATs and sensors to connect many devices without using multiple GPIO pins.

3 Methodology

3.1 Literature Review

A literature review was conducted to get an overview of the state-of-the-art AI and ML research regarding diabetes. The literature review aimed to see if there were any recent AI or ML discoveries regarding diabetes, such as predicting BGL.

The searches for the review were performed in the academic database IEEE and PubMed. The search query structure is as follows:

Table 1: Literature review query

Section	**Partial Query**	**Reason**
1	(AI OR Artificial Intelligence OR ML OR Machine Learning)	AI and ML related papers
	AND	
2	(Diabetes OR T1D)	Diabetes or T1D related
	AND	
3	(BGL prediction OR BG prediction OR prediction)	BGL prediction, or predicting other variables

For the papers found to be included in the review, the following inclusion criteria were set:

- Uses AI or ML within diabetes research,
- Predicts future BGL using AI or ML,
- Testing is done with a single algorithm.

The exclusion criteria were as follows:

- Paper is written before 2019,
- Paper is not in English,
- Paper contains test results on humans with T1D,
- Paper is not a review.

A PRISMA diagram of the search and screening process can be seen in Figure 10.

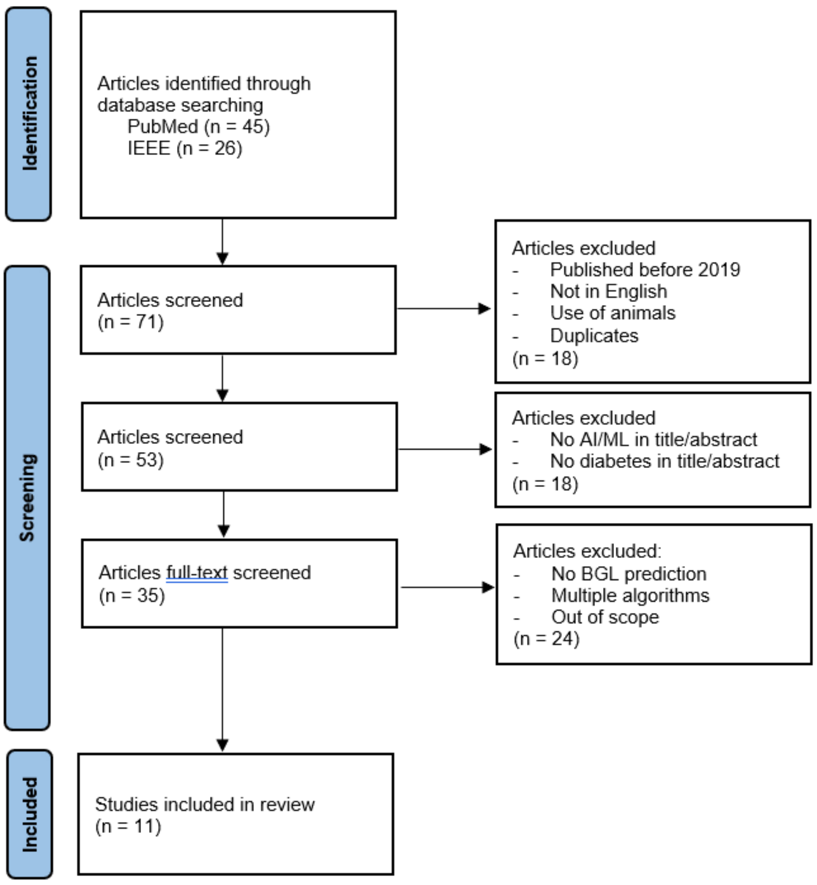


Figure 10: PRISMA diagram from the literature review.

3.2 Research Paradigm

This thesis uses the "design" research paradigm as proposed by Denning et al. in "Computer as a Discipline" [42]. This paper suggests three different paradigms for research in computer science: theory, abstraction, and design. The theory paradigm is rooted in mathematics, the abstraction paradigm is rooted in the scientific method, and design is rooted in engineering. The design paradigm consists of four steps:

1. State the system requirements,
2. State the system specifications,
3. Design and implement the system,
4. Test the system.

The requirements and specifications of the system are described in the Requirements Specification, which is then used as a basis for the Design and Implementation. A participant then tested the system to ensure proper functionality of the system.

3.3 Requirements Specification

The Volere Requirements Specification Template is used as a basis for the requirements specification of this thesis. The template is a framework used in software development to gather and organize various requirements for a project. It is used in this thesis to lay a foundation for the project's functional and non-functional requirements. Functional requirements describe what the system must do or actions it should take, whereas non-functional requirements are the properties that the system's functions must have. [43]

3.4 Diabetes Association Research Forum

diabetesforbundet
The House of Carbs
A (stationary) machine for automatically delivery of carbohydrates in liquid form

Figure 11: Presentation of the master project at Diabetes Association Research Forum 2023

The supervisor of this thesis presented a poster of this thesis at the Diabetes Association Research Forum 2023 (Diabetesforbundets Diabetesforum 2023). A speed presentation and a full presentation with discussion and questions from a panel was presented in a 10-minute time slot at the conference. The presenter asked the following questions to a panel of medical experts to gain insight, which could lead to a more effective and efficient system and create a system that would work properly for those with diabetes.

- Are there any parts of the system that should be done differently?
- How early should a hypoglycemic event be detected while the user is doing physical activity?
- Who would need such a device? Why/why not?
- General feedback

After the presentation, a member from the audience contacted the supervisor with great interest in the project and wanted to be interviewed. The interviewee was a mother who has T1D along with her son.

The interview was conducted with the following interview guide as a basis:

- How do you manage your diabetes on a daily basis? (equipment, etc.)
- Do you manage your diabetes in the same way during physical activity?
- Compared to normal activity, how does your blood sugar fluctuate during physical activity?
- Do you compensate for fluctuating blood sugar while exercising by eating or drinking?
- [After explaining the system detailed in this thesis]
    - Would you use such a system?
    - Why/why not?

The result of the interview is shown in section 7.3.

3.5 Implementation

A systematic approach was followed to develop the hardware of the M-CDS. The first stage of hardware development involved testing the individual components to ensure they worked properly and were usable for the project. If a component did not function properly or unsuitable for the system, it needed to be replaced before the final implementation stage began.

After confirming that all the individual hardware components were functional, the components were connected together using a microcontroller. At this stage, the code written for the microcontroller only tested the components working together to see if there were any conflicts between the components. Once a working prototype was achieved with all the hardware components, the system was then put into the final container, the backpack.

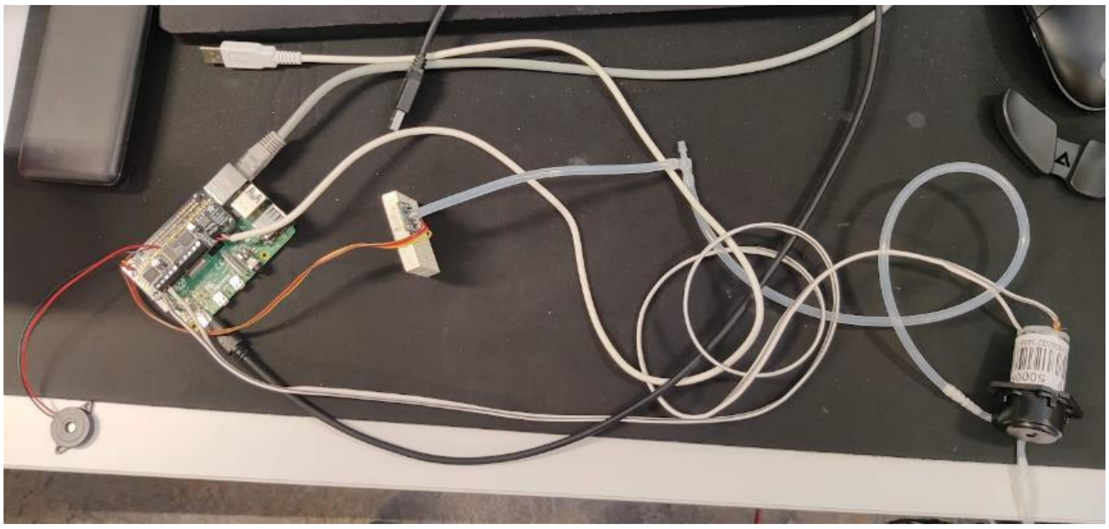


Figure 12: Testing the individual components together. Pictured is the piezo buzzer, Raspberry Pi, motor controller, pressure sensor and peristaltic pump.

The software development began with creating test programs for specific tasks, such as BLE communication between the mobile device and Raspberry Pi, making basic API calls to Nightscout and parsing the data, and figuring out a simple hypoglycaemic detection algorithm. Testing these pieces separately ensured that each function worked independently of each other. Then, all the test programs' functionality was combined into a single script and software, and code for interfacing with the hardware was written. At this stage, the hardware was already tested and integrated into a single device, leaving only the software integration left.

3.6 Tools

To create the system that is M-CDS, many vital tools and resources needed to be used.

3.6.1 3D Printer

3D printing is an additive manufacturing method where plastics are melted and added together layer by layer to create a complete piece. 3D printing allows for faster prototyping and lower costs, and model designs are available for free on the Internet from various providers. [44]

A 3D printer was an essential tool for creating parts of the system that could not be ordered quickly or had particular needs.

3.6.2 IFI Workshop

Figure 13: The IFI Workshop

The Department of Computer Science Workshop (IFI Workshop) is a workshop area designated for creating projects for the C.S. department. The workshop is equipped with various tools and equipment essential for hardware development, including soldering tools, drills, and mechanical and electronic parts. The 3D printer mentioned earlier was also located within the workshop. All the hardware components of the M-CDS backpack were assembled and created in the workshop.

3.6.3 House of Carbs

This project was based on the House of Carbs stationary carbohydrate delivery system. House of Carbs served as a foundation and inspiration for the M-CDS, and uses many of the same components and similar technology. [7]

3.7 Security

When handling sensitive data, such as BG data, it is crucial to ensure that users can trust the system and that their data is stored securely. Their data should not be leaked, and the system should be secure enough not to be breached.

The Raspberry Pi stores the user's BG data locally on the device without any unique identifiers tracing back to the user. If a malicious user were to gain access to the local database file, they would have a user’s data but without any trace back to whose data it is. All data between the Raspberry Pi and the backend server is encrypted and sent over HTTPS.

3.7.1 Application to UiT’s DPO

UiT’s Data Protection Officer (DPO) was contacted to receive advice regarding the data security and privacy for this master project. The project's scope and the system being developed were described in the email, as well as how the user’s data was to be stored and sent. The email in full can be seen in the Appendix, and the answer from the DPO in section 8.3.

3.7.2 Application to Sikt

Sikt – Norwegian Agency for Shared Services in Education and Research is a public administrative body. When doing scientific research involving test subjects, an application stating how the subjects’ personal data will be used and stored usually must be sent to Sikt.

An application[4] was sent to Sikt sent on April 20th stating that a selection of T1D patients would use the M-CDS system, and that their BG data would be collected to use the system. A complete detail of the application is available in the appendix.

4 Requirements Specification

The functional and non-functional requirements for the system are described in this section, following the Volere Requirements Specification Template [43].

4.1 Functional requirements

The project includes hardware and software, and thus the functional requirements are intertwined. Each requirement below uses a subset of the properties defined in the Volere Requirements Specification Template:

- **Requirement Number**: Unique identifier for the requirement.
- **Description**: Definition of the required functionality.
- **Rationale**: Why the requirement is needed.
- **Fit Criteria**: How to measure or conclude that the requirement is met.

[4] Sikt refers to applications as «Notification forms» in English, but application is a better fitting term

- **Priority**: How necessary this requirement is for the system: low, med, high.
- **Dependencies**: What other requirements is this requirement dependent on.

Table 2: Functional requirements for the M-CDS

#	Description	Rationale	Fit Criteria	PRI	Deps.
1	The system should fetch real-time BG data from an external source	Importing BG data is necessary for the system to predict a hypoglycaemic event	The system should successfully read the BG data from an external source	High	
2	The system should give the user the ability to drink juice from a container or other means	To counteract a hypoglycaemic event, the user needs carbohydrates here in the form of juice	The physical system gives the user juice through some means	High	3
3	The system should pump juice continuously to a container	To give the user a specific amount of juice to drink	Juice is pumped normally to a container	High	2
4	It shall be possible to change the thresholds and settings of the system	To allow the user to change settings based on their situation	A settings file or configuration page should be available for the user	Med	
5	The system should detect an oncoming hypoglycaemic event	Avoid the user entering hypoglycaemia	The system detects when the users BGL is falling and issues a dose of juice	High	1
6	The system shall administer a dose of carbohydrates when a hypoglycaemic event is detected	Give the user a dose to avoid hypoglycaemia	The system starts the dosing system before a hypoglycaemic event occurs	High	5
7	The system should allow the user to request a dose of juice	If the user manually requires carbohydrates	A physical button allows the user to request a dose	Low	2, 3
8	The system should allow the user to stop a dose of juice	If the user does not need a dose or wants to stop it prematurely	A physical button allows the user to stop the dose	Med	2, 3
9	The system shall warn the user of different events audibly	To give the user audible feedback on what the system is doing	A speaker or buzzer plays back audio or sounds during different system events	High	5

10	The system shall be powered by a battery	To allow the system to be portable	A battery powers the system	High	
11	The system shall be autonomous, requiring minimal user input	To make the system as simple as possible for any user	The system can recognize an oncoming hypoglycaemic event, give a dose and finished it without user input	High	5, 6
12	The system shall be able to access the user's physical activity data	To enable the system only when the user is physically active and use that data to assist in hypoglycaemia detection	A smartwatch or similar device is used to fetch physical activity data	Low	

4.2 Non-functional requirements

This section describes four of the different non-functional requirements from the Volere Requirements Template for this project.

4.2.1 Look and Feel Requirements

As this system is a prototype, no specific look and feel requirements exist. The appearance and style of the product are not essential for ensuring a working prototype.

4.2.2 Usability and Humanity Requirements

The system should be user-friendly for people of all ages, but the target audience is adults over 18. The system should give necessary feedback to the user on what it is currently doing and should be simple enough for the user to remember how it works easily.

The system should enable personalization through custom settings for different variables and thresholds. It should also be easy to learn to use the system, and users should be able to learn to use it quickly. Long, formal training should not be necessary to use the system.

The system should hide the details of how it is constructed from the user, but the user should instinctively understand the system's mission.

4.2.3 Performance Requirements

The system should give a dose of juice *as soon* as the lower threshold BGL is reached. It should poll for BG data every minute to ensure that it has an updated reading from the user. Any other sensors used in the system should be polled as quickly as possible to ensure updated values, as the system needs to run in real time.

The system should not give doses that can pose a risk of hyperglycaemia, or doses at the wrong time. It should not fail to provide a dose if the conditions are met.

The system should give accurate juice doses, within ±1 g of carbohydrates. In case of error, the system should safely stop and ensure that no components continue to operate. The system should restart automatically in case of an error. The system should run on the battery for as long as possible and safely shut down if the battery were to lose power.

4.2.4 Security Requirements

The system should be accessible only by the user using the system. Data transferred from the system to external servers should be encrypted and protected from abuse. The system should not leak data to malicious users.

5 Design

In this section, both the initial design and the final design are explained, with the differences between the two are highlighted.

5.1 Initial Design

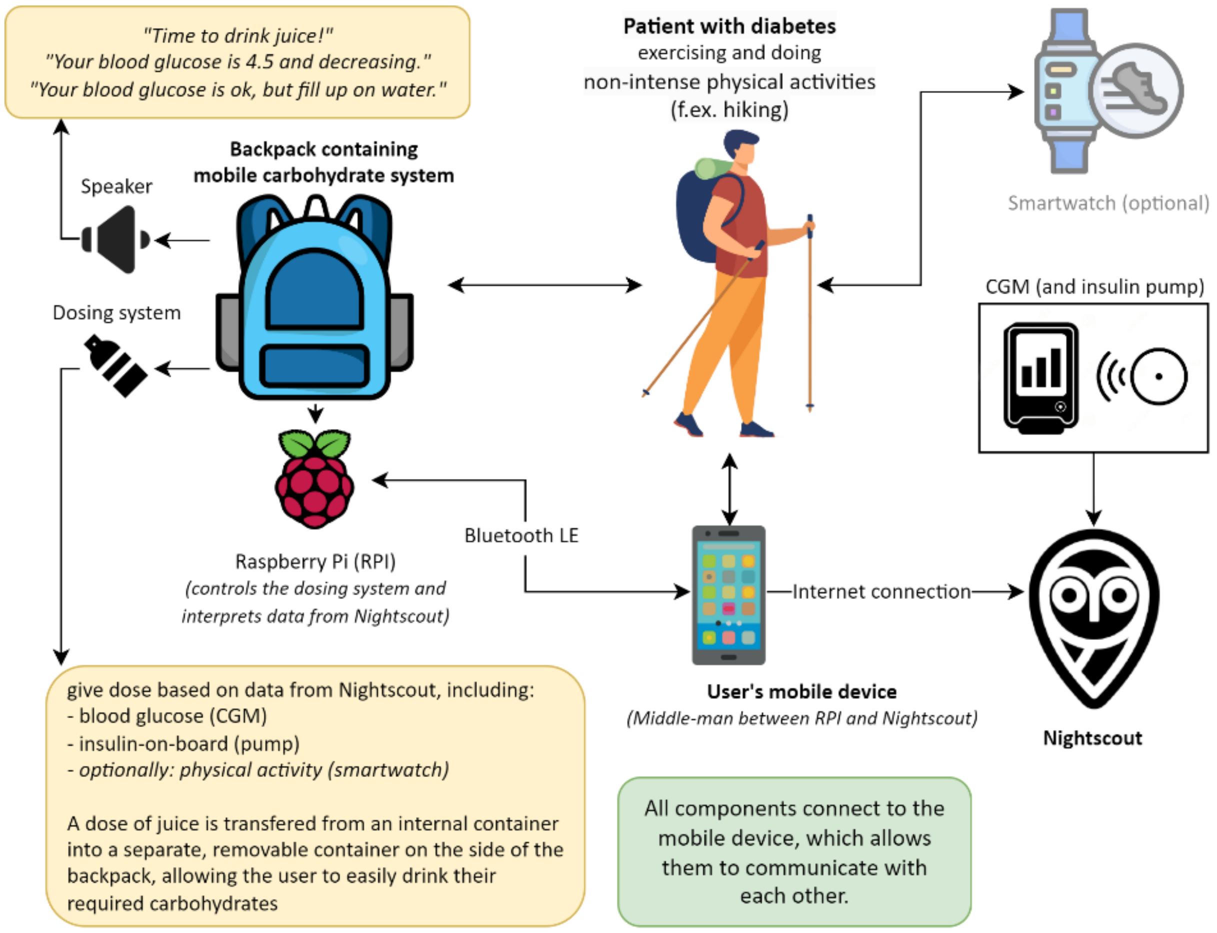

"Time to drink juice!"
"Your blood glucose is 4.5 and decreasing."
"Your blood glucose is ok, but fill up on water."
Patient with diabetes
exercising and doing
non-intense physical activities
(f.ex. hiking)
Smartwatch (optional)
Backpack containing
mobile carbohydrate system
Speaker
Dosing system
CGM (and insulin pump)
Raspberry Pi (RPI)
(controls the dosing system and interprets data from Nightscout)
Bluetooth LE
Internet connection
User's mobile device
(Middle-man between RPI and Nightscout)
Nightscout
give dose based on data from Nightscout, including:
- blood glucose (CGM)
- insulin-on-board (pump)
- optionally: physical activity (smartwatch)
A dose of juice is transfered from an internal container into a separate, removable container on the side of the backpack, allowing the user to easily drink their required carbohydrates
All components connect to the mobile device, which allows them to communicate with each other.

Figure 14: The initial design of the M-CDS, from the capstone project [45]

The initial design had three main components: the backpack, containing the carbohydrate dosing system; the mobile phone, running an app that would allow for configuration of the system and send BG data to the backpack; and data collectors, in the form of a CGM, insulin pump and smartwatch for use in hypoglycaemia prediction.

5.1.1 M-CDS Backpack

The backpack in the initial design was a generic backpack to house the entire mobile system. Inside the backpack would be the Raspberry Pi, speaker, dosing system, and other necessary components (battery, sensors, etc.).

The Raspberry Pi controlled the dosing system and received data from Nightscout. Since the Raspberry Pi would not have access to the Internet in a remote area, it would connect to the user's mobile device using BLE, and through our mobile application, it could receive data from Nightscout. Using this data, it would then calculate when the user needed a dose, and if required, then give the user this dose.

The speaker was designed to be the only feedback to the user on what the system is doing. This ensures that the user needs to do as little as possible to use the backpack because simplicity and autonomy are critical elements of the system. If the user needs to take a dose, stop drinking, or restart the system, a voice from the speaker will tell them.

Initially, the dosing system was planned to be a removable bottle on the side of the backpack. This would allow the user to easily take the required dose quickly and continue with their physical activity. The juice would be pumped into the bottle from an internal container to ensure that only the required amount of juice for a dose would be in the bottle. The Raspberry Pi would use the user's BGL from their CGM, the current amount of IOB, and physical activity data from their smartwatch to figure out whether a dose is required and how much juice to dose to the user.

The system could also notify the user to hydrate and pump water into the bottle rather than juice. If the user has a smartwatch, the system can access the user's physical activity data, enabling it to calculate and predict the optimal time to refill the liquids.

5.1.2 Mobile application

Individual diabetes patients react differently to carbohydrates and experience fluctuating BGLs differently. To adapt the system for different users, the system must be configurable. By developing a mobile application that can connect to the Raspberry Pi using BLE, we can allow for different configurations. The mobile application is also used as a middleman for all the data from Nightscout to the Raspberry Pi, as the Raspberry Pi cannot connect to the Internet without a Wi-Fi connection, and 4G adapters for the Raspberry Pi can be expensive and difficult to setup. Mobile phones usually have a 4G or 5G connection and are always online, making it easy to fetch the data from Nightscout via their phone.

5.1.3 Data collection

The user's BGL, IOB, and COB can all be collected through the Nightscout application. BG data is collected from the user's CGM, IOB from the user's insulin pump, and COB from the user's data input to their insulin pump. If the user has their own Nightscout instance, we can configure the system to use it instead of our server. This data is stored on the Raspberry Pi for logging purposes and is used for the hypoglycaemia prediction. If the user has a smartwatch, the data can be used to detect physical activity, heart rate, and other variables. This data could then be used in hypoglycaemia prediction or seeing if the user needs to take a break.

