UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

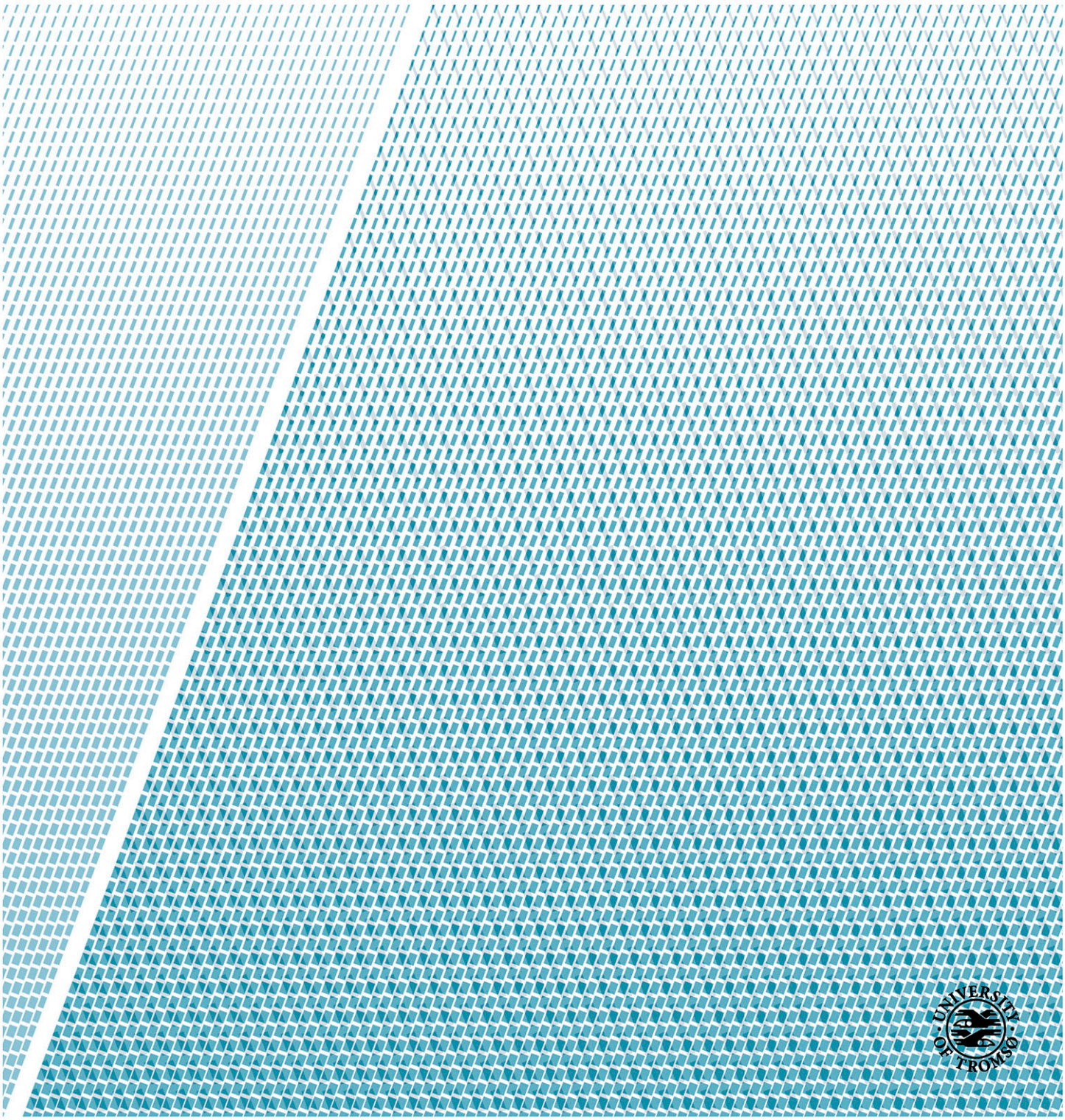
Faculty of Science and Technology
Department of Computer Science

Diabetes Automata For Diabetes-Related Applications: Software Engine For Blood Glucose Level Simulation

—

Aleksandr Agafonov

INF-3990 Master thesis in Computer Science – July 2015



INF-3990

Master's Thesis in Computer Science

DIABETES AUTOMATA FOR DIABETES-RELATED APPLICATIONS: SOFTWARE ENGINE FOR BLOOD GLUCOSE LEVEL SIMULATION

Aleksandr Agafonov

July 2015

Copyright © 2015
by Aleksandr Agafonov

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author.