5.2 Final Design

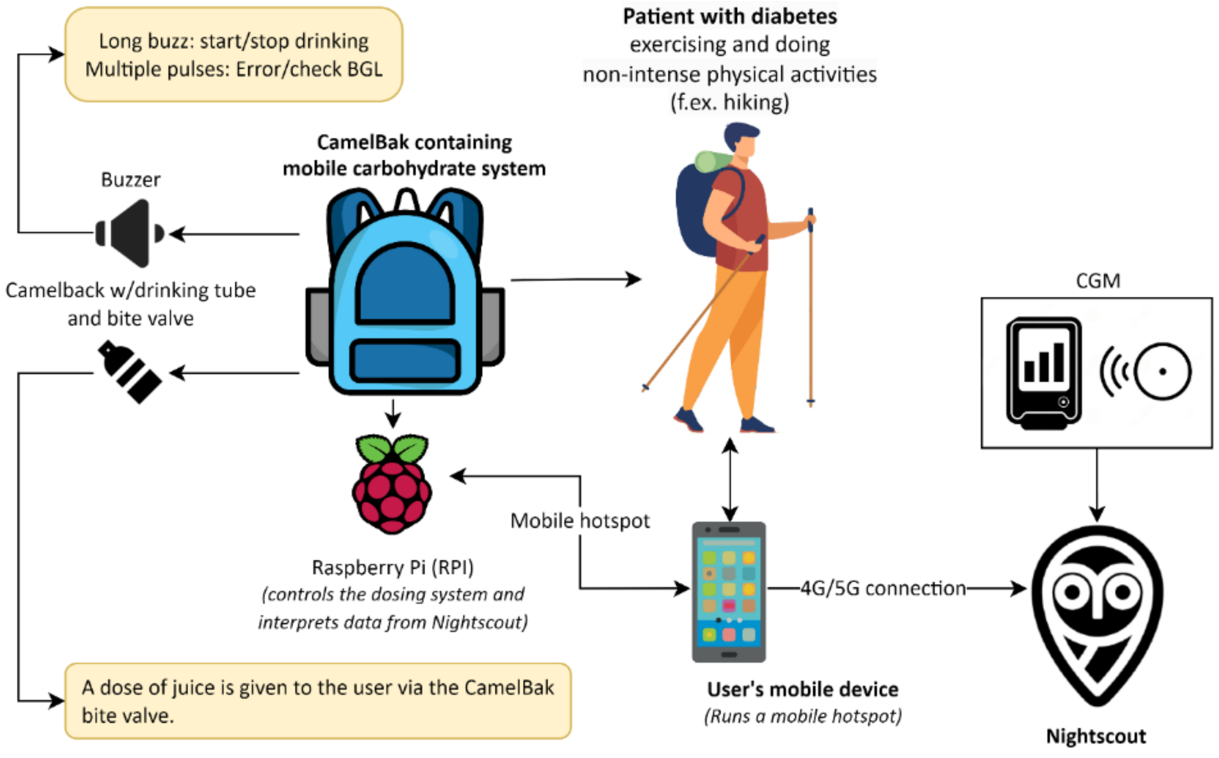

Long buzz: start/stop drinking
Multiple pulses: Error/check BGL
Patient with diabetes exercising and doing non-intense physical activities (f.ex. hiking)
CamelBak containing mobile carbohydrate system
Buzzer
Camelback w/drinking tube and bite valve
CGM
Mobile hotspot
Raspberry Pi (RPI) (controls the dosing system and interprets data from Nightscout)
4G/5G connection
A dose of juice is given to the user via the CamelBak bite valve.
User's mobile device (Runs a mobile hotspot)
Nightscout

Figure 15: Finalized design of the M-CDS

The final design of the M-CDS system has only a couple of minor variations from the initial design. Figure 15 shows the updated design with changes and can be compared with Figure 14.

Firstly, on the left side, we see that the speaker has been replaced with a buzzer and the dosing system with the CamelBak with a drinking tube and bite valve. A buzzer was simpler to program than a robotic voice through a speaker and requires less hardware to implement. The dosing system and backpack have been finalized and chosen, and the separate, removable container has been replaced with the CamelBak drinking tube and bite valve.

A mobile hotspot is used instead of BLE between the Raspberry Pi and mobile device. This design choice was made far into the implementation stage when it became apparent that the complexity of the BLE solution was not feasible to be completed within the project's time constraints. The mobile device shares its 4G/5G connection with the Raspberry Pi.

Finally, only CGM data is used in the new design. Insulin pump data and smartwatch data are not used.

5.2.1 Updated dosing system

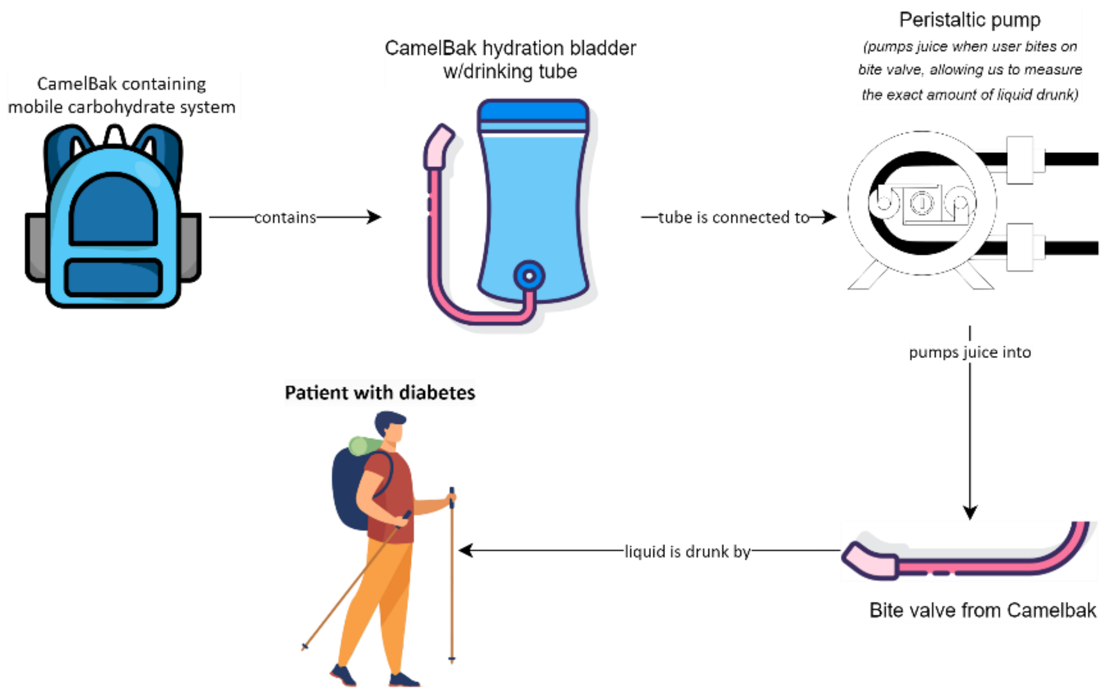

CamelBak containing mobile carbohydrate system
contains
CamelBak hydration bladder w/drinking tube
tube is connected to
Peristaltic pump
(pumps juice when user bites on bite valve, allowing us to measure the exact amount of liquid drunk)
pumps juice into
Bite valve from Camelbak
liquid is drunk by
Patient with diabetes

Figure 16: Overview of the new dosing system

A direct dosing system is used instead of the separate container on the side of the backpack. Figure 16 describes the dosing system. The CamelBak backpack contains a drinking reservoir with a drinking tube. The tube from the reservoir is connected to a peristaltic pump that pumps juice into the drinking tube with a bite valve on the end of it. The user can then drink the liquid by biting on the bite valve.

The mobile application in the final design has been removed, as it was no longer needed with the mobile hotspot. This also simplified the entire process, as no mobile application needed to be developed. The Raspberry Pi can instead handle all communication with Nightscout.

6 Implementation

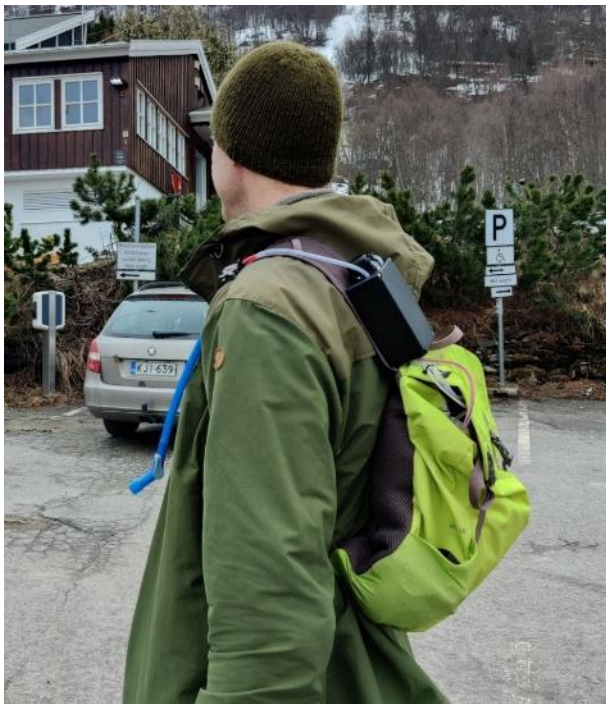


Figure 17: The implemented system. Visible are the backpack, pump housing, and drinking tube.

The final implementation uses a modified CamelBak backpack to encase the entire system, as shown in Figure 17. The backpack contains a CamelBak Crux Reservoir, modified to integrate a peristaltic pump between the reservoir and the drinking tube. The system is controlled by various scripts written in C and Python running on a Raspberry Pi, which is housed in a 3D-printed enclosure. The Raspberry Pi also controls the dosing system and retrieves data directly from Nightscout through a Wi-Fi hotspot. In addition, the CamelBak also contains several necessary components, such as a motor controller, pressure sensors, and a piezo speaker. The entire system is powered by a 20000mAh USB power bank, allowing simple recharging with any USB-C charger.

The system is highly autonomous, minimizing user interaction other than switching the device on, nor does it require a mobile device to operate. A single button on the outside of the backpack allows the user to interact with the system when needed for specific actions.

For data gathering, a server running an instance of Nightscout is deployed to collect BG data from the user. The user may use any CGM that can send data in real-time to Nightscout. A Freestyle Libre 3 CGM was used during testing, allowing a new reading to be sent every minute.

The server also runs a script to fetch data from LibreLinkUp[5], Libre's cloud patient monitoring system.

6.1 M-CDS Backpack

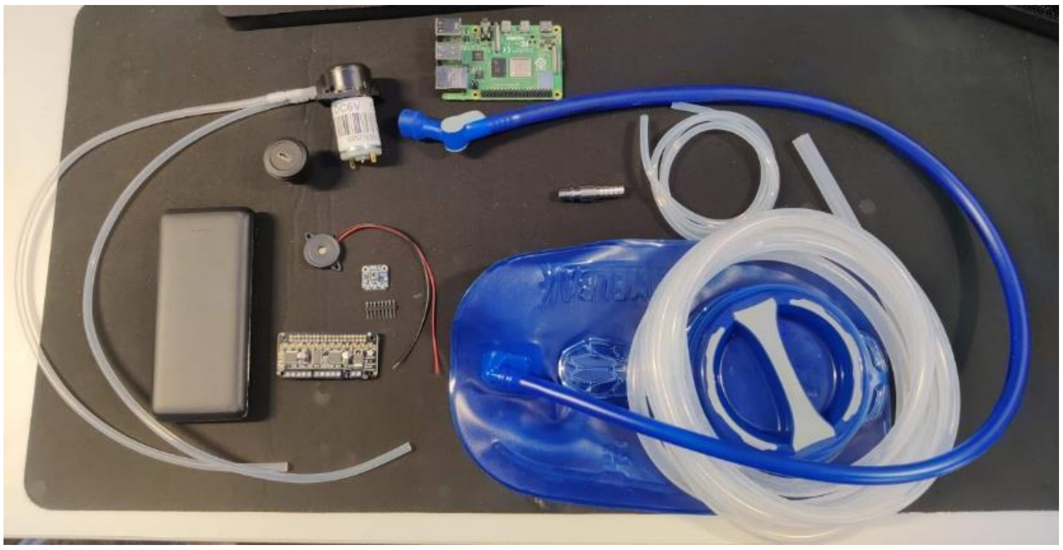


Figure 18: Some of the various parts inside the M-CDS Backpack.

Figure 18 shows some of the different parts used for the M-CDS backpack, and Table 3 provides a complete overview of the system parts.

Table 3: Overview of the parts included in the M-CDS backpack

Part	**Usage**
Camelbak Arete 18 w/ 1.5L Crux Reservoir	*Contains the entire system, including the juice that the user drinks*
Raspberry Pi 4B	*Controls the various hardware components, dosing system and communicates with Nightscout*
Adafruit DC & Stepper Motor Bonnet	*Controls the peristaltic pump motor.*
Adafruit MPRLS Ported Pressure Sensor	*Reads the pressure in the drinking tube*
MPL3115A2 I2C Barometric Pressure/Altitude Sensor	*Reads the ambient pressure*

[5] https://github.com/timoschlueter/nightscout-librelink-up

Adafruit Perma-Proto Board	*PCB where various hardware components are connected*
6V Peristaltic Pump	*Non-invasive pump, pumps liquid from the reservoir into the drinking tube*
20000mAh Powerbank	*Powers the entire system*
Switches and buttons	*Powers on/off the system, button allows user to give input when required by the system*
Tubing connectors and silicone tubes	*Used for connecting tubes of different sizes together, transfers the liquid from the reservoir to the drinking tube*
3D printed parts	*Solved problems that were fixable with custom parts*

6.1.1 CamelBak

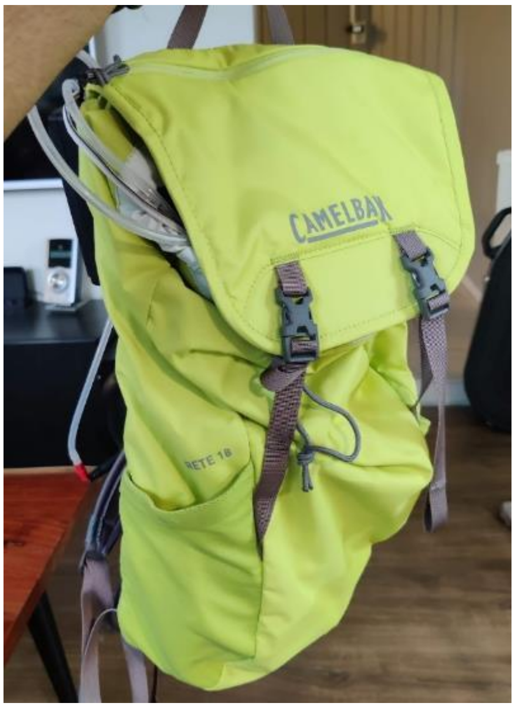


Figure 19: CamelBak Arete 18 backpack, modified

The CamelBak Arete 18 is an 18L outdoor backpack by CamelBak that contains a 1.5L Crux drinking reservoir, as seen in Figure 19. The backpack has a dedicated pocket for the reservoir, two pockets on the sides, and plenty of space inside. Modifying this existing system saved time in demonstrating that the system is viable rather than focusing on creating the perfect backpack. The drinking reservoir that comes with the CamelBak can be seen in Figure 20 below.

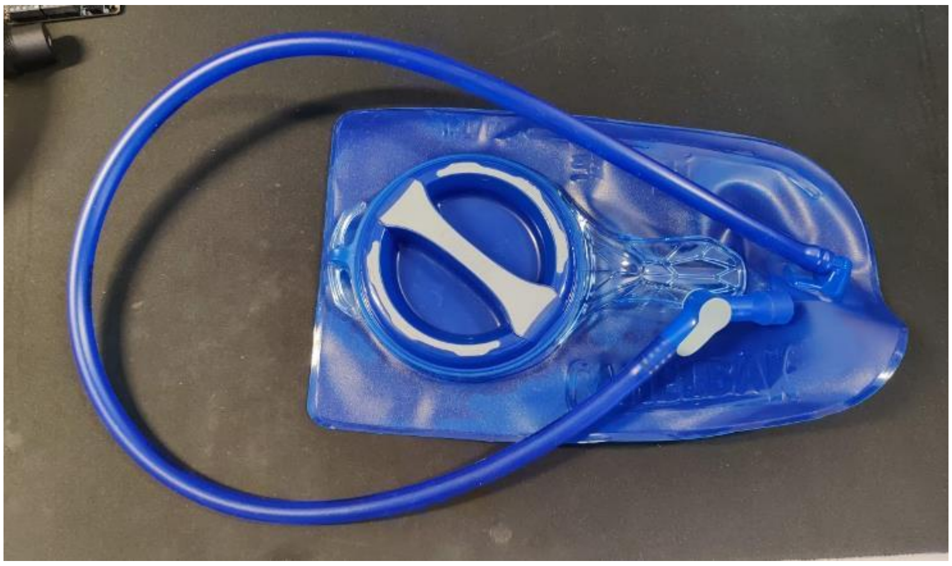


Figure 20: The drinking reservoir, with drinking tube and bite valve.

The backpack was modified to allow cables to pass through the sides of the backpack to connect the required button, switch, and charging port. The pump housing was also attached to the left shoulder of the backpack by drilling small holes in the shoulder arm and securing it with machine screws with nuts and washers, as seen in Figure 21.

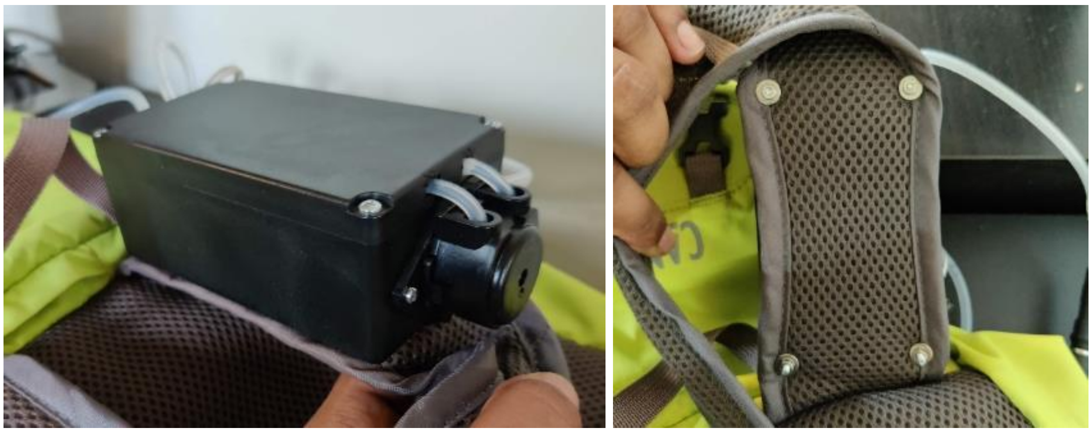


Figure 21: Peristaltic pump housing attached to CamelBak

The drinking reservoir needed to be modified, as the required peristaltic pump had to be connected between the reservoir and the drinking tube. The entire tube was cut, and the section by the reservoir was shortened to ensure as quick of a pathway for the liquid as possible. The drinking tube was then reattached after the peristaltic pump by using tubing adapters.

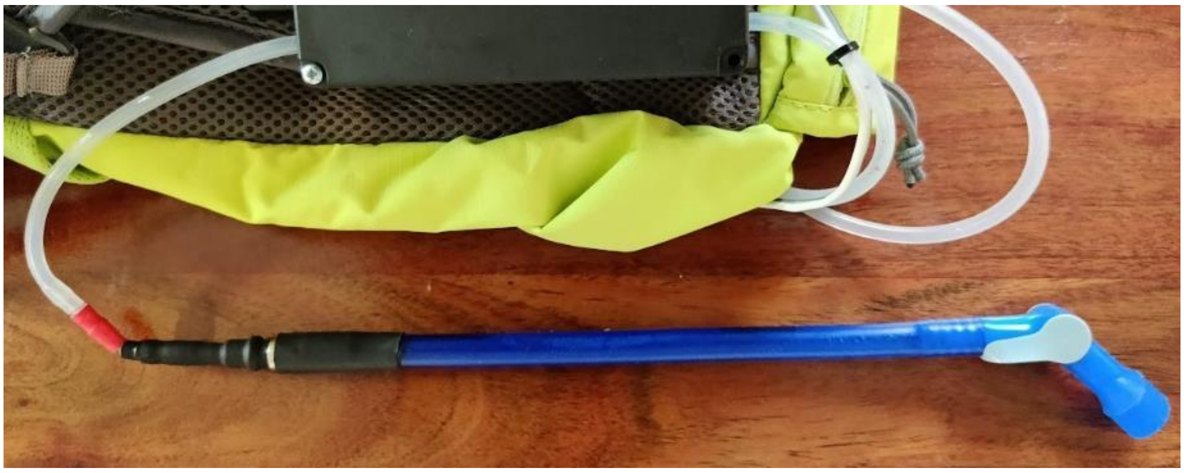


Figure 22: Completed modification of the drinking tube.

In Figure 22, we can see the finished modification of the drinking tube. The blue tube is part of the original drinking tube, and behind the black and red heat shrink are the metal tube adapter and narrower silicon tubing. The tube could have been much shorter, but by using a longer piece of tube the user can “suck out” the liquid from the tube, giving a better drinking experience. The end of the drinking tube has a bite valve attached. The user must bite on this mouthpiece to allow the juice to exit the tube and to be able to drink the juice.

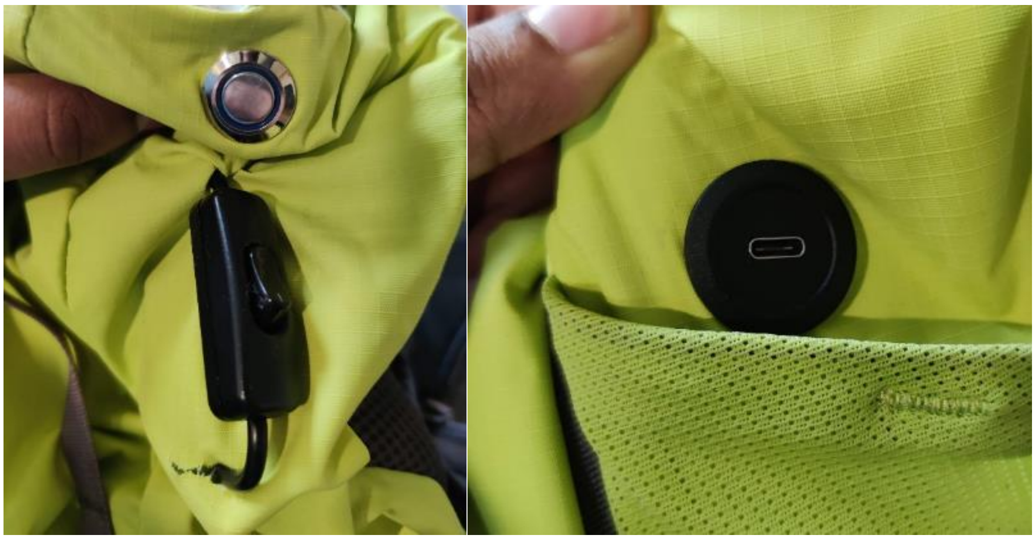


Figure 23: Pushbutton, power switch and USB-C charging port mounted to the backpack.

On the right side of the backpack is a momentary pushbutton, power switch, and a USB-C charging port, as seen in Figure 23. The push button is the only manual user input to the entire system. Based on the current action the system is doing, the button may either start the system, turn off the system, cancel a calibration, or cancel an ongoing dose. The button is mounted with a nut behind the backpack's fabric, giving a clean look. The USB-C charging port is mounted in the same way. The port serves as a method of charging the power bank inside the backpack without the need to remove it. The power switch switches off power from the power bank to the Raspberry Pi and motor controller and is the hard reboot switch for the entire system.

6.1.2 3D Printed Parts

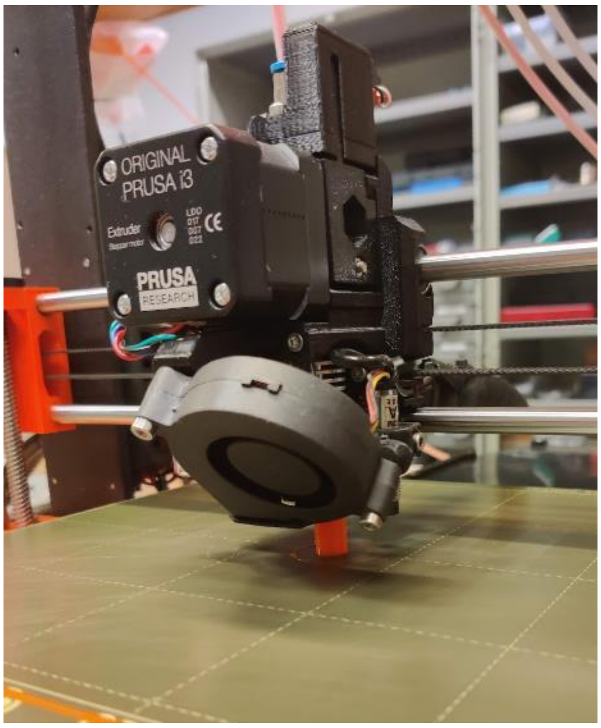


Figure 24: Printing a part on the PRUSA I3 MK3 3D Printer.

During the implementation stage, it became apparent that additional parts were needed. Some of the parts required were very specific, needed to be customized, only orderable in high quantities, or were not available in the IFI Workshop. However, a PRUSA I3 MK3 3D Printer was available for use in the IFI Workshop, visible in Figure 24. The parts printed for this system are all available from Thingiverse.[6]

Although print times can be long, 3D printing proved to be a quick, cheap, and easy way to create custom parts that were not commercially available. Three parts in the M-CDS system are 3D printed: the Raspberry Pi Case, an M16x1 nut, and standoffs for the Raspberry Pi HATs.

[6] https://www.thingiverse.com/

6.1.2.1 RPi Case

Raspberry Pi cases are available on the internet in various designs and sizes. The main issue with these cases is the limitation of customization and modification. The system uses two RPi HATs – the Perma-proto board and the motor controller board – which means the case needs to be tall enough and not have internal bracings in the way of the HATs.

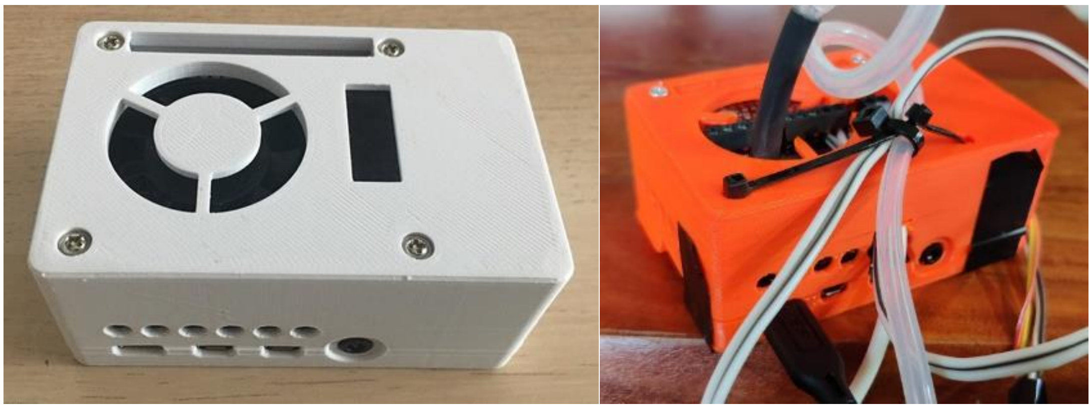


Figure 25: Raspberry Pi 4 Case for RGB Cooling Hat[7] by timmygu, licensed under CC BY 4.0, alongside modified version used in this system.

A case design was found on Thingiverse[7] and printed to solve this issue, as seen in Figure 25. This design was meant to be used for a specific HAT, the RGB Cooling HAT, and therefore requires some modification to fit our HATs and Raspberry Pi in the case.

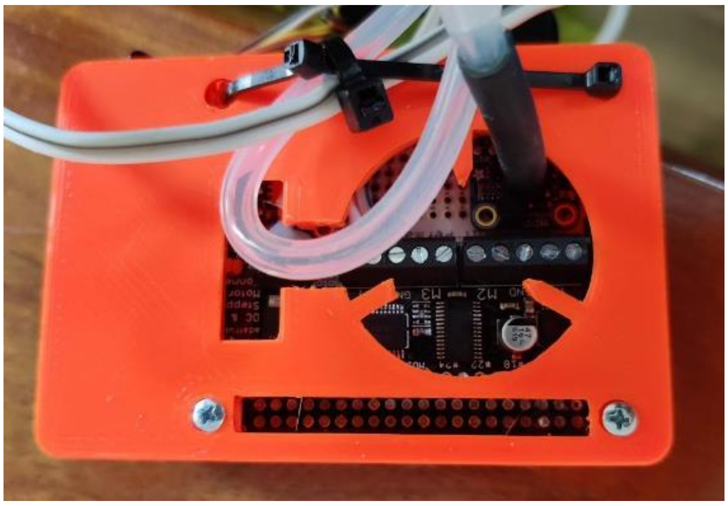


Figure 26: Modification of the fan vent.

[7] https://www.thingiverse.com/thing:4598338

The fan vent on the top was cut out to allow a silicon tube to pass through to the MRPLS pressure sensor and make space for the motor controller's terminal blocks, as seen in Figure 26. The tube may have fit without cutting out the vent, but this gave the tube more leeway and made removing the case easier.

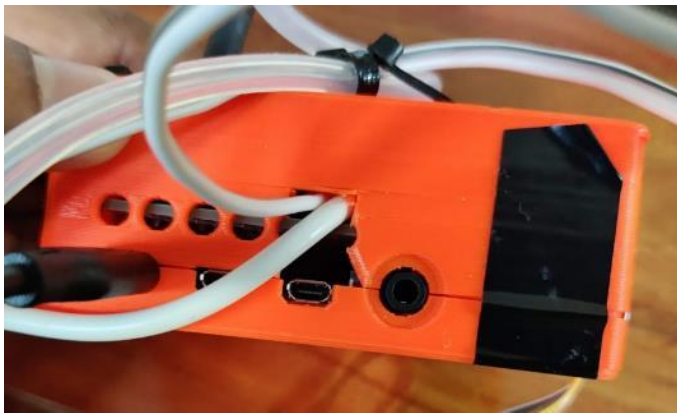


Figure 27: Modification of the side vent holes.

The side of the case had vent holes to allow air circulation inside. These holes were expanded to allow cabling for the motor controller and power pass-through, as visible in Figure 27. Supports by the USB ports were also cut to pass cables for the piezo buzzer and button.

The inside of the case had various supports to ensure a tight fit for the RGB cooling HAT it was designed for. However, with the two HATs we installed, the case did not fit properly. Many of these internal supports were trimmed away to ensure that the case fit, but this did not weaken the case's structural integrity.

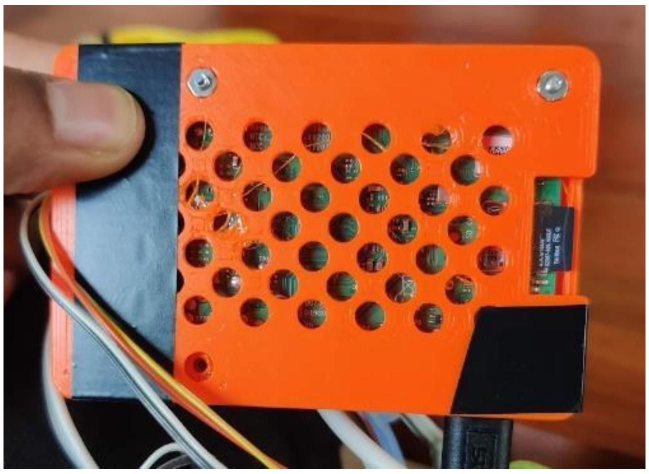


Figure 28: Holding the case together with tape and screws.

Two machine screws were used to secure the Raspberry Pi to the bottom part of the case, but due to the HATs' design and the lack of sufficient parts, the case had to be taped together to ensure it would not come apart, see Figure 28. The case is designed so that it provides a tight fit around the perimeter but does not ensure that the case locks into place. Using tape also allowed for easier debugging in case the internals needed access.

6.1.2.2 M16x1 Nut

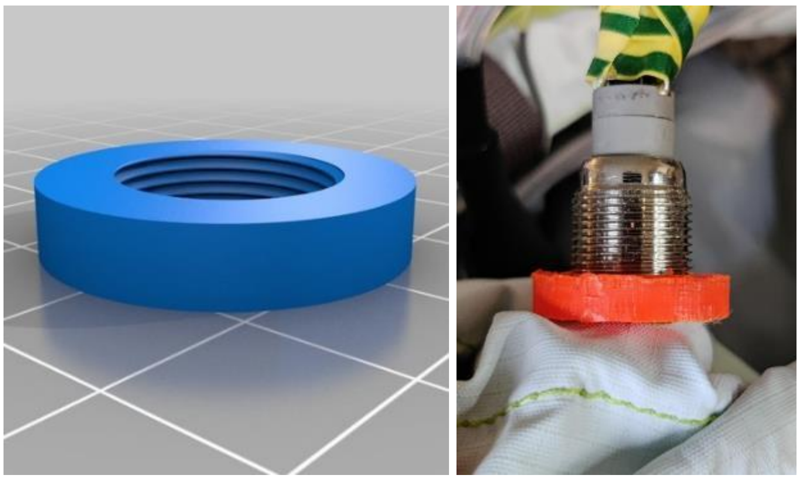


Figure 29: M16x1 GX16 Aviation Connector by jbkuma[8], licensed under CC BY-NC-SA 4.0; the nut attaching the pushbutton to the backpack.

Although the system is autonomous, a single button outside the CamelBak allows user input in some specific situations (as explained in section 6.1.4.7). This pushbutton was found in the IFI Workshop but was missing the nut to attach it in place securely. The size of the threads on button were measured to be M16x1, an uncommon thread pitch for such a large thread diameter. Buying a replacement M16x1 nut would be difficult and not worth shipping such a small item. By 3D printing a new nut, seen in Figure 29, which was done relatively quickly, the button could be used without ordering a new metal nut.

[8] https://www.thingiverse.com/thing:3399570

6.1.2.3 Pi HAT standoffs

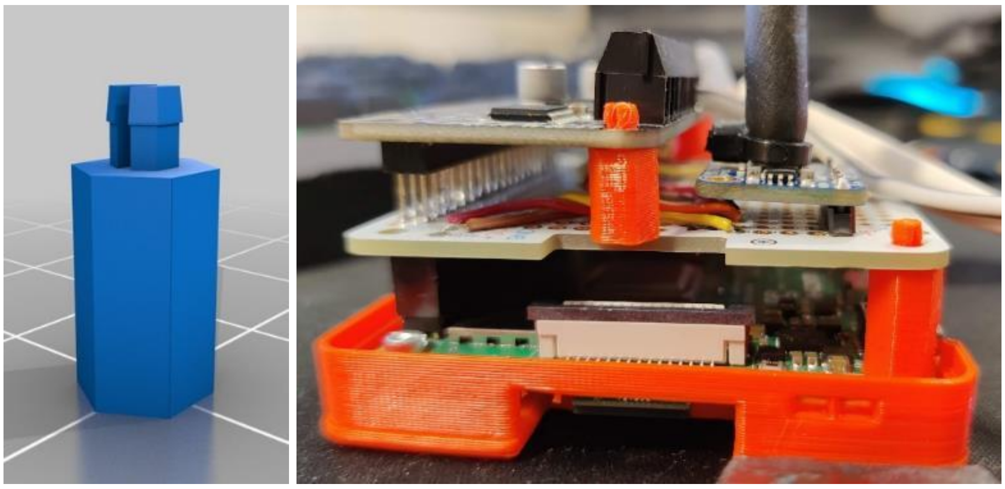


Figure 30: Raspberry Pi Hat Standoff by makenai[9], licensed under CC BY 4.0; printed standoffs on HATs.

Each HAT mounted to the Raspberry Pi is supported by the header pins they are connected to. On the other side, however, the HATs float in midair. Although the header pins are strong enough to hold them, they can flex slightly if pressure is exerted. To avoid the HATs from flexing and to give them a tighter fit, standoffs that could be snapped into the mounting holes were printed, as seen in Figure 30. This gave a simple way of ensuring that the HATs could not flex downwards. Flexing in the opposite direction was not an issue, as the 3D-printed RPi enclosure held the HATs in place.

6.1.3 Raspberry Pi

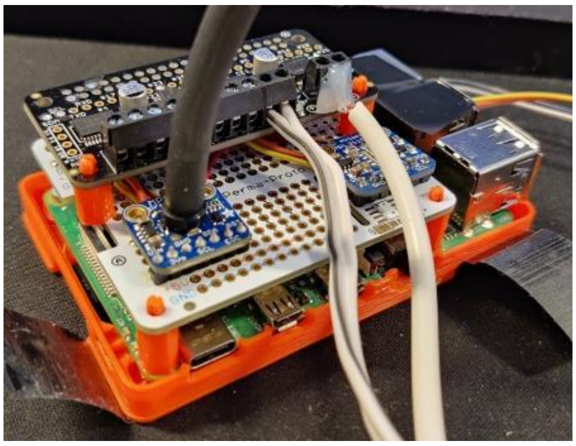


Figure 31: Raspberry Pi 4B with sensors, HATs and various cabling.

[9] https://www.thingiverse.com/thing:1149645

A Raspberry Pi 4B single-board computer is the heart of the M-CDS backpack. The pushbutton, piezo speaker, pressure sensors, and motor controller are directly connected to the RPi using the Perma-Proto board or directly via the RPi's header pins, as seen in Figure 31. The sensors and motor controller communicate with the RPi via the I_2C protocol. Note that the Raspberry Pi itself is unmodified and can be replaced in case of failure; each HAT is removable and can be replaced as well.

The Raspberry Pi runs Raspberry Pi OS, a Debian Linux-based distro, and has three scripts that are necessary for the operation of the M-CDS system:

- **startup.py**: a script that runs on boot for initializing the Raspberry Pi,
- **watchdog.c**: a program that continuously runs the main script, restarting it if necessary,
- **mcds.py**: the main software script that fetches BG data and runs the main algorithm.

The main software script is described in sections 6.1.5 and 6.1.6.

6.1.3.1 Start-up script

After the Raspberry Pi has booted, a script (startup.py) is immediately started via the bash script "rc.local" executed at the end of the multi-user mode after network services have started and the command line is available to the user. The bash script also logs all output to a log file for debugging and historical purposes.

The start-up script initializes the network and waits for the user to be ready to use the system. Firstly, the script kills any existing instances of wpa_supplicant – a Unix program used for connecting to WPA-encrypted Wi-Fi networks – to avoid conflicting connections. The script then checks for a valid ethernet connection by pinging a known IP address with high uptime (8.8.8.8, Google's public DNS server). This check allows a quicker boot time, as an ethernet connection was used during debugging and the programming phase to quickly make changes to the system. If the ping fails after three attempts, the system attempts to connect to a Wi-Fi network from a list of known networks and passwords. The network it connects to is a mobile hotspot started by the user's mobile phone. It attempts each Wi-Fi network four times before failing.

If no network connection is made, the Raspberry Pi will buzz the piezo buzzer with a failure tone, similar to the sound of an ambulance. This indicates to the user that a fatal error has occurred and that they must reboot the system.

If the network connection is successful, the system will make a continuous beeping sound through the piezo buzzer, indicating that the system is ready to start. The user may then press the pushbutton on the outside of the backpack to start the watchdog, which will play a short melody and start the main system.

6.1.3.2 Watchdog

After the user has pressed the pushbutton, the watchdog (watchdog.c) starts execution. This program continuously runs in the background and does not end until the Raspberry Pi is shut down. It is called a watchdog based on the nature of a watchdog timer, a hardware device that can automatically detect errors in software and reset the system if they occur. [46] The watchdog starts the main M-CDS software script and waits for it to exit. When the M-CDS script exits due to a fatal error or user intervention, the watchdog starts listening for a pushbutton press. When the pushbutton is then pressed, the watchdog starts the M-CDS script again. The watchdog uses the WiringPi C library to access the GPIO pins.

This allows the Python script to fail gracefully and restart in the case of an error and ensures that the system does not need user interaction through a terminal or graphical interface to start the program again. Since the system is designed with autonomy in mind, this is an important detail.

6.1.4 Raspberry Pi Hardware and parts

Several hardware devices and parts must be connected to the Raspberry Pi for the M-CDS system to function. A graphical overview of the hardware configuration is depicted in Figure 32.

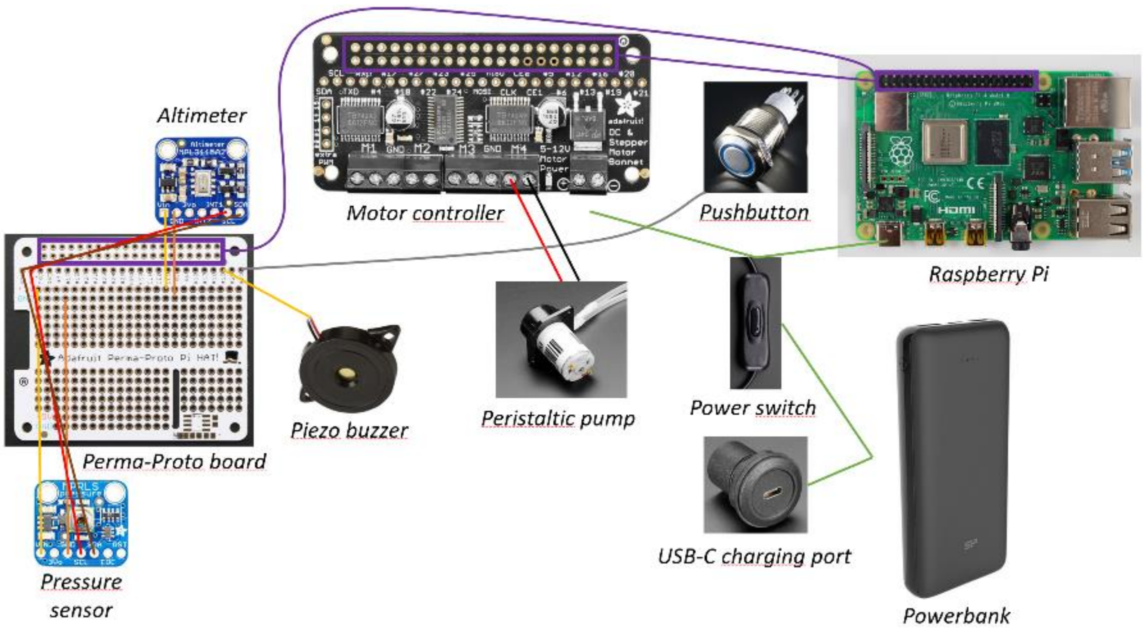


Figure 32: An overview of the hardware and how various parts are connected.

The entire system is powered by a USB power bank, which can be recharged through an external USB-C port on the backpack's exterior. A power switch, also positioned outside the backpack, controls the power supply for the motor controller and the Raspberry Pi. The Raspberry Pi is interfaced with all other necessary hardware components through the two HATs: the motor controller and the Perma-Proto board. The Perma-Proto board serves as a hub by connecting various devices; the piezo buzzer, pushbutton, pressure sensor, and altimeter are connected. Furthermore, the motor controller controls the peristaltic pump, which pumps the juice from the reservoir to the drinking tube.

6.1.4.1 Motor Controller

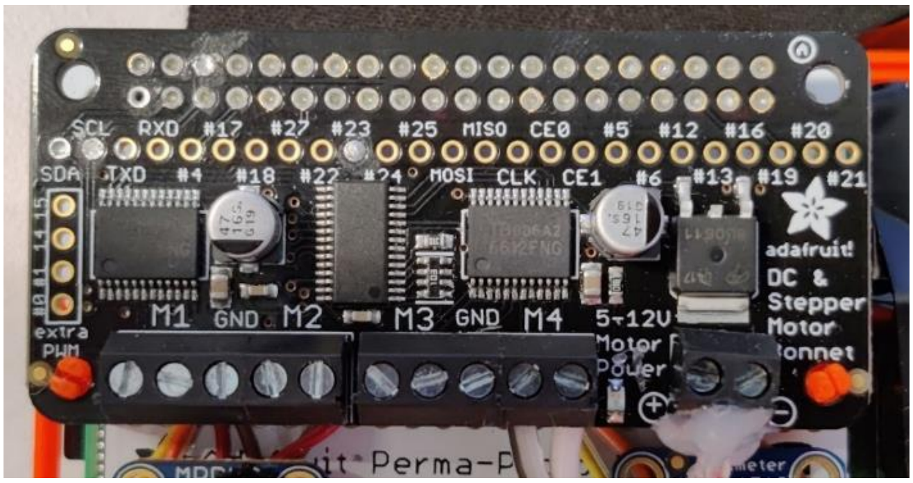


Figure 33: The Adafruit DC & Stepper Motor Bonnet

This project uses the Adafruit DC & Stepper Motor Bonnet[10] to power the motor for the peristaltic pump, as shown in Figure 33 above. This motor controller can run up to four motors simultaneously and uses a dedicated power supply to supply power to the motors. It communicates with the RPi through I_2C and has its own PWM driver chip to control motor direction and speed, freeing up the use of the RPi's already few PWM pins. The motor controller has existing libraries for use in Python, allowing for quick and simple programming. Running the motor forward pumps juice from the reservoir to the drinking tube, and running it in reverse pumps excess juice back into the reservoir after a session to clean the system.

This HAT is relatively overpowered for the use case of this project. Although only a single motor channel is being used, connecting multiple motors in case of a different design, such as the one mentioned in section 8.7.1, is possible. The controller also supports motors with voltages up to 12V, but to keep the project's simplicity and run everything from a 5V USB power bank, a 5V peristaltic pump was used.

The altimeter used in this project also communicates with the RPi through I_2C and has a fixed address of 0x60 that cannot be modified simply. The motor controller also has a starting address of 0x60 which crashed with the altimeter. However, the motor controller has PCB pads that can be soldered together to change the address of the I_2C device. The device address was therefore changed to channel 0x61, as visible in Figure 34.

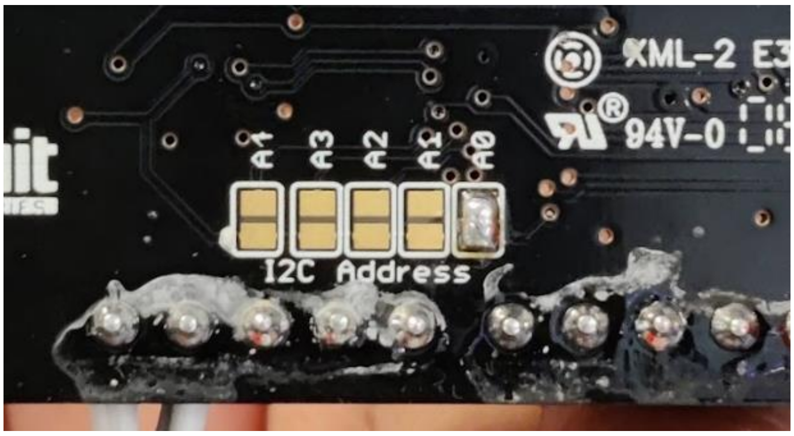


Figure 34: The I_2C addressing PCB pads on the rear side of the motor controller.

[10] https://www.adafruit.com/product/4280

6.1.4.2 MRPLS pressure sensor

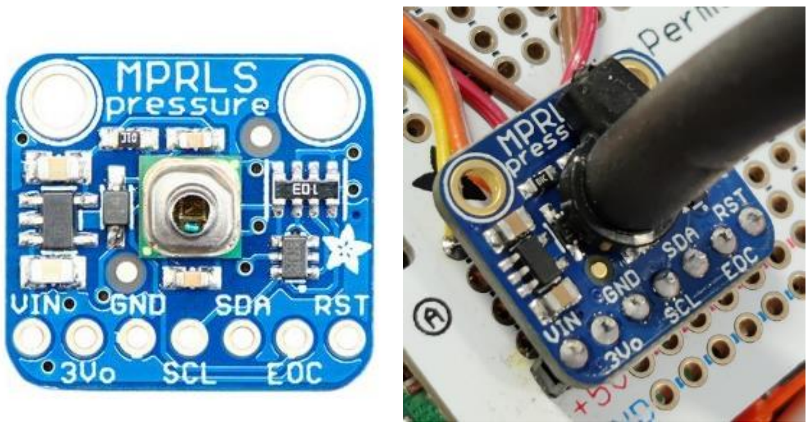


Figure 35: Adafruit MPRLS Ported Pressure Sensor[11]

The Adafruit MRPLS Ported Pressure Sensor[11] is a barometer pressure sensor with a metal port on top of the sensor chip. This allows attaching a tube and measuring pressure in a closed environment, as seen in Figure 35.

The drinking tube has a bite valve on the end of it to stop the juice from overflowing or pouring out unexpectedly. However, during calibration and dosing, pressure will build up inside the system if the user does not bite on the valve, which could lead to tubes loosening and juice spraying everywhere. The bite valve can only hold a certain amount of pressure and could slightly open up and spray juice with high pressure. These scenarios could lead to the electronics of the system being damaged or the user being covered in juice.