Preface

Hundreds of millions of people around the world have diabetes mellitus, which is one of the most common diseases and one of the leading causes of death. Diabetes type 1 is the type of diabetes that cannot be treated and makes the life of one with diabetes type 1 depend on insulin injections. The disease lowers the quality of life, since one with diabetes type 1 has to manage the disease all the time – control the amount of carbohydrates in meals and calculate the insulin dosage correctly, in order to stay alive and feel good. Good management of one's blood glucose levels makes living with diabetes much easier.

There are various existing diabetes-related electronic tools both for patients and specialists. Among them are serious games that are aimed to improve the patient's knowledge about the disease and skills in diabetes self-management; mobile applications and diabetes self-management tools aimed to help patients to track their blood glucose management (an example of such tool is a mobile based few-touch application Diabetesdagboka by Norwegian Center for Integrated Care and Telemedicine (NST); clinical decision support systems that help specialists to provide high quality help and advices to patients.

Many diabetes-related electronic tools have algorithms that try to simulate blood glucose levels depending on biometric information and information about injected insulin, consumed carbohydrates in meals and other elements that affect blood glucose levels. Therefore, the idea of the Diabetes Automata project was to develop a software engine for such diabetes-related tools that can be parametrized for a particular person and output predicted/simulated blood glucose levels, depending on the input information coming from the tool.

The software engine would be an important part of many diabetes-related applications and the heart of their blood glucose calculation functionality. It would be an already implemented module that can be imported and used in applications or systems on various platforms (for example pc-based, mobile-based, web-based), and the developers would not have to focus on implementing complex blood glucose calculation algorithms, but on other important aspects, such as, for example, usability, reliability, quality of their system or application, etc.

In the beginning of the research, the project supposed to be a game-oriented engine and a part of a diabetes-related serious game by NST. During the project development, Diabetes Automata became no longer a part of the game because of the changes in the game concepts. The opportunities for the comprehensive testing in practice disappeared, however, alternative testing methods were found and conducted. The scope of Diabetes Automata project was revised and made wider – from a game-oriented software engine to an engine that could be used in diabetes-related applications and systems, including serious games.

Diabetes Automata is a master project in Computer Science that I started at Department of Computer Science of the UiT The Arctic University Of Norway and NST. The supervisors of the project is Gunnar Hartvigsen, the co-supervisors are Eirik Årsand, Alexandra Makhlysheva and Håvard Blixgård. The prototype developed during the project work, consists of two parts – the software engine itself, and a mobile-phone base demonstrator application.

In order to conduct project research and implement project prototype, several methods were used, among which are the search and the review of literature and information sources, multiple meetings and discussions with supervisors and specialists. Before starting the implementation of the engine and the demonstrator, their design and architecture were created as digital mockups. During the development, Eirik Årsand has been constantly testing and providing useful feedback messages. Before the prototype was tested, several types of testing were defined. Eirik Årsand (test person 1) and Anna Holubová (test person 2) were the testers of the prototype. They were testing the prototype and provided their personal diabetes self-management data, which made me able to do the most difficult part of testing by myself – the prototype accuracy testing. The accuracy testing showed good results with room for improvement.

My motivation, and the reason why I chose Diabetes Automata as my master project, is that the project was interesting, and at the same time complex, very challenging and experimental, while increasing the level of my knowledge in multiple areas. This research project combines software/system design and development, object-oriented programming, mobile application development, experimental health informatics and electronic health (eHealth) in general and mobile health (mHealth) in particular, and some part of medical biology and mathematics.

I would like to thank:

Family, parents Aleksandr Agafonov and Marina Agafonova;

Supervisor Gunnar Hartvigsen (professor, UiT, NST), co-supervisor and tester Eirik Årsand (professor, NST, UiT), co-supervisors Alexandra Makhlysheva (NST) and Håvard Blixgård (NST);

Tester Anna Holubová;

People that were helping during the project work: Ole Hejlesen (professor, Department of Health Science and Technology, Aalborg University, Denmark), Georg Sager (professor, Department of Medical Biology, UiT), Thibaud Freyd (PhD, Department of Medical Biology, UiT), Meghan Bradway (NST);

Jan Fuglesteg, Svein Tore Jensen, and all people at the Department of Computer Science (IFI) and Faculty of Science and Technology, UiT The Arctic University of Norway; (IFI, UiT); Norwegian-Russian Secondary School and Kongsbakken Videregående Skole in Tromsø (Norway); Gymnasium #1 in Murmansk (Russia);

Employees of the Norwegian Center for Integrated Care and Telemedicine (NST);

Employees of the International Office, Department of Academic Affairs, UiT;

Friends and mates: Thomas Hibelot, Jean-Baptiste Koehl, Liudmila Nikanorova, Anna Kuznetsova, Liubov Kirillina, Valentin Grigoryev, Mira Grigoryeva, Varvara Alexeeva, Almudena Valdivia Castaño, Raquel Fernandez Moras, Nicola Maranzano, Anna Zini, Anton Uhanov, Maxim Tsaryov, Boris Gladkov, Tolga Başhan, Thibaud Freyd, Marina Gafarova,

Tatiana Tretiakova, Elizaveta Koshel, Maria Mironova, Anna Pryadunenko, Alexey Adikov, Mikhail Enikeev, Bo Søndergaard Olesen, Bjørnar Prytz;

All UiT International Students and ISU Tromsø, during the period 2010-2015;

Everyone whom I have met during 1998-2015 who has left positive in my life;

And many others.