[11] https://www.adafruit.com/product/3965

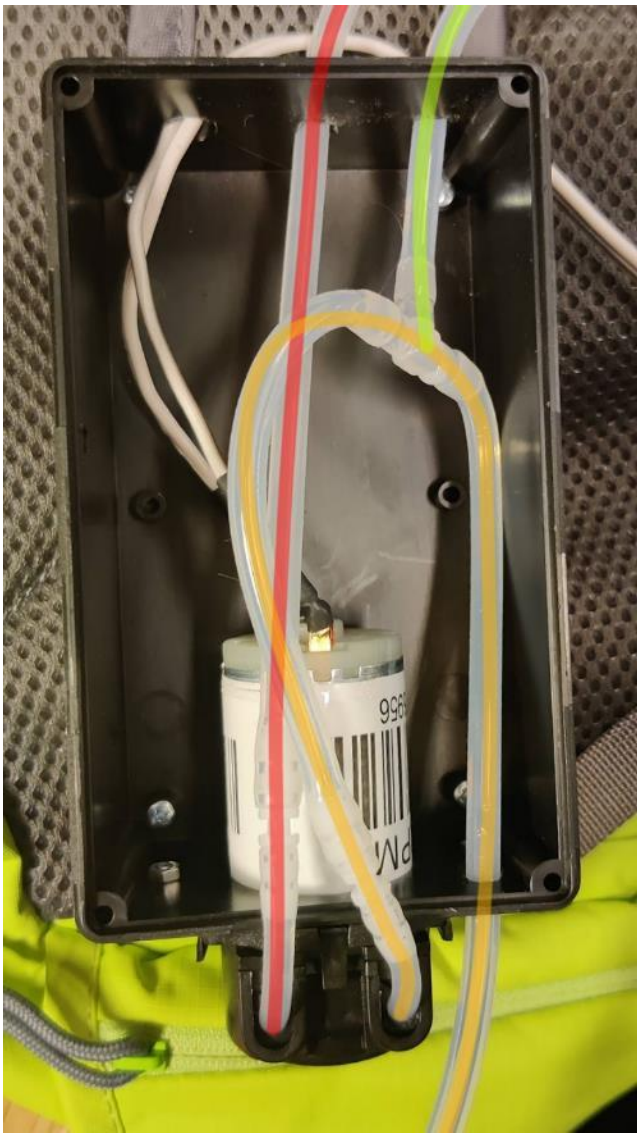


Figure 36: The pump tubing assembly with juice flow directions

The MRPLS pressure sensor is attached to the tubing section after the peristaltic pump to circumvent this issue. In Figure 36, we can see the internal structure of the tubing for the pump enclosure. The red line indicates the juice flowing into the pump, and the orange line is the juice flowing out of the pump and into the drinking tube. The green line is connected to the tube after the pump and leads directly to the MRPLS pressure sensor. This allows us to measure the pressure in the drinking tube.

If the user is not drinking from the tube while the pump is running, the pressure will increase. By attaching the sensor to this part of the tube, we can detect this change in pressure and stop the pump. When the user bites on the bite valve and begins drinking again, the pressure will decrease and will again be detected by the sensor. By measuring the sensor readings during specific events, we can set a minimum and maximum reading for the sensor to ensure that the pressure never exceeds the set limit. A detailed diagram of the system tubing is described in section 6.1.4.8.

The sensor communicates with the Raspberry Pi through I_2C on channel 0x13.

6.1.4.3 Barometer/altitude sensor

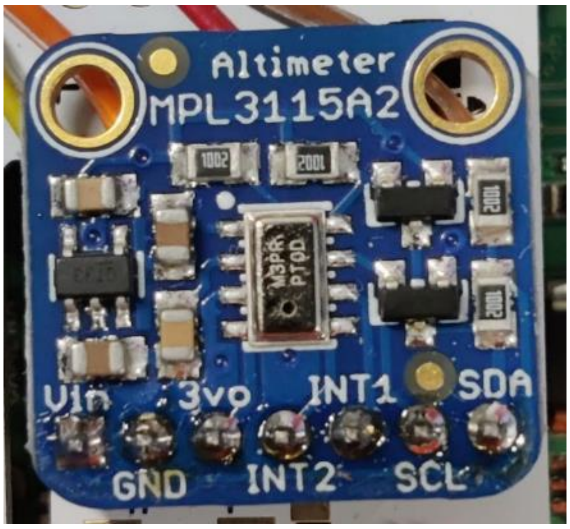


Figure 37: MPL3115A2 I2C Barometric Pressure/Altitude Sensor

The MPL3115A2 I2C barometric pressure sensor[12] is a chip that measures barometric pressure, altitude, and temperature, although only barometric pressure is used in this project. The sensor does the same task as the previous pressure sensor, with the key difference being that this sensor does not have a port, as visible in Figure 37.

The MRPLS sensor reads pressure relative to the current atmospheric pressure. Since atmospheric changes as the sensor’s height above sea level changes, for example, if the user takes the backpack on a mountain trip, the sensor's reading would be different. The standard pressure at sea level is 1013.25 hPa, but at a height of 500m, the sensor would read 995 hPa. The atmospheric pressure also changes based on the temperature and humidity.

[12] https://www.adafruit.com/product/1893

We use this second pressure sensor to calibrate the MRPLS sensor to counteract this change. By doing so, we will always be at an approximate zero value when the pressure inside the tube is the same as the pressure in the atmosphere at the current location. This gives us a relative reading of the pressure inside the tube compared to the atmospheric pressure.

The sensor communicates with the Raspberry Pi through I$_2$C on fixed channel 0x60. As this crashed with the motor controller, the motor controller's channel was changed to 0x61.

6.1.4.4 Adafruit Perma-Proto Board

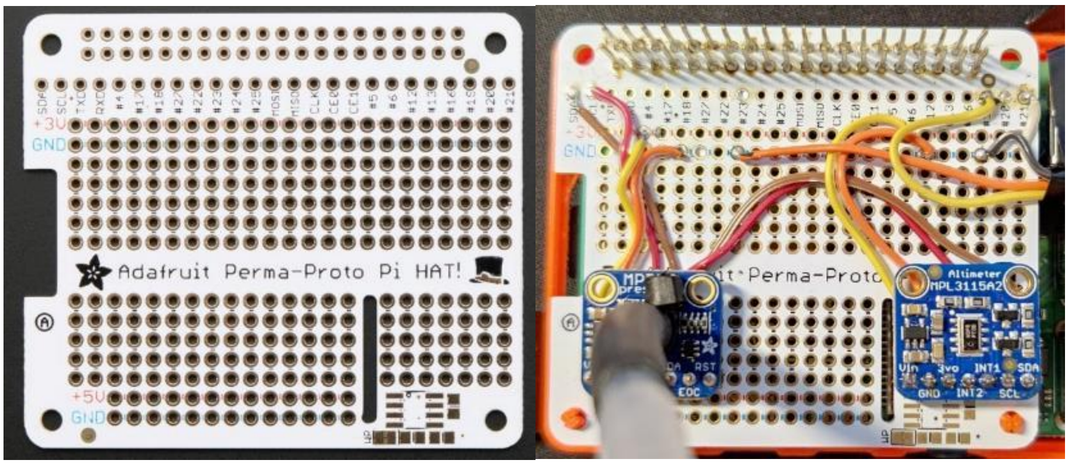


Figure 38: Adafruit Perma-Proto HAT for Pi[13], before and after adding components

When connecting the sensors and various other parts, it was essential to avoid soldering or making modifications directly to the Raspberry Pi. If the Raspberry Pi were to fail, replacing it would require unsoldering all connections and soldering them to the new Raspberry Pi as well as any other modifications.

Previously, the motor controller was soldered directly to the components, which you can see some remnants of in the top left corner in Figure 33 on page 45. However, this was difficult to do and destructive to the integrity of the motor controller. Cables were soldered directly to the pads rather than in through-hole slots.

By using the Adafruit Perma-Proto board, the pressure sensors, buzzer, and pushbutton could be connected to the RPi in a non-destructive, secure, and permanent manner, as seen in the right

[13] https://www.adafruit.com/product/2310

side of Figure 38. The Perma-Proto is a breadboard-like PCB that allows for “permanent prototyping”. Almost all the Raspberry Pi’s GPIO pins are exposed to separate PCB pads, and the bottom section of the board mimics a standard breadboard.

Figure 38 shows how the various components were connected to the Perma-Proto board. In the top left corner, we can see the cables for the I$_2$C communication protocol, on the pins marked SCL and SDA, connected to each of the pressure sensors. Each sensor also retrieves ground and 3.3V from the Perma-Proto board. On the top right side, we can see the cables for the push button on GPIO pin #21 and the piezo buzzer on the PWM-compatible GPIO pin #19.

6.1.4.5 Peristaltic pump

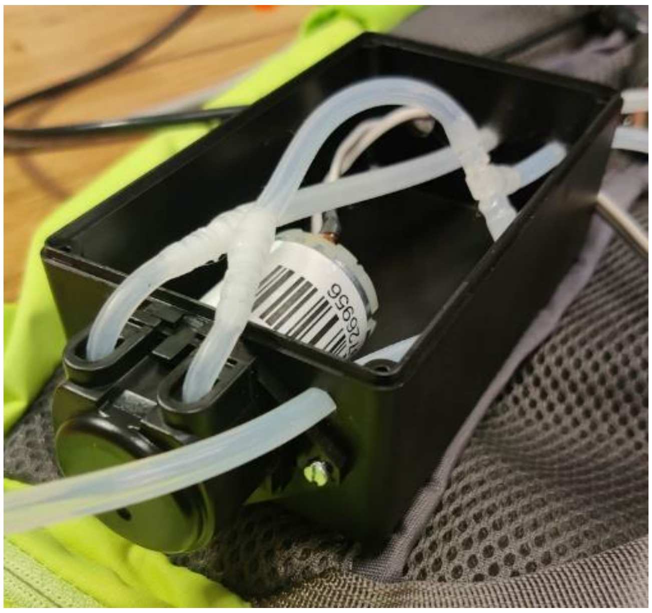


Figure 39: The peristaltic pump[14] in its enclosure

A pump of some kind is required to transfer the juice from the reservoir into the drinking tube. The pump needed to be small enough to fit in the backpack and have a low voltage and power consumption to avoid requiring a large battery. It also needed to avoid touching the liquid directly, such as a centrifugal or gear pump, to prevent juice contamination. A peristaltic pump meets all these criteria.

[14] https://www.adafruit.com/product/3910

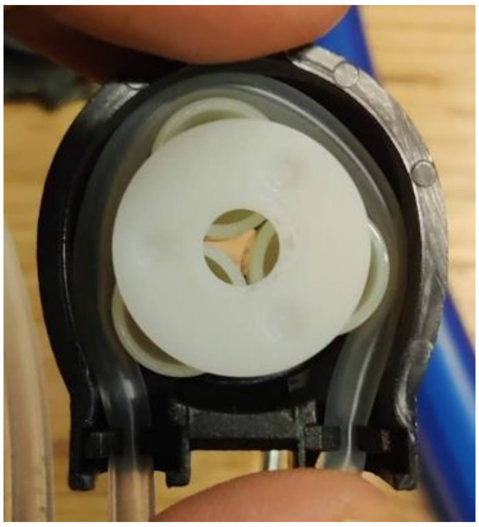


Figure 40: The pump with the plastic head removed. The three rollers are visible and are driven from the motor through a rod that goes through the center of the rollers.

In Figure 40, the pump section behind the plastic covering can be seen. A peristaltic pump is a positive displacement pump that moves liquid by rotating rollers across silicone tubing, compressing the tube and forcing the liquid to move forward through the tube. The liquid is untouched by the pump, keeping it away from contaminating the pump components [47]. Figure 41 shows a frame-by-frame animation of how fluid is passed through the peristaltic pump.

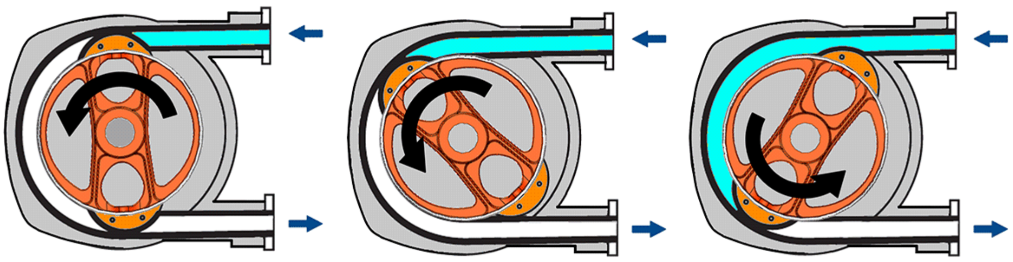


Figure 41: Peristaltic pump mechanism. As the roller turns, the liquid is forced through the tube. [48]

The motor controller drives the pump, allowing juice to be pumped forward or in reverse. According to the official documentation, the pump is rated for 5V and 6V[15], but running the motor at 5V gives a lower throughput. However, 5V is a simpler voltage to use, as it is common in power banks and microcontrollers. Due to the lower throughput, the entire tubing for the system is made as short as possible to decrease waiting time during the calibration phase.

[15] https://www.adafruit.com/product/3910

6.1.4.6 Power bank

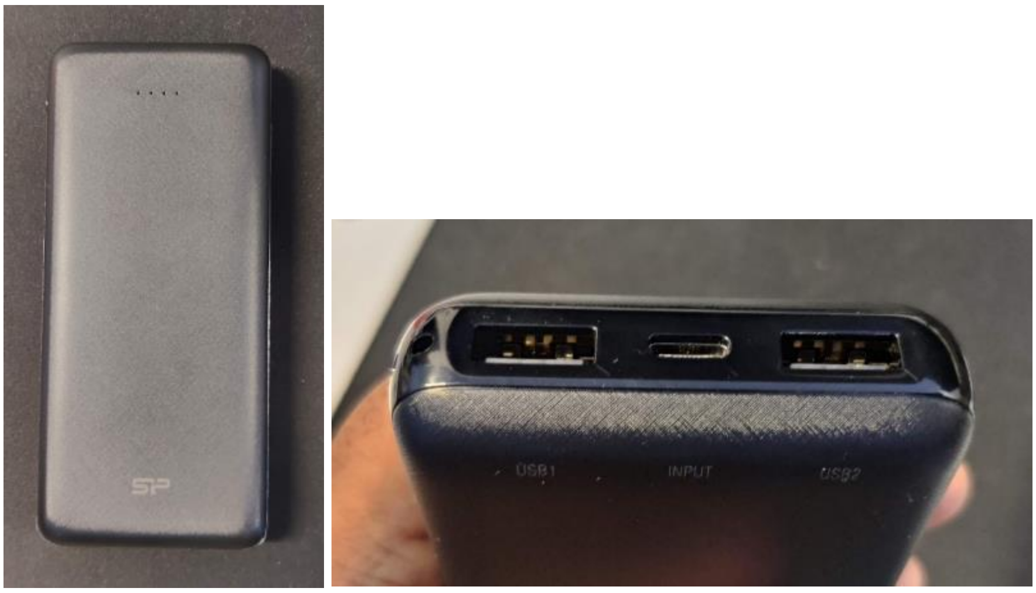


Figure 42: Generic 20000 mAh power bank

As a mobile system, multiple devices require power in one way or another, thus requiring a battery. Initially, the project required a 12V motor, which was fine in a stationary device, but a mobile device would need a specialized battery and charging circuit to be safe and practical. When the 5V motor was chosen for the project, a power bank could be used instead, as seen in Figure 42.

USB power banks have built-in charging, overload, and safety circuitry, alleviating the need for a custom design. A 20000 mAh power bank was chosen for the project, which could last a Raspberry Pi between 19 – 20 hours of intensive use,[16] which is more than what the RPi in this project will use on average. 2A of power can be drawn from the power bank, which is more than enough to power the Raspberry Pi and a single peristaltic pump. The power bank is easy to connect to the system using USB cables and can be recharged by the user with any USB-C charger.

[16] 1010 mA when running “ab -n 100 -c 10”, see https://www.pidramble.com/wiki/benchmarks/power-consumption

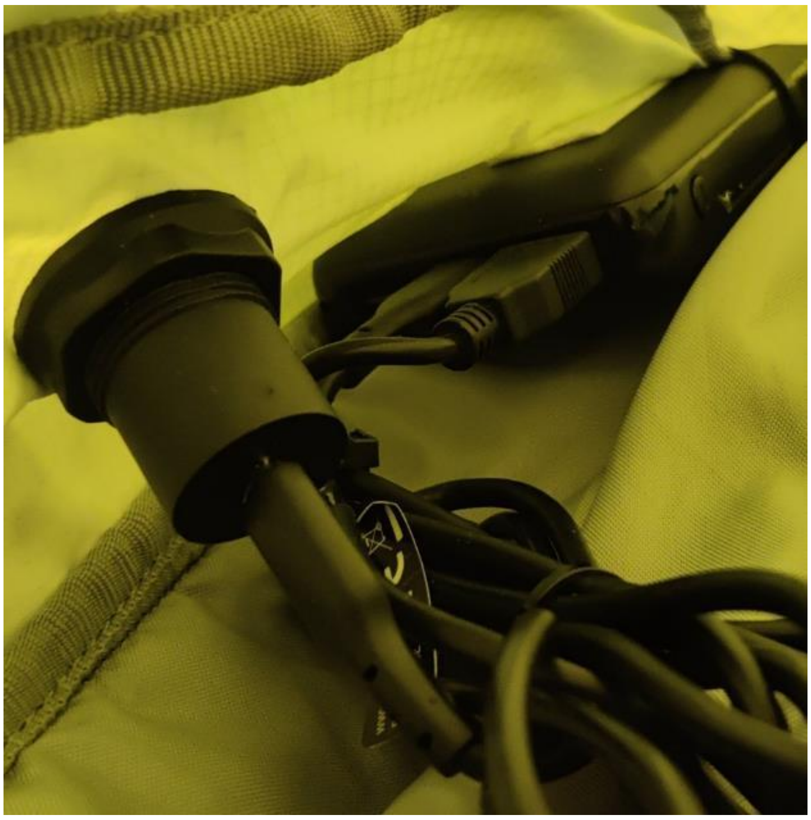


Figure 43: The powerbank and charging port mounted inside the backpack

The power bank is mounted securely inside the backpack with double-sided mounting tape and zip-tied, as seen in Figure 43. A USB-C charging port is mounted on the outside of the backpack and connected directly to the power bank's input. A cable from the USB port goes to the power switch on the outside of the backpack and is then split to the motor controller and Raspberry Pi.

6.1.4.7 Various electronic parts

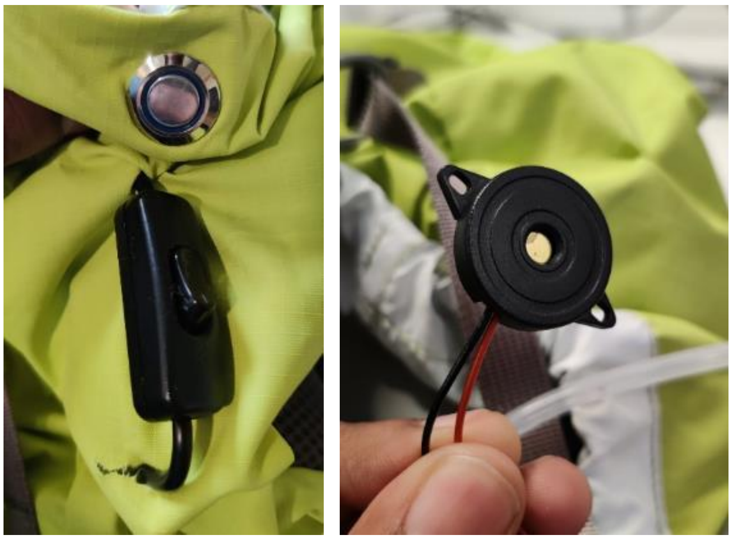


Figure 44: The pushbutton, power switch, and piezo buzzer.

The backpack has a couple of other parts required for operation, a power switch, a pushbutton, and a piezo buzzer, as seen in Figure 44.

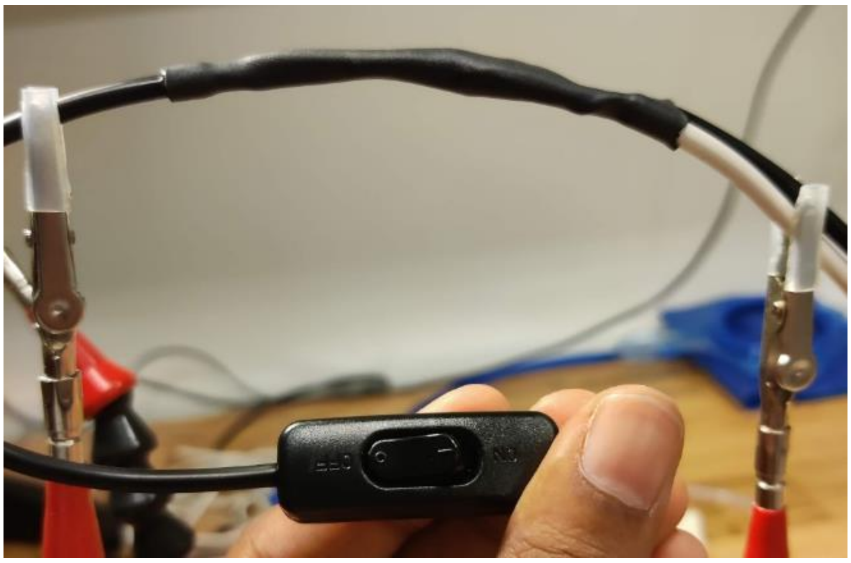


Figure 45: Modified power switch cable, connected to the RPi and motor controller.

The power switch is connected to a single USB port on the powerbank and to the RPi and the motor controller, as each requires separate power inputs. The motor controller has a separate power input for motors requiring a higher voltage than the Raspberry Pi can deliver. As seen in Figure 45, the power switch was modified to connect to both the Raspberry Pi's USB-C port and the motor controller’s terminal block. The cables were soldered together and then covered in heat shrink to avoid shorting them.

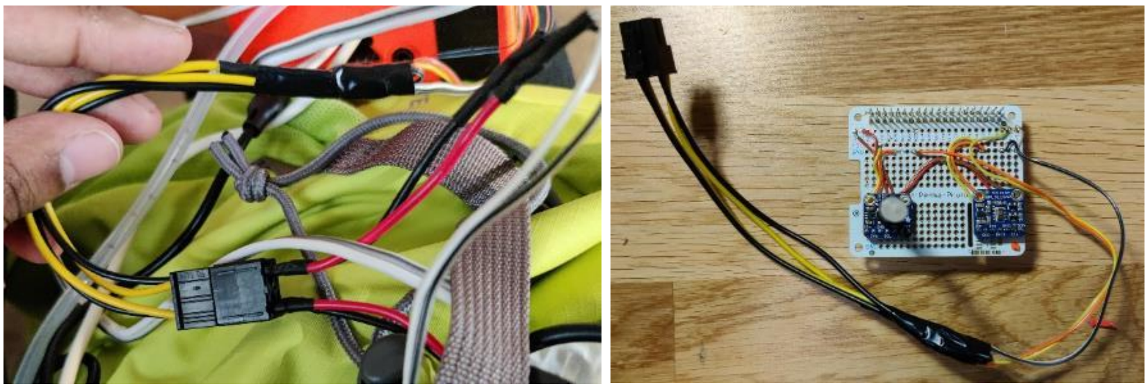


Figure 46: The Molex connector going to the pushbutton and piezo speaker.

The pushbutton and piezo buzzer have short cables connected to them. To make debugging easier, a Molex connector was used to uncomplicate disconnecting the cables, see Figure 46. This allowed servicing the Perma-proto board and the other components attached to it much easier, as seen in the right image of Figure 46.

Although the M-CDS backpack runs autonomously and requires little user feedback, the pushbutton is used for a few cases requiring user input. The pushbutton attached to the outside of the backpack is used for four different functions.

Firstly, after the system has booted up and connected to the internet, the piezo speaker will continuously beep in short intervals to indicate that the system is ready. The user must then press the button to start the system.

Secondly, during calibration, pressing the button will stop the calibration process. If the user restarts the system after a dose or initial calibration, there is no need to calibrate the system again. If the calibration took less time than expected, the user can press the button again to stop the calibration.

Finally, after the calibration process, the button can be used to shut down the system gracefully at any time. If the button is pressed during a dose, the system will cancel the dose and shut down.

The piezo buzzer is the only source of feedback to the user on what the system is currently doing. Different sounds are associated with different events in the system. An overview of the various events and sounds can be seen in Table 4.