Tromsø, July 2015

Aleksandr Agafonov

Abstract

Motivation, Goal:

Diabetes is one of the most common diseases and one of the leading causes of death. Diabetes type 1, which is also called insulin-dependent diabetes, is the type of diabetes that cannot be treated and makes the life of a person with diabetes type 1 depend on insulin injections. The disease lowers the quality of life, since the person has to manage his/her diabetes all the time, control the amount of carbohydrates in meals and calculate the insulin dosage correctly, in order to stay alive and feel good. However, good management of one's blood glucose levels makes living with diabetes much easier.

There are several groups of diabetes-related software. One of them is serious games, which are aimed to improve the patient's knowledge about the disease and skills in diabetes self-management. Another group is mobile technology (which is a promising technology that has been rapidly growing during the last years) and mobile applications, which can be for example self-management tools, serious games or simulators. Another group of diabetes-related software is clinical decision support systems used by specialists in order to provide better blood glucose management and advices to patients.

Some diabetes-related tools try to reflect the blood glucose behavior and have their own implementation. What if software engine that calculates, mimics, and predict blood glucose levels depending on input data and various parameters, existed? Diabetes-related serious games, systems and applications would be able to use this engine as the blood glucose simulation heart of the game/system/application, and focus on other important aspects, such as usability, reliability, quality of their system or application, etc.

The goal of this project was to conduct research and develop a prototype of the described software engine. In addition to the engine, the goal was also to create a demonstrator that would show all engine's features and functionality.

Methods:

In order to conduct project research and implement project prototype, several methods were used. The search and the review of literature and information sources, as well as multiple meetings and discussions with supervisors and specialists (both face-to-face and in a distance), have been done during the whole period of project work.

As mentioned earlier, the prototype developed during the project work consists of two parts – the software engine itself, and a mobile-phone base demonstrator application. Before starting the implementation of the engine using Java programming language and Eclipse development framework, its architecture was created as a paper draft and later as a digital document. Before starting the implementation of an Android-based demonstrator application with help of Eclipse and Android Development Platform, its design was created as digital mockups. During the development, the author has been constantly receiving useful feedback messages from testers.

Prototype testing was done in the same time with the last stages of the prototype development. Several types of testing were defined. The testers tested the prototype and provided their personal diabetes self-management data, which made me able to do the most difficult part of testing by myself – the prototype accuracy testing.

Testing Results:

Due to the time limitation and the need of diabetes self-management personal data for testing purposes, not many people were involved in testing. The testing results were presented and discussed. The accuracy testing showed good results with room for improvement. The testing results and some other aspects of the project research, development and testing, were discussed after presenting the results. The project requires much more time for continuous and comprehensive testing with people with diabetes of different gender and age, due to its complexity and experimentalism.

Further Work, Conclusion:

During the project research, a prototype consisting of the Diabetes Automata software engine and the demonstrator, was designed, implemented, tested and discussed. The prototype meets the idea and the goals of the project. However, there are several future works to be done in order to improve the engine's accuracy and functionality, and the usability of the demonstrator. Among the further works are comprehensive and long-term testing, a deeper research in order to improve the implemented algorithms, as well as to implement new important features and functionality. Some of the further works were planned during the development but not implemented due to the time limitation, another works were considered by the author or suggested by testers.

Diabetes Automata engine has great potential to be the blood glucose simulation heart of various diabetes-related systems and applications on different platforms.

Contents

List Of Figures

List Of Tables

1. Introduction

1.1. Motivation, Background

Diabetes mellitus is a group of metabolic diseases characterized by high blood sugar levels that result from defects in insulin secretion, or its action, or both (MedicineNet, 2014c). It is a "chronic disease that occurs either when the pancreas does not produce enough insulin or when the body cannot effectively use the insulin it produces" (World Health Organization, 2015).

347 million people worldwide have diabetes. In 2004, an estimated 3.4 million people died from consequences of high fasting blood sugar. In 2014, 9% of adults had diabetes (World Health Organization, 2015). In 2012, the disease was the direct cause of 1.5 million deaths in the world; World Health Organization expects diabetes to become the 7th leading cause of death in 2030. Healthy diet, regular physical activity, and maintaining a normal body weight can prevent or delay the onset of type 2 diabetes (World Health Organization, 2015). See section 2.1 for more information about diabetes.