Table 4: Overview of the different buzzer events and sounds

Event	**Sound**	**Length**
Boot finished, waiting for user	Constant beep with short intervals	Until user presses the pushbutton
Event finished (button press after boot, calibration complete, dose complete)	Uplifting "fanfare", musical notes expressing a positive experience	2 seconds
System shutdown	Opposite notes of "Event finished"	2 seconds
Calibration or dose ongoing	Short "beep" every second	Until event is finished
Pressure too high in system	Different "beep" sequence than calibration/dose	Until user lowers pressure in system, by biting or sucking on the bite valve
Fatal error	"Fail" sound effect	3 seconds

6.1.4.8 Silicone tube and tubing adapters

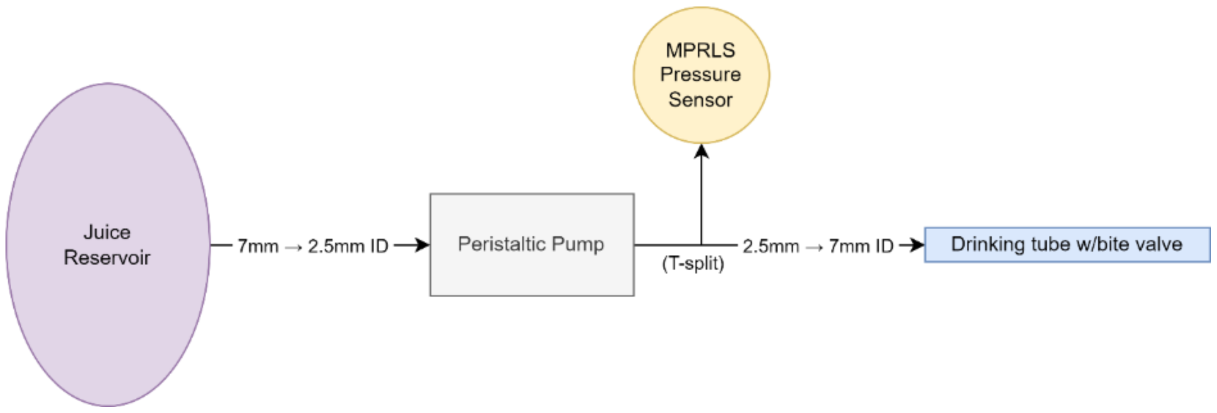


Figure 47: A diagram of the tubing connections

The most essential function of the M-CDS backpack is pumping juice safely from a reservoir into the drinking tube. A couple of challenges had to be solved to use the system properly.

Silicone tubing is used throughout the system as the peristaltic pump uses it, and the food-grade qualities of silicone tubing are required for this project. Silicone tubing is widely available in various sizes, so that was not an issue for this project. The tubing connection diagram can be seen in Figure 47.

However, the different sizes in the silicone tubing posed another issue; how can tubes of different sizes be connected? The CamelBak system uses tubes with a 7mm ID, whereas the peristaltic pump and MPRLS sensor use tubes with a 2.5mm ID. Although some adapters are available for purchase online, they are usually not food grade or must be ordered from online marketplaces such as eBay or AliExpress from individual sellers rather than a reputable source. These websites also typically have long shipping times, which would slow down the development process.

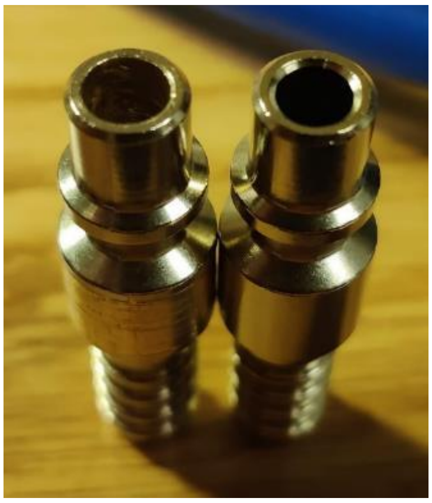


Figure 48: Metal hose adapters, modified adapter on the left.

Metal hose adapters, see Figure 48, were bought to extend the CamelBak tubes but were instead used as makeshift adapters. The ID of the metal adapters was almost 4.5mm – the OD of the peristaltic pump silicone tubes. The inside of the adapters was drilled out to fit the silicone tubes, as seen in Figure 48, which allowed the tubing to fit inside perfectly. This was done at the adaption from 7mm to 2.5mm ID and from 2.5mm to 7mm ID. As the system had to withstand high pressure and be completely watertight, a heat shrink was wrapped around the adapter and tubing to ensure no liquid would escape. The completed adapters wrapped in heat shrink can be seen in Figure 49.

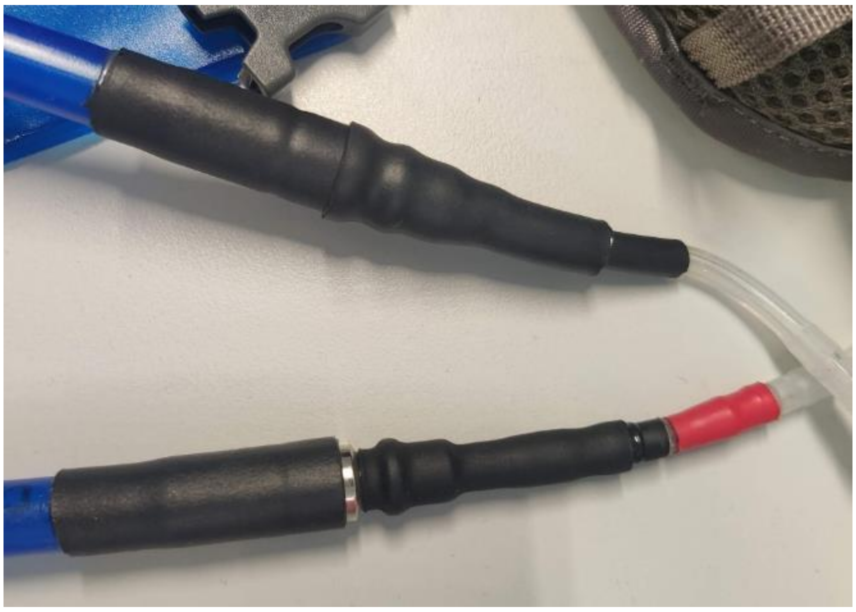


Figure 49: The heat-shrink wrapped adapters. Top: from reservoir to pump, bottom: from pump to drinking tube

The pump assembly, as seen in section 6.1.4.2, leads to a 2.5mm ID T-split between the drinking tube and the MRPLS pressure sensor. The pressure sensor is heat-shrink wrapped and carefully tightened with a zip-tie to ensure the tube does not disconnect.

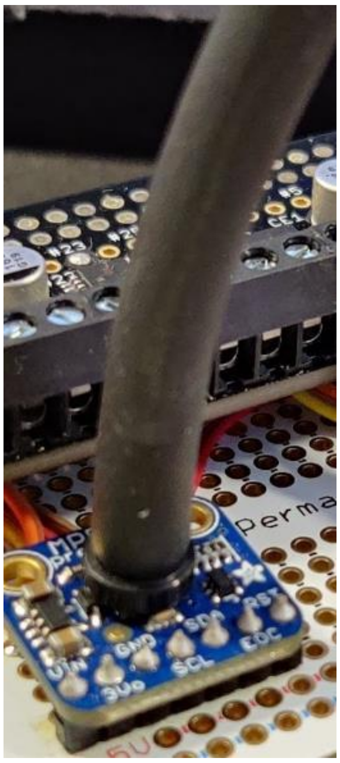


Figure 50: The MRPLS sensor with the tubing securely attached.

After the peristaltic pump, the only place where the pressure in the system can be relieved is at the bite valve. If the sensor tubing were to detach, the liquid would follow the path of least resistance and pour out over the entire hardware setup. Therefore, a watertight connection that could withstand high pressure was very important.

6.1.5 M-CDS Software

The M-CDS software script is the heart of the entire system. The program is explained in the following segments in the following sections: Initialization, BG data retrieval, and the dosing system.

As the system runs headless without any graphical interface or user interaction, it is crucial that the system only ends if a fatal error occurs. In addition to most exceptions being handled in code, an error handler will catch the exception and exit gracefully in the case of unhandled exceptions. This ensures that all threads are ended and that the pump does not run indefinitely. The program can then be restarted by pressing the pushbutton on the backpack.

6.1.5.1 Initialization

After the watchdog (see section 6.1.3.2) starts the main program, it initializes the database, hardware, and various threads.

The local SQLite database is initialized if it does not exist. This database contains the BG data from the current session and when doses were given to the user. Then, all necessary hardware is initialized, such as the GPIO pins for the button and buzzer, the motor controller, and two pressure sensors. GPIO pins are accessed using the RPi.GPIO library in Python and the motor controllers and pressure sensors have publicly available Python libraries.

The script runs a total of four threads for various tasks. Two of the threads are dedicated to updating the readings from the pressure sensors. The ambient pressure sensor has a slow read time of up to one second, which, if running on the main thread, would halt the execution of the entire script. Running this update on a separate thread ensures the main thread is not blocked often. The MPRLS pressure sensor is also updated on a separate thread, as this sensor needs to be updated as quickly as possible. Another thread handles the buzzer melodies, as they are blocking with sleep() function calls to create the melodies, and the last thread runs pump dosing code. It starts at initialization, but only begins once a dosing event begins by using Python's threading.Event().

6.1.5.2 Calibration process

Once everything is initialized, the system then begins the calibration process. When the user first uses the backpack, all the tubes will be clear of any juice and empty. If the user were to receive a dose, the juice would have to travel throughout the entire system before the user could drink anything. This process was measured to take approximately 30 seconds after three tests, which means the user would have to wait 30 seconds before getting the initial dose of juice.

To circumvent this, we "calibrate" the system by pumping juice for at least 30 seconds. The user must bite on the bite valve to release pressure within the system and allow juice to flow freely through the drinking tube. If they stop biting, the pressure sensor will detect an increase in pressure, and the user will hear a warning sound from the piezo buzzer. When the user bites the valve again, the pressure will decrease, and the calibration will continue.

After the calibration process, the system reads and retrieves the user's BG data every minute.

6.1.5.3 BGL data retrieval

In the initial design in section 5.1**Feil! Fant ikke referansekilden.** mentions a mobile device for retrieving data from Nightscout and forwarding it to the Raspberry Pi. Due to time constraints and to simplify the development process, the mobile application was removed from the implementation and instead replaced with a mobile hotspot that the Raspberry Pi could connect to directly. This simplified how the Raspberry Pi managed data from Nightscout and was significantly faster to implement.

The M-CDS script fetches data from Nightscout every minute. The Freestyle Libre 3 CGM sends data every minute, and most other CGMs send data every five minutes, allowing us to get the data as soon as possible from the user. [45] When a new, unique data point is retrieved from Nightscout, it is stored in the local SQLite database. The system then checks if a dose is currently running, in which case we do not need to check for a hypoglycaemic event as we are already giving the user a treatment. If we are not dosing, we check for a hypoglycaemic event.

The system's standard thresholds are 106 mg/dl for a low BGL and 90 mg/dl for a dangerously low BGL. If the user's BGL is below the low threshold and the Nightscout trend line shows a decline or lower, we give a dose to the user. If the user's BGL is below the dangerously low threshold and the Nightscout trend line is flat or lower, we also give a dose to the user. If their BGL is low, but the Nightscout trend line displays an up arrow, we can assume that the user's BGL will increase shortly.

When the conditions are met for a dose, the database is updated to indicate that a dose was given on this BG reading, and the dosing thread begins.

6.1.6 M-CDS Dosing system

6.1.6.1 Calculating dose length and amount

Multiple calculations and measurements had to be done on the system to determine how much juice to dose.

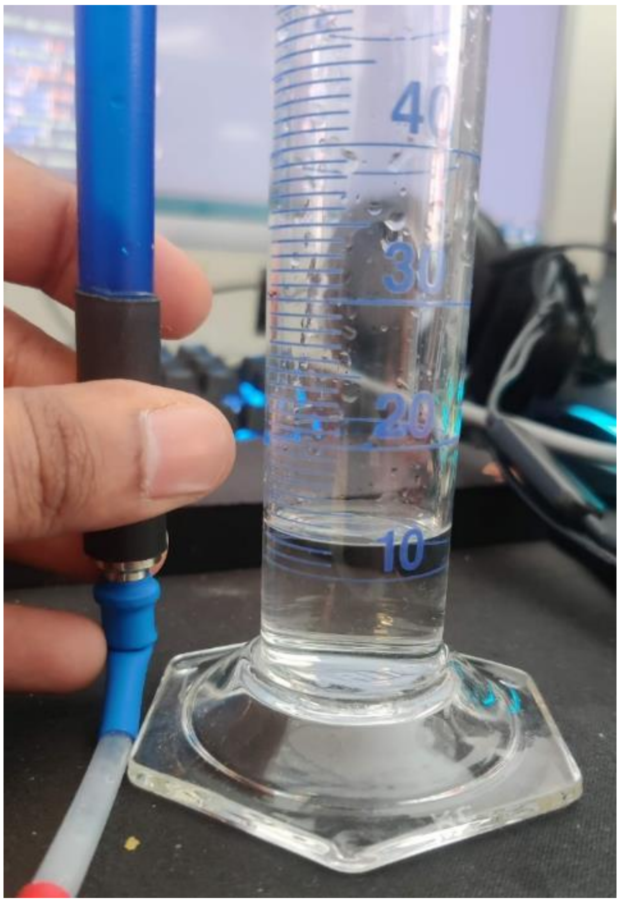
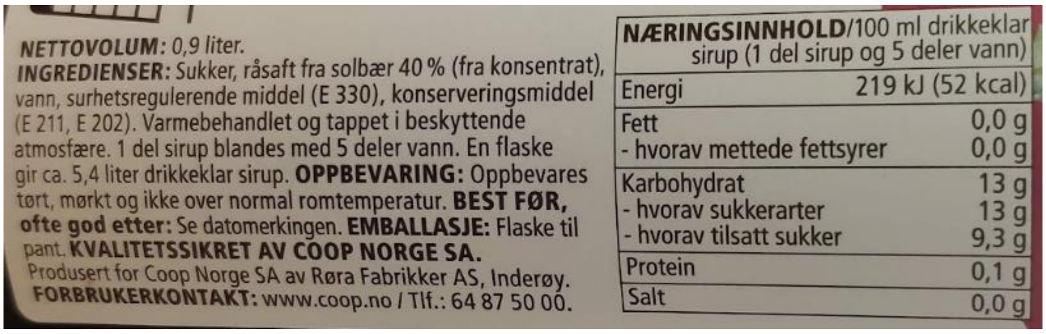


Figure 51: Measuring the tube inaccuracy.

Firstly, the drinking tube is long enough to contain a significant amount of juice. Since the user can suck the liquid out of the drinking tube, even when the pump is not operating, it is important to calculate the inaccuracy of each dose. The liquid measured in the tube was approximately 12.5 ml, as seen in Figure 51.

The throughput of the pump also had to be measured to be able to calculate how much liquid the user would be able to drink per second. After three tests, the average throughput per minute was approximately 41 ml/min ≈ 0.683 ml/s. This is much lower than the 100 ml/min figure the manufacturer gave,[17] but it is not a problem for our system. It is easier for the user to sip the liquid while in physical activity rather than drinking continuously.

[17] https://www.adafruit.com/product/3910



NETTOVOLUM: 0,9 liter.
INGREDIENSER: Sukker, råsaft fra solbær 40 % (fra konsentrat), vann, surhetsregulerende middel (E 330), konserveringsmiddel (E 211, E 202). Varmebehandlet og tappet i beskyttende atmosfære. 1 del sirup blandes med 5 deler vann. En flaske gir ca. 5,4 liter drikkeklar sirup. **OPPBEVARING:** Oppbevares tørt, mørkt og ikke over normal romtemperatur. **BEST FØR, ofte god etter:** Se datomerkingen. **EMBALLASJE:** Flaske til pant. **KVALITETSSIKRET AV COOP NORGE SA.**
Produsert for Coop Norge SA av Røra Fabrikker AS, Inderøy.
FORBRUKERKONTAKT: www.coop.no / Tlf.: 64 87 50 00.

NÆRINGSINNHOLD/100 ml drikkeklar sirup (1 del sirup og 5 deler vann)	
Energi	219 kJ (52 kcal)
Fett	0,0 g
- hvorav mettede fettsyrer	0,0 g
Karbohydrat	13 g
- hvorav sukkerarter	13 g
- hvorav tilsatt sukker	9,3 g
Protein	0,1 g
Salt	0,0 g

Figure 52: The black currant juice nutritional information

The juice used for the system was black currant juice. This specific brand of black currant juice (Coop Solbærsirup) contained the most carbohydrates per 1 dl of mixed juice; 13 grams of carbohydrates, see Figure 52. This meant the user did not need to drink as much juice as other brands. Since the throughput of the pump is relatively low, any time saved per dose is important. It was also important to mix the juice at the recommended ratio to keep the flavor consistent, ensuring that the user would enjoy drinking the juice rather than a highly concentrated mix.

The M-CDS software assumes a dose of 15 grams of carbohydrates for the user. This means that we need $\frac{15}{13} \approx 1.153\, dl$ of juice for 15g of carbohydrates. With the error of 12.5 ml – equivalent to 1.625g of carbohydrates – each dose can be calculated to be approximately 15.8125g $\pm$ 0.8125g. Figure 53 shows the calculation for the total dose length.

$$\frac{115.3g}{0.683\, ml/s * 60} \approx 2.8135\, min \approx 2\, min\ 48.8\, seconds = \mathbf{168.8}\ \boldsymbol{seconds}$$

Figure 53: Calculating dose length

6.1.6.2 Giving user a dose

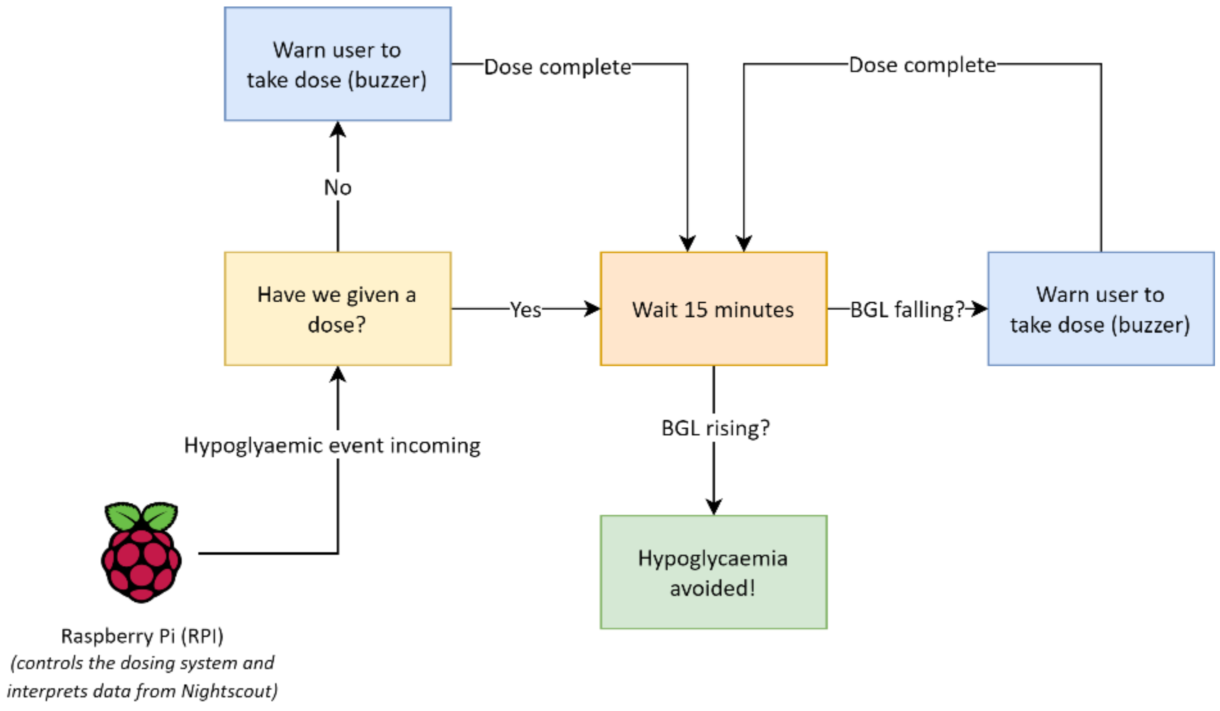

Warn user to take dose (buzzer)
Dose complete
Dose complete
No
Have we given a dose?
Yes
Wait 15 minutes
BGL falling?
Warn user to take dose (buzzer)
BGL rising?
Hypoglyaemic event incoming
Hypoglycaemia avoided!
Raspberry Pi (RPI)
(controls the dosing system and interprets data from Nightscout)

Figure 54: Flowchart showing how the dosing process works.

When a dose is required due to oncoming hypoglycaemia, a dose is given based on the 15/15 method, which is the same algorithm House of Carbs uses: intake 15 grams of carbohydrates, wait 15 minutes, and repeat if necessary. [7] A detailed flowchart showing the dosing process is seen in Figure 54.

The dosing system works very similarly to the calibration phase. The peristaltic pump will continuously pump for the required time (168.8 seconds) while the buzzer beeps every second to indicate that a dose is ongoing. If the user stops drinking, pressure will build up in the system, and the pump will stop. The buzzer will make a new sound, indicating that the pressure must be released before continuing. In contrast to the calibration process, the user must suck on the tube and create “negative” pressure in the system to continue the dose. This ensures that the user is still drinking rather than just releasing the pressure in the system.

When the dose is finished, the pump will stop, and the piezo buzzer will play a “finished” melody. The dosing code now waits 15 minutes, and updates from Nightscout are saved to the database, but no action is taken. If the user’s BGL is still not above the threshold and climbing after 15 minutes, a new dose is given to the user. If the BGL is above the threshold and climbing, the hypoglycaemic event has been avoided.

6.2 Backend server

Any user with their own Nightscout instance may use their own server with the M-CDS backpack, but during user testing, the student’s physical server running the Nightscout instance and the LibreLinkUp uploader script inside a Docker Compose instance was used.

The LibreLinkUp uploader script is required to get data from the Freestyle Libre sensors into Nightscout, as Nightscout does not have native plugin support for the Abbott CGMs. LibreLinkUp is a program meant for diabetes patient’s close contacts, such as family members or parents, allowing them to see the patient’s BGL on their phone in real-time. Although this program has a closed API, the developer behind the uploader script has reverse-engineered the LibreLinkUp API, allowing us to parse the data from LibreLinkUp, and then upload it directly to a Nightscout Server. The patient must first invite our account as a “follower” of their BG data, allowing us to access it through LibreLinkUp [49].

If multiple M-CDS backpacks needed to be deployed simultaneously, the same server could be used by running multiple Docker instances and proxying them through a http server with proxying capabilities, such as ngnix or Apache2. The Docker Compose file is set up to allow this quick expansion of the system. A single server can then be used for multiple instances of Nightscout, as it does not use many resources.

7 Evaluation and Results

7.1 Literature Review

The literature search ended with 11 relevant papers using AI, ML, deep learning or neural networks to predict future BGLs in T1D patients. Below is a table showing all the studies and their prediction algorithm's accuracy. The system's accuracy is measured using the root mean square error (RMSE) and is sorted by prediction horizons (PH) of 15 min, 30 min and 60 min. Some studies contained other PHs, but these three were the most prevalent. All RMSE measurements are in mmol/L. Papers that did not test their algorithms in this way were not included.

Table 5: Overview of the test results from the chosen papers.

Paper	**Year**	**Method**	**Input data**	**15 min PH**	**30 min PH**	**60 min PH**
[50]	2019	Clu-RNN (recurrent neural network)	BG	N/A	0.322	0.746
[51]	2019	RNN	BG	N/A	0.647	N/A
[52]	2020	Autoregression with Exogenous inputs (ARX) neural network (NN)	BG, IOB, COB, Heart rate	N/A	1.081	N/A
[53]	2021	I3-GWO-KELM, ensemble learning model	BG	0.353	0.918	N/A
[54]	2021	Artificial NN regression	BG	0.505	N/A	N/A
[55]	2021	Long short-term memory (LSTM) based deep RNN	BG	N/A	0.358	0.957
[56]	2021	Fine-tuned convolutional neural network (CNN)	BG	N/A	0.988	1.560
[57]	2021	TCN based predictive model	BG	N/A	1.289	N/A
[58]	2021	Edge-LSTM model, deep learning	BG	N/A	1.060	1.777
[59]	2022	Random forest (RF) regression model	BG	0.863	1.532	N/A
[60]	2023	Fast-adaptive and Confident Neural Network (FCNN)	BG	N/A	1.035	1.724

Most of these papers use the OhioT1DM dataset to test their algorithms, primarily using only historical CGM data as training data. The 30 min PH RMSE ranges from 0.332 mmol/L to 1.289 mmol/L. Some papers had open-source code available, with most programmed in Python using TensorFlow.

7.2 Diabetes Association Research Forum

Unfortunately, the panel gave no new feedback or insights on the system. Some in the panel instead mentioned that they would most likely not use the system as they already plan their physical activity well enough.

However, this negative feedback was in fact helpful in designing the system. An important part of the backpack was to make it as simple and autonomous as possible. The less the patient had to do, the more likely they would use the system. This was considered during the design and implementation process, resulting in a system requiring minimal user interaction.

7.3 Interview

As mentioned in section 3.4, the mother and her son (here referred to as subjects A and B) were interviewed based on the interview guide in that section.

Question #1: How do you manage your diabetes on a daily basis? Equipment, etc.

Subjects A and B use the Dexcom G6 CGM and the Omnipod Dash 780G insulin pump. They use the Loop iPhone application with the iAPS algorithm to run their insulin pump. Subject A uses their Fitbit to see their current basal insulin rate and BGL, but manually inputs their carbs and meals through Loop. Subject B uses an Apple Watch to see the same statistics but has a simple interface to choose bolus insulin doses for specific meals quickly. This makes it quick and easy for subject B to set the pump to give an insulin dose.

Both subjects have glucose meters to take finger tests when necessary.

Question #2: Do you manage your diabetes in the same way during physical activity?

They both usually change their pump to give a lower basal insulin rate and smaller bolus insulin if they were to be physically active after eating a meal.

Question #3: Compared to normal activity, how does your blood sugar fluctuate during physical activity?

They usually do not experience hyperglycaemia while exercising but have experienced hypoglycaemia due to exercise or miscalculating how much insulin they needed before physical activity.

Question #4: Do you compensate for fluctuating blood sugar while exercising by eating or drinking?

Subject B takes dextrose tablets to increase their BGL which are easy to chew. He also takes the tablets at night if needed. Subject A also mentions drinking juice or eating a banana while exercising.

A follow-up question was asked; do you use the 15/15 rule for delivering carbohydrates during physical activities? Subject A answered that the 15/15 rule is generally outdated and was more common in the past when we did not have any much insight on diabetes. The 15/15 rule is too high for children and adults; it should instead be based on their body weight. The amount of carbohydrates they would dose also depends on their IOB, but they said a general rule would be 8 grams of carbohydrates at the start of a jogging trip, and they would take a new dose when the delta between two CGM readings was -1. In general, they would always intake some form of carbohydrates before exercising.

The system detailed in this thesis was then explained to the subjects, and they were then asked:

Question #5: Would you use this system? Why or why not?

Subject A initially said that they would like to try out the system. A smaller version for children would have to be made for subject B. He also mentioned that his backpack has a rain cover that the system would have to avoid puncturing to function.

Subject A was also concerned about the amount of space it used. She would rather have a mini version that could be placed inside the backpack or on the side of the backpack in a bottle holder. She would also like to see notifications on your phone for a dose in case you were to miss the buzzer making a noise or was unsure of what the system was currently doing. Integration with an Apple Watch or Fitbit would also be interesting, as they both use smartwatches.

7.4 System

The system implemented in this master thesis adheres to the proposed design, albeit with some simplifications to keep the project scope manageable and to account for time constraints. It is important to acknowledge that this project is a proof of concept and should not be considered a refined product.

7.5 Prototype Costs

Table 6: Overview of the costs of the prototype parts

Part	**Price**	**Num.**	**Total Price**
Raspberry Pi 4B	kr 999,00	1	kr 999,00
20000 mAh Powerbank	kr 466,25	1	kr 466,25
DC Motor Controller	kr 337,00	1	kr 337,00
CamelBak Arete 18 with drinking reservoir	kr 299,00	1	kr 299,00
Peristaltic Pump	kr 260,20	1	kr 260,20
House coupling 8mm ID, 10pk	kr 239,56	1	kr 239,56
MPRLS Pressure Sensor	kr 200,08	1	kr 200,08
Micro-USB to USB-C cable	kr 199,90	1	kr 199,90
MPL3115A2 Pressure Sensor	kr 167,50	1	kr 167,50
Silicone Tube, 2.5 mm ID	kr 36,52	3	kr 109,56
USB-C to USB-C cable	kr 105,13	1	kr 105,13
Micro-USB to USB-A cable w/switch	kr 101,50	1	kr 101,50
Enclosure for pump	kr 69,90	1	kr 69,90
USB-C Port	kr 62,05	1	kr 62,05
T-connector, 2.5mm ID, 5pk	kr 46,95	1	kr 46,95
Plastic tube coupling, 2.5 ID, 5pk	kr 36,52	1	kr 36,52
Piezo buzzer	kr 13,13	1	kr 13,13
Total cost			**kr 3 713,23**

Table 6 displays an overview of the costs for each of the parts of the project. The total cost of the project was 3713,23 kr. Note that this does not include extra parts that were ordered in case of faulty parts, nor does it include parts already available in the IFI Workshop (pushbutton, Perma-Proto Board, screws, nuts), but only includes parts that needed to be ordered. The total cost of all the system parts would be slightly higher.

7.6 User Testing

Testing was conducted with an individual who has T1D. The participant was recruited from within the HIT research group and has experience in computer science and diabetes. They provided valuable insights regarding the usability and functionality of the system.

The test environment was set between 11:30 and 13:00 at the stone stairway Sherpatrappa in Tromsø. The participant was to turn on the backpack and go on a mountain hike for 1 – 2 hours, allowing their BGL to drop and seeing if the system would correct the BGL before a hypoglycaemic event. They used the Nightscout server described in section 6.2 and uploaded data to the server using a Freestyle Libre 3 CGM. The data was uploaded to Nightscout using the LibreLinkUp uploader script.

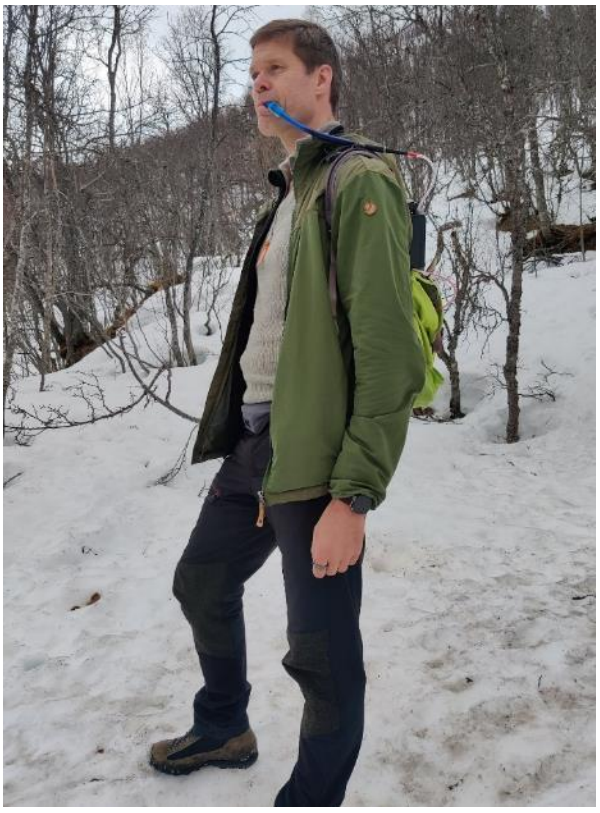


Figure 55: The participant receiving a dose of juice from the M-CDS system.

Unfortunately, the weather was suboptimal, and the snow had not melted from the stairway, as seen in Figure 55. This led to less poor walking conditions, and the hike up the mountain only lasted 30 minutes before the participant went back down. The rest of the trip was a walk around the area, up and down some hills, to see if the system could correct for the low BGL.

7.6.1 Results

Figure 56 shows a chart showing the change in the participant’s BGL during the testing period.

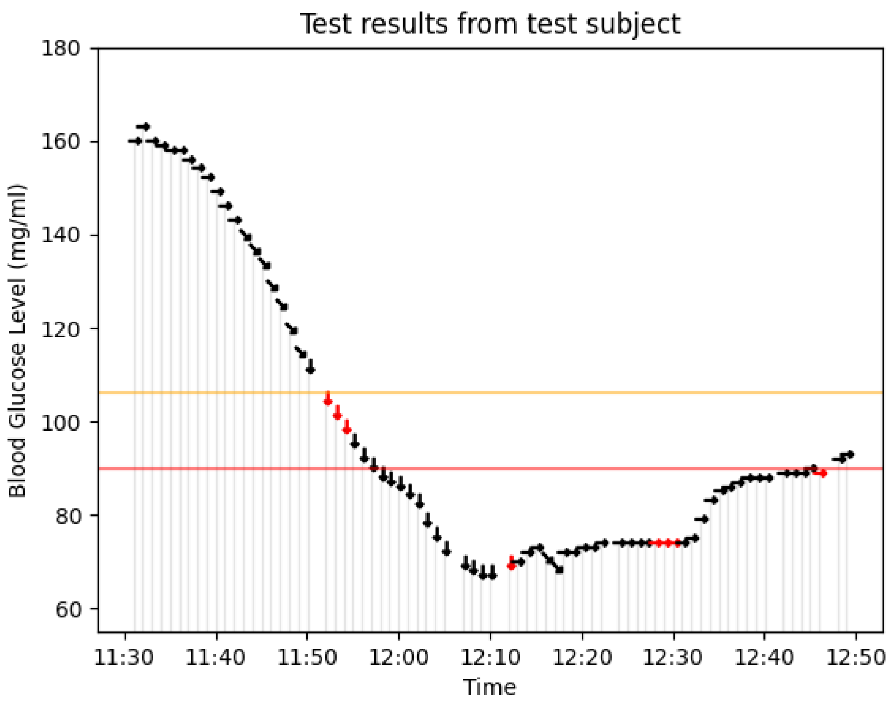


Figure 56: Test results from the test participant

The markers indicate the current trend line from Nightscout during that reading. The yellow horizontal line is the low threshold for a hypoglycaemic event oncoming (106 mg/ml), and the red horizontal line marks the dangerously low threshold (90 mg/ml). Markers in red indicate that a dose was given at this point. Three doses were given to the user during the trip at 11:52, 12:12, and 12:28. The dose at 12:46 was canceled prematurely by the participant, as they noticed their BGL was rising correctly after the previous three doses. Multiple red markers in succession are due to a bug in the code. If the user stops drinking during a dose, the pump will stop due to increased pressure, changing the program's internal state. This then incorrectly updates the entry in the database.

8 Discussion

8.1 Literature Review

The papers chosen in the literature review seem to predict the user’s future BG fairly well. Using AI or an ML algorithm could help predict BGLs in the M-CDS, allowing for even quicker hypoglycaemia detection. However, there are some issues with this approach.

Although the algorithms seem to work fine, the highest RMSE of 1.239 mmol/L shows that the prediction algorithm could potentially miss an oncoming hypoglycaemic event if the patient’s BGL was under four mmol/L. These algorithms also mainly work best for those with very stable BGLs. During physical activity, a patient’s BGL can fluctuate much more than a typical day while doing regular tasks or sitting in an office.

Many of these algorithms' source code is publicly available and open source, but they are not simple to integrate into the existing codebase. Training an ML model also requires significant amounts of training data, processing power, and time. In addition, due to a lack of knowledge in AI and ML algorithms, it would not be feasible to attempt to use these algorithms or create a new implementation within the timeframe and scope of the project.

8.2 Interview

The two subjects interviewed had good feedback for the system and gave an insight into the type of userbase this system is for.

Both subjects have a good overview of their diabetes and how they manage it in all situations. Most people do not need a system like the M-CDS backpack, as they feel they understand how diabetes affects their bodies better than a predictive system, which is usually true. This is a very important point and one of the main reasons the M-CDS system is designed to be as simple and user-friendly as possible. The less headache for the patient, the more likely they are to use it.

Subject A also mentioned that the 15/15 method is outdated, but as the interview was conducted after the implementation process was finished, no changes could be made. A less primitive algorithm to give doses would be better for the system.

The idea of a portable system is very interesting, as it would make the system even easier to use and support the concept of mass production. Although it was not possible to execute this idea during this thesis, it would be an interesting topic for future work.

8.3 Sikt

Unfortunately, the UiT DPO did not answer the first email, even after multiple attempts to contact them through email and phone. After an extended amount of time, an application to Sikt was sent in regardless of the DPO, as it was assumed that Sikt would have to be contacted anyways.

After applying to Sikt, an answer was given on May 16th. The advisor responsible for the application asked some questions about the M-CDS system and its effect on the users, data collection, and risks, and asked if we had considered this work "Experimental treatment with a different primary purpose other than providing health care to a single patient" in which case the project must be sent to REK, the "Regional committee for medical and health research ethics." We replied to all their questions and stated that this project would not need approval from REK due to the nature of the system not giving new knowledge about health or sicknesses, as it only utilizes existing knowledge about diabetes.

They replied on May 19th and again recommended that an application to REK be sent in, and if we decided not to apply, document that we had done an assessment concluding that the project was out of the scope of REK. We did not go forward with the application to REK, as the system was only tested on a single individual who is a member of the HIT research group.

Although applying for Sikt was a quick process, the Sikt application was sent in late in the project's timeframe. The average response time from Sikt is approximately one month, which was too late for the project. Due to this, the system was only tested with a single individual from the HIT research group rather than recruiting multiple users to test the system. Sikt also recommended that the project be sent to REK, which has a waiting time of two weeks. This again would be too late to do proper user testing.

Obtaining the necessary permissions to collect personal data is complicated due to these long waiting times. It would have been better to send in these applications much earlier in the process to ensure early approvals. Sending applications as soon as the data requirements were figured out would have been more effective than waiting until after the system was implemented.

UiT’s DPO was also a roadblock in the application process. The application to Sikt was delayed due to waiting for the DPO's response. The DPO was to provide clear guidance on the necessary applications to be filled.

8.4 Prototype Costs

Table 6**Feil! Fant ikke referansekilden.** on page 70 shows the most expensive parts of the system: the Raspberry Pi, power bank, motor controller, CamelBak, and peristaltic pump. These parts alone make up two-thirds of the total cost, which make them good points of discussion for lowering the cost of the prototype.

The Raspberry Pi is a very simple device to program and use and, before the recent chip shortage, was readily available for purchase. However, it is a power-hungry device compared to using a microcontroller and is much larger in size. Using a M5Stack[18] controller or a FiPy[19] would keep the cost down and allow for lower power consumption and a smaller battery, further reducing the cost. Both devices can be programmed in MicroPython, allowing the Python code on the Raspberry Pi to be reused, only requiring modification for standard Python libraries that aren’t available in MicroPython. The FiPy has a built-in LTE SIM card slot, alleviating the need for a Wi-Fi connection and saving the user’s mobile device’s battery. Unfortunately, the FiPy is no longer manufactured and had a listing price of ~900 kr. M5Stack has a variety of controllers ranging from 100 kr to 500 kr and different modules that “stack” onto the main controller. This would create a small device that could house the sensors and motor controller easily and cheaply.

A power bank was chosen for this project as it contained charging and protection circuits and worked well for a prototype. However, if the goal was to create a cheaper prototype, the power bank could have been replaced with Li-ion batteries, a charging circuit, and a voltage regulator with a boost converter. Although these components could be cheaper than a power bank

[18] https://m5stack.com/

[19] https://docs.pycom.io/datasheets/development/fipy/

separately, it would require more planning and testing rather than the simplicity of just connecting a USB cable to the power bank and everything working.

The motor controller is an essential part of the system. Without it, running the pump in different directions is impossible. Obtaining a cheaper motor controller with fewer channels may be possible, but not much can be saved here. The motor controller here will also allow for multiple pumps if future work requires it.

The CamelBak is a major part of the system; without it, there is no drinking reservoir or tube. There are other manufacturers of drinking reservoirs that could be used, but CamelBak is a popular and reputable brand, making it a clear choice for a prototype. Since the system is only available by using this backpack, it limits patients from using their own backpack, which is a desire from users that may not want to use this system, as mentioned in sections 7.3 and 8.2. Containing the system in a single enclosure that can be moved from backpack to backpack would not require the CamelBak at all.

Peristaltic pumps are generally expensive, as they are commonly used in medical cases due to the ability to measure the fluid passing through the pump accurately. The pump used in this project is meant for small hobby projects and is the cheapest of its kind. A pump of twice the capacity could easily cost 10x the price from a professional company. It is possible to buy generic pumps from online marketplaces, such as eBay and AliExpress, but the reliability and quality of these products are questionable.

8.5 Design and Implementation

8.5.1 3D Printing

While 3D printing was a great help throughout the project, it was not used to its full potential. Knowledge of 3D modeling is required to create custom 3D prints, a specialized field. Due to time constraints, it was not possible to learn to create complicated 3D models during this project.

The Raspberry Pi case that was used in this project required modifications that could have been avoided if the 3D model had been modified beforehand, such as for passing cables and making space for the HATs. As a result of these modifications, the aesthetic appeal of the case was diminished.

8.5.1.1 Failed 3D Prints

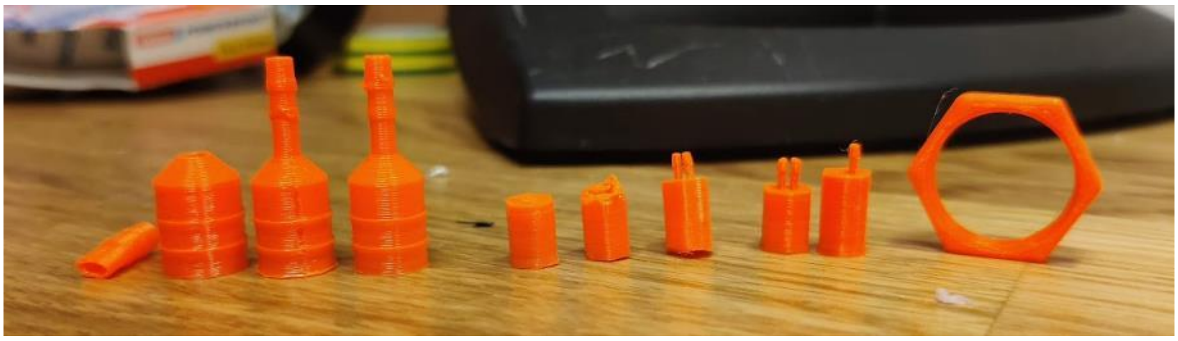


Figure 57: An assortment of failed 3D prints.

When 3D printing small parts, things can very easily go wrong. 3D printers can only print at a specific size, depending on what nozzle is used. On the 3D printer used in this project, the nozzle size was 0.4mm. This meant that any part that required a higher precision than 0.4mm could be printed incorrectly. Unfortunately, this project required multiple small parts, as seen in Figure 57.

On the left, we see three tube adapters. These were printed almost perfectly and worked in most cases. 3D prints are weakest between each layer, as seen in the left tube adapter, which snapped at a layer when trying to bend the tubing with the adapter attached. Number 2 had a tiny hole in the adapter due to an inaccurate print, and the last adapter was printed too small. These adapters were also impractical, as the tube had to be put atop the adapter. This meant that the inner diameter of the tube by the adapter was even smaller than before, possibly lowering the flow rate and throughput.

The following five prints are the Raspberry Pi HAT standoffs mentioned in section 6.1.2.3. The two HATs used in this project had slightly different heights and required different standoff sizes. By trial and error, the correct sizes were found. The mechanism that allows the standoffs to snap into place is flimsy due to its small size and broke off multiple times. Multiple standoffs were printed at once to ensure that backup pieces were available.

The M16x1 nut on the right was printed for the pushbutton, as seen in section 6.1.2.2, Due to the precision required to print a nut with this thread pitch. Unfortunately, this nut did not screw onto the pushbutton properly and had to be discarded.

8.5.2 Waterproofing and sealing

One of the most important dilemmas the M-CDS backpack needed to deal with was waterproofing and sealing the system. During development, water spilled on a Raspberry Pi due to insufficient waterproofing, leading to the Raspberry Pi being bricked. The juice should not spill out from the tubes and on a vital component, hampering the system. Ensuring that all the tubes were sealed and waterproof was a high priority.

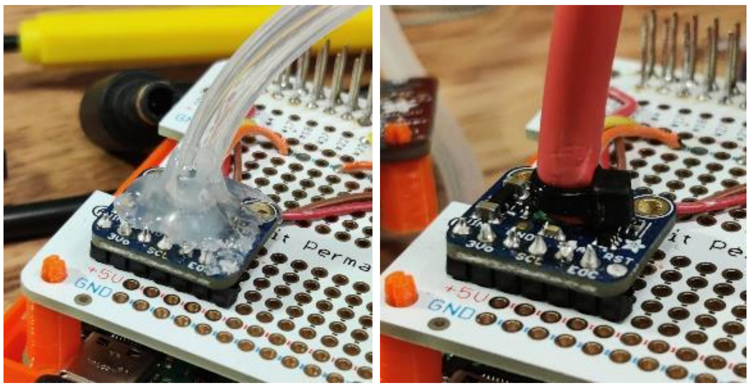


Figure 58: Two attempts at attaching the silicone tube to the MPRLS sensor. Left: hot glue, Right: heat shrink without glue, with a zip tie

The first attempts at waterproofing the system involved covering components and tubes with hot glue, as seen in Figure 58. At first, this seemed to work fine, but as the glue hardened, it became apparent that the silicone was not sticking to the hot glue. The silicone tube could easily be pulled out from wherever it was glued. A second attempt was made to attach the tubing using heat shrink and a zip-tie. This worked temporarily, but the zip tie used was too large and did not give a proper fit, and since the heat shrink did not have any glue, it did not stick to the tubing at all.

The solution, as seen in section 6.1.4.2, involved using heat shrink with glue inside along with a smaller zip tie. The glue from the heat shrink poured out from underneath the heat shrink as it was heated, fusing to the sensor and the heat shrink. Together with the zip tie, the tubing was stuck to the sensor.

The same issue occurred with the metal adapters attached to the drinking tubes. As seen in Figure 59, the silicone tube did not adhere to the heat shrink.

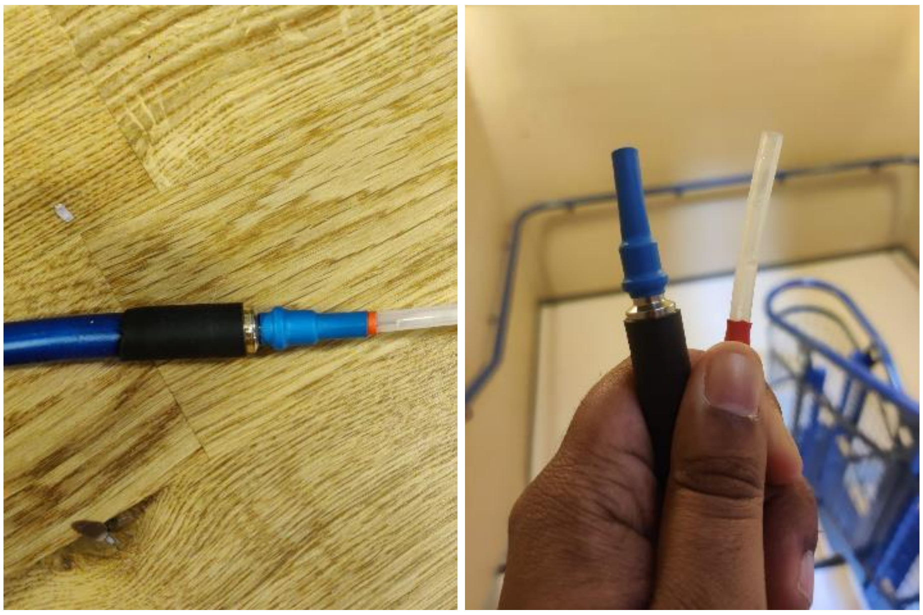


Figure 59: Failed attempt at attaching silicone tubes with glueless heat shrink

Instead, multiple layers of heat shrink with glue were added to ensure the glue could stick between the layers. The heat shrink shrank around the silicone tube very tightly to ensure it would stay stuck on the tube.

As mentioned in the section above, 3D-printed adapters were easy to produce but unreliable. The only proper adapters that would work were pressure fittings. Pressure fittings were used in the system for T-connectors and tube-to-tube adapters, such as the ones in Figure 60.

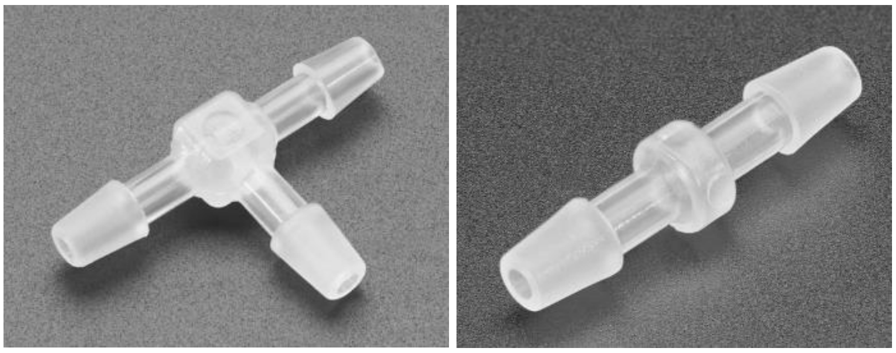


Figure 60: A T-connector and 2-prong adapter[20]

[20] https://www.adafruit.com/product/4662, https://www.adafruit.com/product/4764

These parts only require pushing the silicone tube over the end of the piece, and the connection is watertight and can withstand enough pressure. Unfortunately, adapters that convert from one size of silicone tube to another are difficult to find from professional retailers. They can be ordered from online marketplaces such as eBay and AliExpress, but they are not food grade and cannot be shipped quickly. However, they would simplify the prototype creation process, and if ordered early in the planning process, they could come on time.

8.5.3 Raspberry Pi Development

The development of scripts and main software was fairly straightforward, but some issues were met along the way.

When creating the watchdog application in the C programming language, there was a bug that didn't allow button presses to be read from the GPIO pins. A GPIO pin for a button must be initialized with the internal pull-up resistor enabled on the pin, otherwise, it is not possible to read the input from the button. After running a Python application that initialized the GPIO pins with the pull-up resistor, the C program suddenly worked as expected. The issue lay in the WiringPi library for developing C applications that could interface with GPIO pins. The library is shipped with all builds of the Raspberry Pi OS but has not been in active development for the past four years by its single developer.[21] A bug in the library's code broke the pull-up resistor functionality on the Raspberry Pi 4B. Running the Python script that initialized the GPIO pin enabled the C program to work as intended. The solution was to install a new, maintained version of the WiringPi library available on GitHub.[22]

The initial design used BLE to communicate between the Raspberry Pi and a mobile device. Implementing a BLE server on the Raspberry Pi in Python proved challenging, with tangled code and odd behaviors when trying to debug the code. Unfortunately, plenty of time was used to create this code, which left less time for the rest of the project and could have been avoided.

During development and testing, the backpack was accidentally dropped on the floor. While it seemed like everything was fine at first, the MPRLS sensor on the main board had stopped

[21] https://hackaday.com/2019/09/18/wiringpi-library-to-be-deprecated/

[22] https://github.com/WiringPi/WiringPi

working and would no longer read any data, crashing the M-CDS software on start. Thankfully, multiple sensors were ordered in case of such a situation. The sensor was replaced, and the software worked as intended again.

8.5.4 Mobile Application Development

As mentioned in the initial design section, the Raspberry Pi was originally planned to communicate with a mobile phone running an M-CDS mobile application through BLE, as the Raspberry Pi could not easily get a Wi-Fi connection while on the move. However, the final design and implementation does not use a mobile application.

At the beginning of the project, a lot of time was spent figuring out how to develop a mobile application as simple as possible and incorporate BLE into the application. Various languages, frameworks and technologies can be used, and many of these were tested to see which was the easiest to use. Unfortunately, many problems came along the way. A sample application was first created in Java using Android Studio, with the idea of creating an Android-only solution. As Java was a new language for me, many problems occurred during development, such as handling Bluetooth and communication permissions when creating a BLE connection.

A new application was then attempted using Xamarin.Forms in C#, to create a cross-platform program that would run on both Android and iOS. A test program was successfully created which sent data over BLE to the Raspberry Pi. However, the code was unstable and had bugs, such as not always connecting to the Raspberry Pi successfully. In addition, the code was not running in the background and would stop working when the user locked their phone or exited the application. A service would have to be made to allow for background data transfer, but this proved to be a more complicated solution.

Due to this design's lack of time and complexity, it was decided that it was better to use the mobile device’s Wi-Fi hotspot and connect the Raspberry Pi to the ad hoc network.

8.5.5 Backpack Design

The backpack is a proof-of-concept and works as expected, but specific parts of the system could be created differently.

The CamelBak is not designed to mount devices and parts inside or outside the backpack. When attaching the button, power switch, USB-C connector, power bank, and pump housing, holes were made in the backpack to fit the parts, screws, or zip ties through the backpack. The power bank is attached with a single zip-tie and double-sided tape, and although the power bank is secure, it is not very practical. If the power bank were to fail and require replacing, it would not be easy to remove it from the backpack. Creating holes in the backpack can also lead to the holes expanding over time, which could lead to parts and components becoming loose.

Aesthetically, the backpack does not look that good. There is an assortment of cables running in all directions, and taking the system out of the backpack without twisting the cables is difficult. The pump housing on the user's shoulder is very out of place as well, and the screws that go through the arm could potentially scratch the user's shoulder, although this did not happen during user testing.

8.5.6 Hypoglycaemia Detection Algorithm

The previous capstone project concluded with using the algorithm that Push-button-get-candy uses for calculating how many carbohydrates the user requires by using the user's carbohydrate sensitivity factor along with the current BGL to reach a target BGL. [35, 45] Due to time constraints and lack of user input to calculate the user's carbohydrate sensitivity factor, this method was dropped in favor of the simpler 15/15 method that House of Carbs uses. [7]

The algorithm is very primitive and assumes the user has a regular carbohydrate sensitivity, but the user testing shows that it worked well for the single participant. A better algorithm could also incorporate the user's weight and calculate their needed carbohydrates.

8.6 Testing and Usability

From the test result chart in Figure 56, we can see that the system detected four occurrences of a low or dangerously low BGL. The first incident was a low BGL detection, and the last three were dangerously low detections. The system successfully gave each of these doses (apart from the last dose which was cancelled) and did not crash or fail during execution. An effect from each dose occurred approximately 20 minutes after the dose was started. The first dose stopped the oncoming hypoglycaemia, as the participant's BGL never dropped below 69 mg/ml, and the second and third doses increased the user's BGL above the dangerously low threshold, proving that the system worked as intended.

A couple of issues became apparent during the testing and evaluation of the system’s usability. During a dose, there was no way to stop a dose and wait a specific amount of time, nor could the user request a dose on demand. Pressing the push button during a dose stopped the dose, but also turned off the software. If the user were to press the button again, the software would begin with the calibration phase, which could easily be skipped by pressing the button. However, if the user still had a low BGL, the system would immediately start a dose. Adding multiple actions through a mobile application could give the user more options than just stopping the dose. Alternatively, changing the software to stop a dose and wait a specific waiting period upon button press could also address the issue.

When testing the system with the participant, the Nightscout server described in section 6.2 was used along with a Freestyle Libre 3 sensor. This server URL and other settings were modified in the code before testing. If a new user were to test the system, the server URL and settings would have to be changed in code before they could use the system. This solution is suboptimal as the user needs knowledge of the internal workings of the code to change their settings, or the researcher responsible must change it. A dedicated settings file or website could have been used to modify settings or a mobile application could be used by directly connecting to the Raspberry Pi to alter these settings.

The system must be filled with juice to be used properly. Again, the properties of the juice are stored in the code on the Raspberry Pi, so using a juice with fewer carbohydrates per deciliter would give inaccurate doses if the settings were not changed. The drinking reservoir can be taken out of the system, but there is no valve to close the end of the tubing coming out of the reservoir. This could lead to liquid spilling out when filling it but can be circumvented by using a clip to seal the end, as shown in Figure 61. This is not convenient for the user at all. Adding a shutoff valve to the drinking reservoir could make it more secure for the user.

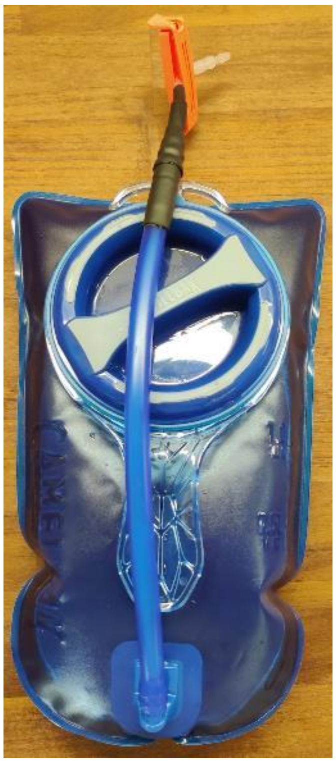


Figure 61: The drinking reservoir filled with juice and sealed with a bag clip

The Raspberry Pi connects to the internet using the user's mobile hotspot. The user must set their hotspot details (SSID and passphrase) to the details stored in the startup code on the Raspberry Pi. This is not user-friendly, but works fine for a proof-of-concept. A better solution would be to use BLE or 4G/5G to receive data from Nightscout and connect the Raspberry Pi to the internet.

8.7 Future Work

8.7.1 Multiple containers of liquid (with and without sugar, water)

Currently, the system only gives doses of juice with sugar to increase the user's BGL. An idea that occurred early in the project, but was dropped due to time constraints and complexity, was having three separate reservoirs of liquid: one with water, one with unmixed juice with artificial sweeteners, and one with unmixed juice with sugar. You could then pump out water if the user needs to be rehydrated, juice with sugar if they need to increase their BGL, and juice with artificial sweeteners if they want to drink juice without increasing their BGL. This would give the system a wider variety of use cases and further increase the likelihood of users using the system.

8.7.2 Better quality pump

The peristaltic pump used in this project is very slow. Generally, peristaltic pumps are not meant to be quick but are very accurate. Even though it was not a problem for the prototype, it would be better to have a quicker pump in order to reduce dosing times. A pump that worked twice as fast could lower the dosing times from 2:50 to 1:25. This would increase the user's blood glucose quicker, as more they would consume more carbohydrates in a shorter period.

8.7.3 Better hypoglycaemia detection algorithm

As mentioned in previous sections, the hypoglycaemia detection algorithm is very primitive. It only looks at the current BGL and Nightscout trend line to calculate whether it is time to give a dose or not, the same way House of Carbs does it.

Instead, the system could use the current insulin dose data and carbohydrates on board to determine if a dose is needed, which would probably be necessary for long-running sessions. In addition, the system could try to dose for a specific BGL target instead of a fixed amount of carbohydrates. If the user only has a low BGL with a rapidly declining trend line, it would be better to give a higher dose than if they had a low BGL with a flat trend line.

In the past months, AI and ChatGPT have become household names. It could be possible to feed the user's current and historical Nightscout data to GPT-4 and ask it to predict when the user will encounter a hypoglycaemic event. Other applications claim that they can accurately predict a user's BGL up to one hour in advance, but integration of these mobile applications data is unlikely to be possible.

8.7.4 Mobile phone and smartwatch usage

Although a mobile application was dropped, as mentioned in section 8.5.4, a future implementation of the M-CDS should have a mobile application to allow for simpler modification of settings on the Raspberry Pi, such as thresholds for hypoglycaemia detection and the user's Nightscout URL. Requesting or canceling doses, calibration, and other user input could also be done from the phone instead of a physical button on the backpack. Transfer of data from Nightscout could also be done from the mobile application, alleviating the need for a Wi-Fi connection on the Raspberry Pi. If a proper mobile application was developed and a good implementation of BLE on the Raspberry Pi, this solution would be much better and allow for an easier user experience.

Using a smartwatch to detect physical activity and other variables could also be helpful. If the user were to always have the M-CDS with them, the system could detect when the user is physically active and enable the system at specific times.

8.7.5 Low-power hardware

Briefly mentioned in section 8.4 were two other microcontrollers that could be used, an M5 Stack or FiPy. Both these devices would use less power and could be powered by a smaller battery. Still, the software would most likely have to be heavily modified to support different libraries and methods of accessing hardware and possibly programmed in a different language.

Another solution could be to use a Raspberry Pi Pico or Zero 2 W as the system's brains. The Pico is a microcontroller rather than a microcomputer, whereas the Zero 2 W is the smallest microcomputer available from Raspberry Pi. Both can be seen in Figure 62. The M-CDS does not require the use of any computer peripherals but uses HATs to power the motors and connect the various sensors. The Zero 2 W has a HAT pin header, allowing drop-in replacement of the current Raspberry Pi 4B. For the Pico, the HATs could be placed in a new container, and a ribbon cable could be connected from the required pins on the HAT to the PCB pads on the Pico. These devices support MicroPython, which would only require minor modifications of the current M-CDS code. A recent update to the Pico allowed for BLE connections, which would be useful for the suggested mobile application in the section above.[23]

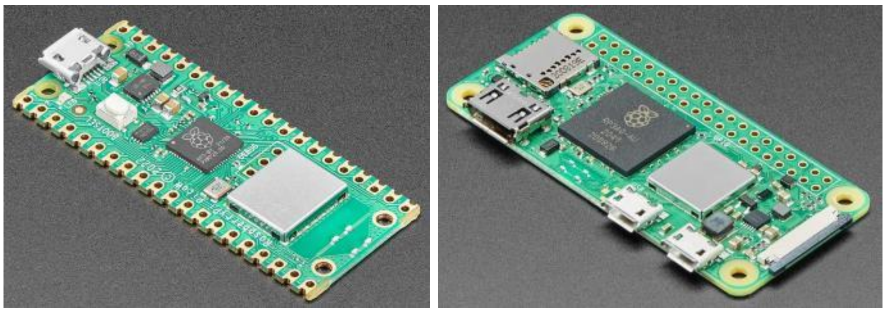


Figure 62: The Raspberry Pi Pico W and Zero 2 W[24]

[23] https://www.raspberrypi.com/news/new-functionality-bluetooth-for-pico-w/

[24] https://www.adafruit.com/product/5526, https://www.adafruit.com/product/5291

8.7.6 4G/5G direct connection

As mentioned in previous sections, the Raspberry Pi does not have any methods of connecting to the internet other than Wi-Fi and instead. If BLE is not an option either, a 4G or 5G HAT or mobile router could be connected to the Raspberry Pi. This would give it an internet connection as stable as the user's mobile phone's connection. However, due to time and budget constraints, this could not be done for this project. These devices are costly and would take time to set up to work properly. The FiPy mentioned in section 8.4 could have been used, as it natively supports a 4G SIM card.

8.7.7 Custom PCB

As a prototype, it was important to create a clear PCB design where cabling was visible and it was simple to see where all the different parts were connected. If a smaller end product was a requirement, it could be better to design a custom PCB board to house all the required components, such as the motor controller and sensors. This would allow the system to be physically smaller and more mass-production-ready. If using the system with a Raspberry Pi Pico, some sort of PCB would be better than using HATs, as the Pico does not have a 40-pin GPIO connector and instead PCB pads around the perimeter of the microcontroller.

8.7.8 Enclosed container for the entire system

One issue with the M-CDS backpack that came to light during the interview was the fact that the system was a backpack. The interviewee would be more willing to use the system if it was enclosed in something they could put into their existing backpack. This would also make the system more secure, as the internal parts would not be able to be tampered with or easily damaged. A large plastic enclosure that could fit in a backpack pocket, laptop insert or side bottle pocket would make the system portable and more production ready.

8.8 Research Problems

In section 1.2, the main research problem and the subproblems were presented. In this section, we discuss and answer each problem.

RP: How do we design a mobile carbohydrate delivery system?

The system designed in this thesis is a successful prototype of a mobile carbohydrate delivery system. A practical solution can be made by ensuring minimal user input to use the system, simple feedback through a buzzer, a sealed system ensuring no liquid leaking, and a simple drinking tube to intake carbohydrates. Using an existing backpack to house the system led to simpler prototyping and a quicker proof-of-concept. The Design and Implementation sections answer this research problem.

RP1: How can we use a sealed system to pump liquid from a container?

By utilizing a flexible drinking reservoir that can expand and contract as liquid is added or removed, we can create a sealed system that does not need a fixed-size container or a pressure release valve. In addition, by waterproofing all tubes and only allowing a single exit point for the internal pressure – using the bite valve – we can limit the points where liquid could spill in the system. To avoid pressure building up, we utilize a MPRLS pressure sensor to stop the pump in case the pressure becomes too high in the system. The peristaltic pump will run only when the user is drinking using the bite valve. This research problem is answered in sections 6.1.1, 6.1.4.2, and 8.5.2.

RP2: How can we safely encase the system in a portable container?

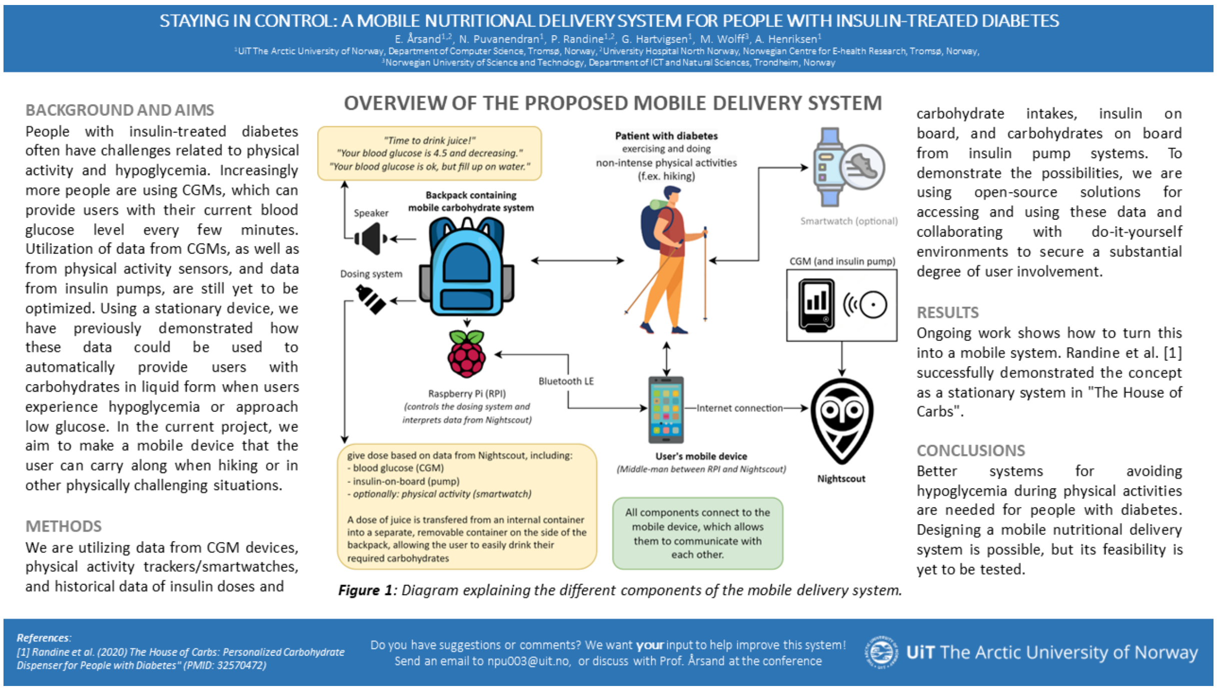
In order to secure the fragile electronics of the system, a 3D printed case was used to enclose all the hardware components. The liquid and non-liquid parts of the system were separated as much as possible, and almost every component was fixed to the M-CDS backpack using zip ties, screws, or double-sided tape.

9 Dissemination

The findings from this project and proposed designs have been disseminated to two conferences, ATTD 2023 and Diabetes Forum 2023.

9.1 Poster for ATTD 2023

A poster and audio recording of the poster was created for ATTD 2023 [61]. The poster was submitted as an ePoster, as there were no physical posters at the conference. As this poster was submitted in February 2023, the information included the proposed design and descriptions of how the system would be implemented. The poster can be seen in Figure 63.



STAYING IN CONTROL: A MOBILE NUTRITIONAL DELIVERY SYSTEM FOR PEOPLE WITH INSULIN-TREATED DIABETES

E. Årsand[1,2], N. Puvanendran[1], P. Randine[1,2], G. Hartvigsen[1], M. Wolff[3], A. Henriksen[1]

[1]UiT The Arctic University of Norway, Department of Computer Science, Tromsø, Norway, [2]University Hospital North Norway, Norwegian Centre for E-health Research, Tromsø, Norway, [3]Norwegian University of Science and Technology, Department of ICT and Natural Sciences, Trondheim, Norway

BACKGROUND AND AIMS

People with insulin-treated diabetes often have challenges related to physical activity and hypoglycemia. Increasingly more people are using CGMs, which can provide users with their current blood glucose level every few minutes. Utilization of data from CGMs, as well as from physical activity sensors, and data from insulin pumps, are still yet to be optimized. Using a stationary device, we have previously demonstrated how these data could be used to automatically provide users with carbohydrates in liquid form when users experience hypoglycemia or approach low glucose. In the current project, we aim to make a mobile device that the user can carry along when hiking or in other physically challenging situations.

METHODS

We are utilizing data from CGM devices, physical activity trackers/smartwatches, and historical data of insulin doses and carbohydrate intakes, insulin on board, and carbohydrates on board from insulin pump systems. To demonstrate the possibilities, we are using open-source solutions for accessing and using these data and collaborating with do-it-yourself environments to secure a substantial degree of user involvement.

OVERVIEW OF THE PROPOSED MOBILE DELIVERY SYSTEM

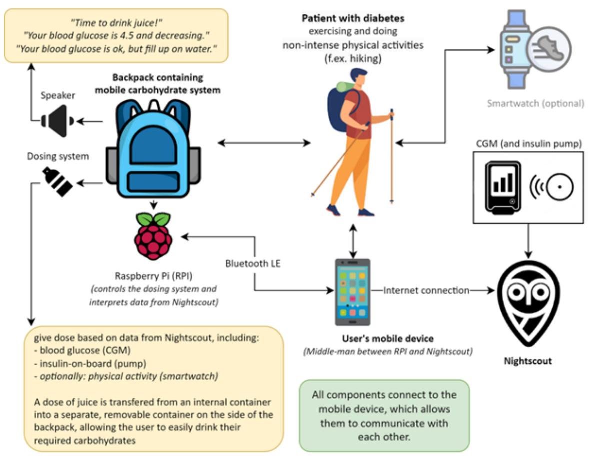

"Time to drink juice!"
"Your blood glucose is 4.5 and decreasing."
"Your blood glucose is ok, but fill up on water."
Speaker
Backpack containing mobile carbohydrate system
Dosing system
Patient with diabetes exercising and doing non-intense physical activities (f.ex. hiking)
Smartwatch (optional)
CGM (and insulin pump)
Raspberry Pi (RPI) (controls the dosing system and interprets data from Nightscout)
Bluetooth LE
Internet connection
User's mobile device (Middle-man between RPI and Nightscout)
Nightscout
give dose based on data from Nightscout, including:
- blood glucose (CGM)
- insulin-on-board (pump)
- optionally: physical activity (smartwatch)
A dose of juice is transfered from an internal container into a separate, removable container on the side of the backpack, allowing the user to easily drink their required carbohydrates
All components connect to the mobile device, which allows them to communicate with each other.

***Figure 1**: Diagram explaining the different components of the mobile delivery system.*

RESULTS

Ongoing work shows how to turn this into a mobile system. Randine et al. [1] successfully demonstrated the concept as a stationary system in "The House of Carbs".

CONCLUSIONS

Better systems for avoiding hypoglycemia during physical activities are needed for people with diabetes. Designing a mobile nutritional delivery system is possible, but its feasibility is yet to be tested.

References:
[1] Randine et al. (2020) The House of Carbs: Personalized Carbohydrate Dispenser for People with Diabetes" (PMID: 32570472)

Do you have suggestions or comments? We want **your** input to help improve this system!
Send an email to npu003@uit.no, or discuss with Prof. Årsand at the conference

UiT The Arctic University of Norway

Figure 63: The ATTD 2023 poster.

9.2 Poster Presentation for Diabetes Forum 2023

As mentioned in section 3.4, a poster presentation was created for Diabetes Forum 2023, displaying the House of Carbs and a detailed, proposed design of the M-CDS backpack. This presentation took place in April 2023 and contained more detailed information on how the specific parts of the system work together. The supervisor of this project presented this project, as seen below.

The House of Carbs
A (stationary) machine for automatically delivery of carbohydrates in liquid form
Previous master thesis by Pietro Randine, 2018
Described in paper: Randine, Pietro; Micucci, Daniela; Hartvigsen, Gunnar; Årsand, Eirik.
The House of Carbs: Personalized Carbohydrate Dispenser for People with Diabetes. Studies in Health Technology and Informatics 2020; Volum 270. s. 693-697
diabetesforbundet

Figure 11: Presentation at Diabetes Forum 2023

10 Conclusion

This thesis presents a proof-of-concept and prototype of a mobile carbohydrate delivery system, as existing systems are neither mobile nor exist. The system presented began with a design that changed over the progress of the implementation stage, as new ideas and problems appeared along the way.

The final implementation is a modified CamelBak that uses a drinking reservoir and peristaltic pump to pump juice into a drinking tube, allowing the user to consume carbohydrates when required easily. Data from the user's CGM is transferred to Nightscout, which a Raspberry Pi seated in the CamelBak can then read. When the user's BGL drops below a fixed point, the system starts a dose of 15 grams of carbohydrates and then waits 15 minutes before making a new decision.

The results show that the system works as intended and can stop a hypoglycaemic event during physical activity by pumping juice from a container into a drinking tube multiple times in a row without mechanical failure. However, as this is a proof-of-concept, many elements could be improved to create a product of higher quality. Future work includes using a better algorithm to detect hypoglycaemia, the introduction of a mobile application to control and configure the system, and containing the system in a smaller enclosure that can be moved freely without requiring a modified backpack.

References

[1] D. E. Warburton, C. W. Nicol, and S. S. Bredin, "Health benefits of physical activity: the evidence," (in eng), *Cmaj,* vol. 174, no. 6, pp. 801-9, Mar 14 2006, doi: 10.1503/cmaj.051351.

[2] American Diabetes Association. "Blood Sugar and Exercise | ADA." https://diabetes.org/healthy-living/fitness/getting-started-safely/blood-glucose-and-exercise (accessed March, 2023).

[3] P. V. Röder, B. Wu, Y. Liu, and W. Han, "Pancreatic regulation of glucose homeostasis," (in eng), *Exp Mol Med,* vol. 48, no. 3, p. e219, Mar 11 2016, doi: 10.1038/emm.2016.6.

[4] M. Bisgaard Bengtsen and N. Møller, "Mini-review: Glucagon responses in type 1 diabetes – a matter of complexity," *Physiological Reports,* vol. 9, no. 16, 2021, doi: 10.14814/phy2.15009.

[5] N. Lascar *et al.*, "Attitudes and barriers to exercise in adults with type 1 diabetes (T1DM) and how best to address them: a qualitative study," (in eng), *PLoS One,* vol. 9, no. 9, p. e108019, 2014, doi: 10.1371/journal.pone.0108019.

[6] V. Lyssenko, L. Groop, and R. B. Prasad, "Genetics of Type 2 Diabetes: It Matters From Which Parent We Inherit the Risk," (in eng), *Rev Diabet Stud,* vol. 12, no. 3-4, pp. 233-42, Fall-Winter 2015, doi: 10.1900/rds.2015.12.233.

[7] P. Randine, D. Micucci, G. Hartvigsen, and E. Årsand, "The House of Carbs: Personalized Carbohydrate Dispenser for People with Diabetes," 2020.

[8] Various Authors. "How do I upload my data? - Nightscout." https://nightscout.github.io/uploader/uploaders/ (accessed October, 2022).

[9] World Health Organization. "Diabetes." https://www.who.int/news-room/fact-sheets/detail/diabetes (accessed October, 2022).

[10] A. R. Gosmanov, E. O. Gosmanova, and A. E. Kitabchi, "Hyperglycemic Crises: Diabetic Ketoacidosis and Hyperglycemic Hyperosmolar State," in *Endotext*, K. R. Feingold *et al.* Eds. South Dartmouth (MA): MDText.com, Inc., 2000.

[11] D. J. Magliano, E. J. Boyko, and IDF Diabetes Atlas 10th edition scientific committee, "IDF Diabetes Atlas," in *Idf diabetes atlas*. Brussels: International Diabetes Federation, 2021.

[12] Diabetesforbundet. "Blodsukker og måling." https://www.diabetes.no/diabetes-type-1/behandling/blodsukker/ (accessed March, 2023).

[13] A. Nakhleh and N. Shehadeh, "Hypoglycemia in diabetes: An update on pathophysiology, treatment, and prevention," (in eng), *World J Diabetes,* vol. 12, no. 12, pp. 2036-2049, Dec 15 2021, doi: 10.4239/wjd.v12.i12.2036.

[14] L. C. M. Stene and Various Authors. "Diabetes i Norge." https://www.fhi.no/nettpub/hin/ikke-smittsomme/diabetes/ (accessed March, 2023).

[15] National Institute of Diabetes and Digestive and Kidney Diseases. "Symptoms & Causes of Diabetes - NIDDK." https://www.niddk.nih.gov/health-information/diabetes/overview/symptoms-causes (accessed March, 2023).

[16] M. Ibrahim *et al.*, "Hypoglycaemia and its management in primary care setting," (in eng), *Diabetes Metab Res Rev,* vol. 36, no. 8, p. e3332, Nov 2020, doi: 10.1002/dmrr.3332.

[17] D. C. Klonoff, D. Ahn, and A. Drincic, "Continuous glucose monitoring: A review of the technology and clinical use," *Diabetes Research and Clinical Practice,* vol. 133, pp. 178-192, 2017/11/01/ 2017, doi: 10.1016/j.diabres.2017.08.005.

[18] National Institute of Diabetes and Digestive and Kidney Diseases. "Continuous Glucose Monitoring - NIDDK." https://www.niddk.nih.gov/health-information/diabetes/overview/managing-diabetes/continuous-glucose-monitoring (accessed March, 2023).
[19] H. A. Wolpert, "Use of continuous glucose monitoring in the detection and prevention of hypoglycemia," (in eng), *J Diabetes Sci Technol*, vol. 1, no. 1, pp. 146-50, Jan 2007, doi: 10.1177/193229680700100126.
[20] M. C. Riddell and J. Milliken, "Preventing exercise-induced hypoglycemia in type 1 diabetes using real-time continuous glucose monitoring and a new carbohydrate intake algorithm: an observational field study," (in eng), *Diabetes Technol Ther*, vol. 13, no. 8, pp. 819-25, Aug 2011, doi: 10.1089/dia.2011.0052.
[21] Diabetesforbundet. "Insulinpumper og sensorer | Diabetesforbundet." https://www.diabetes.no/diabetes-type-1/behandling/insulinpumper-og-sensorer/#section7 (accessed October, 2022).
[22] Supersapiens. "Supersapiens | Glucose Monitor For Metabolic Efficiency." https://www.supersapiens.com/ (accessed.
[23] C. Josh. (2021, 2021/05/05/T16:22:50Z) Supersapiens review: The first continuous blood glucose monitor for performance. *cyclingnews.com*. Available: https://www.cyclingnews.com/reviews/supersapiens-review-the-first-continuous-blood-glucose-monitor-for-performance/
[24] iFixit. "FreeStyle Libre 2 Sensor Teardown." https://www.ifixit.com/Teardown/FreeStyle+Libre+2+Sensor+Teardown/137719 (accessed.
[25] minimallooper. "How to setup FreeStyle Libre 2 and OOP2 to use a native Bluetooth connection in xDrip+." https://www.minimallooper.com/post/how-to-setup-freestyle-libre-2-and-oop2-to-use-a-native-bluetooth-connection-in-xdrip (accessed October, 2022).
[26] Veri. "Veri - Discover Better Metabolic Health." https://www.veri.co/ (accessed.
[27] Veri. "Learn More About Your Sensor | Veri." https://help.veri.co/en/articles/6295557-learn-more-about-your-sensor (accessed.
[28] Veri. "Who is Veri for? | Veri." https://help.veri.co/en/articles/4769250-who-is-veri-for (accessed.
[29] Levels. "About Levels - Levels Support." https://support.levelshealth.com/article/86-what-is-levels (accessed.
[30] Levels. "Levels - Unlock Your Metabolic Health." https://www.levelshealth.com/ (accessed.
[31] Levels. "Set Up Levels with FreeStyle Libre CGM - Levels Support." https://support.levelshealth.com/category/149-setting-up-your-levels (accessed.
[32] January AI. "Glucose Tracking, Insights, and a 24/7 AI Health Coach | January AI." https://www.january.ai/ (accessed.
[33] Ashkan Dehghani Zahedani; Arvind Veluvali; Saransh Agarwal; Jiayu Zhou; Jingyi Ruan; Michael Snyder; Nima Aghaeepour; January AI, "Virtual blood glucose monitoring and prediction using machine learning," 2023.
[34] January AI. "The January Program." https://support.january.ai/support/solutions/43000371643 (accessed.
[35] C. Hannemann. "push-button-get-candy." https://github.com/channemann/push-button-get-candy (accessed November, 2022).

[36] J. Kropff and J. H. DeVries, "Continuous Glucose Monitoring, Future Products, and Update on Worldwide Artificial Pancreas Projects," (in eng), *Diabetes Technol Ther,* vol. 18 Suppl 2, no. Suppl 2, pp. S253-63, Feb 2016, doi: 10.1089/dia.2015.0345.
[37] L. Bally *et al.*, "Closed-Loop Insulin Delivery for Glycemic Control in Noncritical Care," *New England Journal of Medicine,* vol. 379, no. 6, pp. 547-556, 2018, doi: 10.1056/NEJMoa1805233.
[38] N. Taleb *et al.*, "Efficacy of single-hormone and dual-hormone artificial pancreas during continuous and interval exercise in adult patients with type 1 diabetes: randomised controlled crossover trial," (in eng), *Diabetologia,* vol. 59, no. 12, pp. 2561-2571, Dec 2016, doi: 10.1007/s00125-016-4107-0.
[39] Various Authors. "What is Nightscout?" https://nightscout.github.io/ (accessed December, 2022).
[40] Raspberry Pi Ltd. "Raspberry Pi." https://www.raspberrypi.com/ (accessed.
[41] SparkFun, "I2C - SparkFun Learn." [Online]. Available: https://learn.sparkfun.com/tutorials/i2c/all.
[42] P. J. Denning *et al.*, "Computing as a discipline," *Computer,* vol. 22, no. 2, pp. 63-70, 1989, doi: 10.1109/2.19833.
[43] J. Robertson and S. Robertson, "Volere Requirements Specification Template Edition 16," 01/01 2012.
[44] V. Gokhare, D. Raut, and D. Shinde, "A Review paper on 3D-Printing Aspects and Various Processes Used in the 3D-Printing," *International Journal of Engineering and Technical Research,* vol. 6, pp. 953-958, 06/06 2017.
[45] N. Puvanendran, "Designing an Intelligent Mobile Nutritional Delivery-System," Unpublished capstone project, 2023.
[46] N. Murphy and M. Barr, "Watchdog timers," *Embedded Systems Programming,* vol. 14, no. 11, pp. 79-80, 2001.
[47] J. Klespitz and L. Kovács, "Peristaltic pumps—A review on working and control possibilities," in *2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2014: IEEE, pp. 191-194.
[48] PSG. "Peristaltic Pumps." https://www.psgdover.com/abaque/products/peristaltic-pumps (accessed.
[49] Abbott. "LibreLinkUp." https://librelinkup.com/ (accessed October, 2022).
[50] Y. Dong, R. Wen, Z. Li, K. Zhang, and L. Zhang, "Clu-RNN: A New RNN Based Approach to Diabetic Blood Glucose Prediction," in *2019 IEEE 7th International Conference on Bioinformatics and Computational Biology (ICBCB)*, 21-23 March 2019 2019, pp. 50-55, doi: 10.1109/ICBCB.2019.8854670.
[51] Y. Dong, R. Wen, K. Zhang, and L. Zhang, "A Novel RNN-Based Blood Glucose Prediction Approach Using Population and Individual Characteristics," in *2019 IEEE 7th International Conference on Bioinformatics and Computational Biology (ICBCB),* 21-23 March 2019 2019, pp. 145-149, doi: 10.1109/ICBCB.2019.8854657.
[52] J. Xie and Q. Wang, "Benchmarking Machine Learning Algorithms on Blood Glucose Prediction for Type I Diabetes in Comparison With Classical Time-Series Models," (in eng), *IEEE Trans Biomed Eng,* vol. 67, no. 11, pp. 3101-3124, Nov 2020, doi: 10.1109/tbme.2020.2975959.
[53] X. Chen, X. Ma, and L. Shi, "Research on blood glucose data prediction based on I-GWO-KELM algorithm," in *2021 40th Chinese Control Conference (CCC),* 26-28 July 2021 2021, pp. 2698-2702, doi: 10.23919/CCC52363.2021.9550442.

[54] S. L. Cichosz, M. H. Jensen, and O. Hejlesen, "Short-term prediction of future continuous glucose monitoring readings in type 1 diabetes: Development and validation of a neural network regression model," (in eng), *Int J Med Inform,* vol. 151, p. 104472, Jul 2021, doi: 10.1016/j.ijmedinf.2021.104472.

[55] M. F. Rabby, Y. Tu, M. I. Hossen, I. Lee, A. S. Maida, and X. Hei, "Stacked LSTM based deep recurrent neural network with kalman smoothing for blood glucose prediction," (in eng), *BMC Med Inform Decis Mak,* vol. 21, no. 1, p. 101, Mar 16 2021, doi: 10.1186/s12911-021-01462-5.

[56] W. Seo, S. W. Park, N. Kim, S. M. Jin, and S. M. Park, "A personalized blood glucose level prediction model with a fine-tuning strategy: A proof-of-concept study," (in eng), *Comput Methods Programs Biomed,* vol. 211, p. 106424, Nov 2021, doi: 10.1016/j.cmpb.2021.106424.

[57] S. M. A. Zaidi, V. Chandola, M. Ibrahim, B. Romanski, L. D. Mastrandrea, and T. Singh, "Multi-step ahead predictive model for blood glucose concentrations of type-1 diabetic patients," (in eng), *Sci Rep,* vol. 11, no. 1, p. 24332, Dec 21 2021, doi: 10.1038/s41598-021-03341-5.

[58] T. Zhu, L. Kuang, K. Li, J. Zeng, P. Herrero, and P. Georgiou, "Blood Glucose Prediction in Type 1 Diabetes Using Deep Learning on the Edge," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 22-28 May 2021 2021, pp. 1-5, doi: 10.1109/ISCAS51556.2021.9401083.

[59] M. Syafrudin, G. Alfian, N. L. Fitriyani, I. Fahrurrozi, M. Anshari, and J. Rhee, "A Personalized Blood Glucose Prediction Model Using Random Forest Regression," in *2022 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, 22-23 June 2022 2022, pp. 295-299, doi: 10.1109/ICETSIS55481.2022.9888838.

[60] T. Zhu, K. Li, P. Herrero, and P. Georgiou, "Personalized Blood Glucose Prediction for Type 1 Diabetes Using Evidential Deep Learning and Meta-Learning," (in eng), *IEEE Trans Biomed Eng,* vol. 70, no. 1, pp. 193-204, Jan 2023, doi: 10.1109/tbme.2022.3187703.

[61] "The Official Journal of ATTD Advanced Technologies & Treatments for Diabetes Conference 22 ‐ 25 February 2023 I Berlin & Online," *Diabetes Technology & Therapeutics,* vol. 25, no. S2, pp. A-1-A-269, 2023, doi: 10.1089/dia.2023.2525.abstracts.

Appendix

First email to UiT's DPO

Hi!

I am a master student at the Department of Computer Science writing a master on a mobile carbohydrate delivery system for patients with diabetes. My main supervisor (Eirik Årsand) is CC'd on this mail.

This system will read a user's blood glucose level, insulin levels, and carbohydrate intake from an online system to determine whether the patients require more carbohydrates to increase their blood glucose level.

The following data is collected from the user (both historical data and real-time)

- Blood glucose levels
- Insulin-on-board/insulin levels*
- Carbohydrate intake*
- Physical activity data from user's smartwatch/fitness tracker*

** optional data*

The data is collected from a private server running the Nightscout software. Nightscout is a continuous blood glucose monitor that runs in the cloud, allowing for access of a patient's blood glucose levels from any device with an internet connection.

Although this is personal data, there is no way to trace the data back to a user. There is no name, birthdate, email, phone number or any other personally identifiable data that can be directly or indirectly used to figure out whose data is uploaded to the server. Their data will also be deleted from the server when the project is finished.

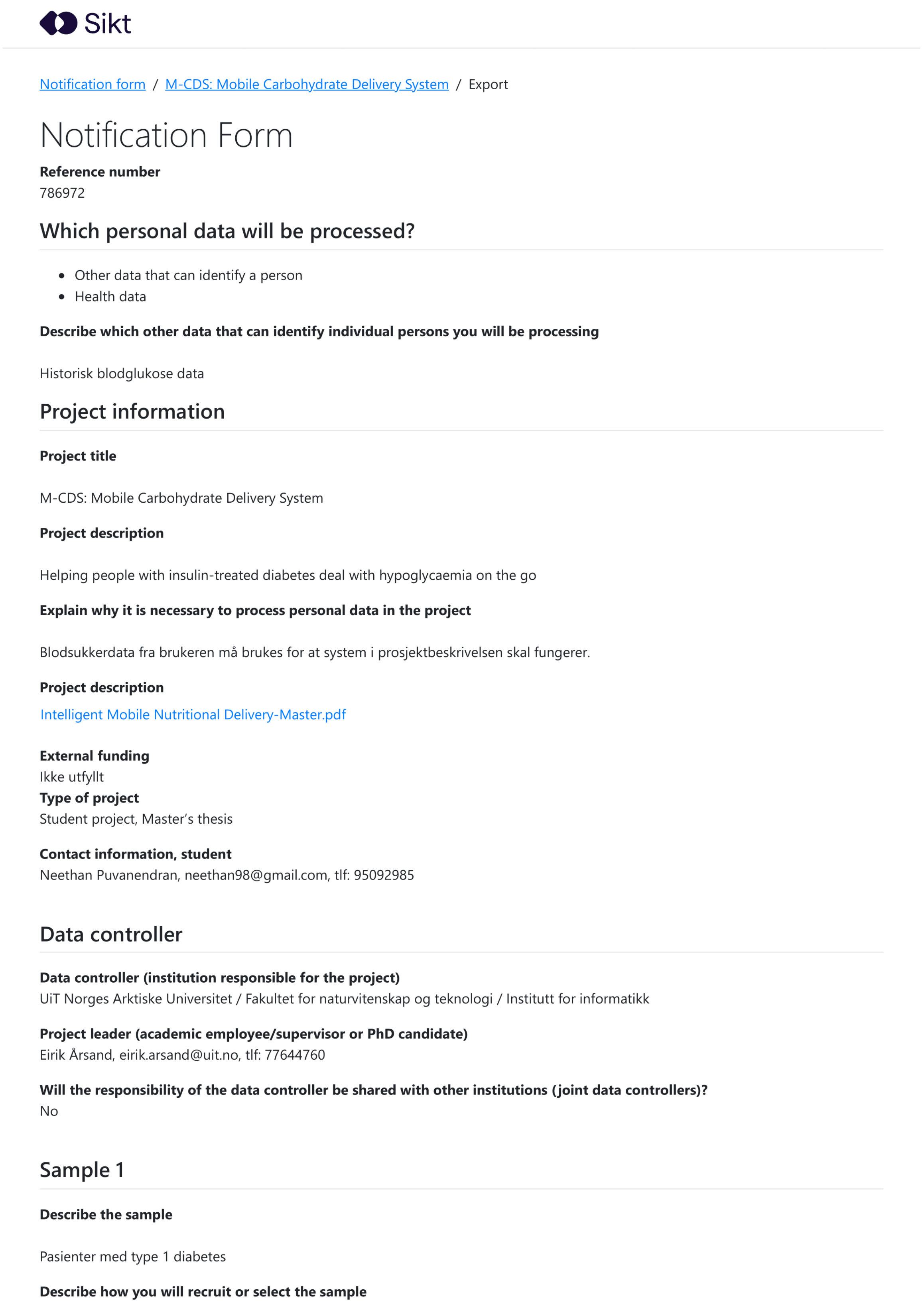
In addition, Nightscout is an open-source software and can be run by the user themselves. If the user chooses to use their own Nightscout server with my system, then I have no access to the server, and can only access the diabetes-related data without any identification of the user.

I will also be conducting interviews with users who have diabetes, for user testing and for designing the system described above. The interviews will be audio recorded, transcribed and then anonymized.

I assume I will have to send an application to Sikt for the interviews, but do I need to include the system in the application as well? Do I need to send an application to REK as well?

Regards,
Neethan Puvanendran
95 09 29 85

Application to Sikt



Sikt

Notification form / M-CDS: Mobile Carbohydrate Delivery System / Export

Notification Form

Reference number
786972

Which personal data will be processed?

- Other data that can identify a person
- Health data

Describe which other data that can identify individual persons you will be processing

Historisk blodglukose data

Project information

Project title

M-CDS: Mobile Carbohydrate Delivery System

Project description

Helping people with insulin-treated diabetes deal with hypoglycaemia on the go

Explain why it is necessary to process personal data in the project

Blodsukkerdata fra brukeren må brukes for at system i prosjektbeskrivelsen skal fungerer.

Project description

Intelligent Mobile Nutritional Delivery-Master.pdf

External funding
Ikke utfyllt
Type of project
Student project, Master's thesis

Contact information, student
Neethan Puvanendran, neethan98@gmail.com, tlf: 95092985

Data controller

Data controller (institution responsible for the project)
UiT Norges Arktiske Universitet / Fakultet for naturvitenskap og teknologi / Institutt for informatikk

Project leader (academic employee/supervisor or PhD candidate)
Eirik Årsand, eirik.arsand@uit.no, tlf: 77644760

Will the responsibility of the data controller be shared with other institutions (joint data controllers)?
No

Sample 1

Describe the sample

Pasienter med type 1 diabetes

Describe how you will recruit or select the sample

Utvalget er pasienter som studenten har personlig kontakt med

Age
21 - 65

Are any of these groups included in the sample?
- Patients or ill people

Personal data relating to sample 1
- Other data that can identify a person
- Health data

How will you collect data relating to sample 1?

Web-based experiment

Legal basis for processing general categories of personal data
Consent (General Data Protection Regulation art. 6 nr. 1 a)

Legal basis for processing special categories of personal data
Explicit consent (General Data Protection Regulation art. 9 nr. 2 a)

Explain your choice of legal basis

Information for sample 1

Will you inform the sample about the processing of their personal data?
Yes

How?
Written information (on paper or electronically)

Information letter

Vil du delta i en forskningsprosjekt mcds.docx

Third Persons

Will you be processing data relating to third persons?
No

Documentation

How will consent be documented?
- Manually (on paper)

How can consent be withdrawn?

Muntlig, prate med veileder eller studenten

How can data subjects get access to their personal data or have their personal data corrected or deleted?

Muntlig, prate med veileder eller studenten

Total number of data subjects in the project
1-99

Approvals

Will you obtain any of the following approvals or permits for the project?
Ikke utfyllt

Processing

Where will the personal data be processed?

- Physically isolated computer belonging to the data controller

Who will be processing/have access to the collected personal data?

- Project leader
- Student (student project)

Will the collected personal data be transferred/made available to a third country or international organisation outside the EU/EEA?

No

Information Security

Will directly identifiable data be stored separately from the rest of the collected data (e.g. in a scrambling key)?

No

Explain why directly identifiable data will be stored together with the rest of the collected data

Det er ingen personopplysninger som lagres enn blodsukkerdata

Which technical and practical measures will be used to secure the personal data?

- Restricted access
- Personal data will be sent/transferred in encrypted form
- Personal data will be anonymised as soon as no longer needed
- Personal data will be stored in encrypted form

Duration of processing

Project period

15.01.2023 - 15.06.2023

What happens to the data at the end of the project?

All data will be deleted (deleting raw data)

Will the data subjects be identifiable (directly or indirectly) in the thesis/publications from the project?

No

Additional information