Good management of one's blood glucose levels makes living with diabetes much easier. Most people with diabetes type 1 (T1D) use a blood glucose meter for blood glucose monitoring and insulin pen or pump for insulin injections. Through frequent measurements and injections of different types of insulin, people with diabetes can generally maintain a blood glucose level within the recommended range. For adolescents with T1D, managing this important element can be a challenge. Measuring blood glucose and taking insulin can be perceived by adolescents as stigmatizing activities. Sometimes patients forget to administer insulin injections or underestimate the effect of the amount of carbohydrates they have eaten. However, neglecting the disease leads to serious long-term consequences. Therefore, adolescents must be made aware of the long-term consequences of poor blood glucose management and be motivated to adequately manage their diabetes – for example, using their own media channels.

One example of such channel for achieving this is serious games. Serious games have proved to be a great leaning technique. They are not just games – they have an implicit objective, which is increasing knowledge, experience and skills; they are motivated by a need to educate, train or inform regarding particular problem. The big advantage that they have is the ability to balance interactivity, entertainment and re-playability. See section 2.2 for more information about serious games. There are many health- and medicine-related serious games, including diabetes-related serious games. The information about some of them can be found in section 2.4.2.

Another channel is mobile technology, which is a promising technology that has been rapidly growing during the last years, together with number of applications for mobile phones, including mobile health (mHealth) applications in general and applications for self-management of diabetes in particular (Chomutare et al., 2011). However, the situation with good solutions and accuracy is not as promising as we would think. Good example is that researchers found that 67 percent of mobile insulin dose calculator applications have risk of generating inappropriate dose recommendations (Huckvale et al., 2015; Pai, 2015).

Decision support systems is a contribution to clinicians. They use these systems in order to provide better blood glucose management and advices to patients. An example of such

system is DiasNet (Hejlesen et al., 2000; Plougmann et al., 2001), for more examples see section 2.5.

In these three presented examples, the applications/systems have own algorithms that simulate or predict one or several key-aspects in diabetes – the algorithms that play the key-role in these applications and systems, and the quality of services that they provide depends on the quality and accuracy of these algorithms, which can be called the heart of the application or system. But what if all diabetes-related systems and application could be based on an already implemented software core that makes necessary calculations, uses biometric information about patient, simulates the behavior of the key-aspects in diabetes, and outputs blood glucose levels, which could be further used by the applications?

Diabetes Automata project addresses this challenge – the project is aimed to develop a software engine prototype that calculates and provides the blood glucose level based upon relevant patient-gathered background data such as insulin, carbohydrates, physical activity, along with the user's biometry such as age, gender, height and weight; which could be used in various diabetes-related programs and applications based on different technologies, including mobile technology.

1.2. Research Problem Statement & Goals

The project addresses the problem and challenge mentioned above, and is aimed to develop an important component that can be the heart of diabetes-related systems and applications.

The research problem is formulated as follows:

Is it possible to develop a software engine that can be used by external applications and is able to simulate blood glucose behavior and calculate blood glucose levels in people with diabetes type 1 with help of input information such as user's biometry and information about insulin, carbohydrates and physical activity?

The goal of this project is to answer this question, and develop a prototype of the software engine, which could be used in diabetes-related serious games and various systems and applications like blood glucose simulators, which are aimed to calculate, mimic, or predict blood glucose levels depending on input data and various parameters. In addition to the engine, the goal was also to create a demonstrator that would show all engine's features and functionality.

The sub-goals can be formulated as follows:

1. To identify a set of methods (such as equations, formulas, graphs) to base the future software model on.
2. To create the architecture for the future system, and the design for the demonstrator UI.
3. To implement prototype of the engine based on the created architecture and the set of collected methods; and prototype of UI that is using the engine and demonstrating its features and functionality; in the way that it could be used on different platforms, including mobile phone applications, computer applications, web applications, etc., to make the usage opportunities of the engine as wide as possible.
4. To test the engine's accuracy.

1.3. Assumptions and Limitations

The project is focused on diabetes type 1 (however, in the future the project can be implemented with focus on diabetes type 2 as well).

Initially, the project supposed to be a part of diabetes-related game that has been developed for the project “Spill og Lær med Diabetesvenner” by NST. This could provide opportunities for a great comprehensive testing in practice. However, in the middle of the development of Diabetes Automata, plans for “Spill og Lær med Diabetesvenner” project had been changed, Diabetes Automata became no longer a part of it, and the opportunities for the comprehensive testing disappeared.

The alternative testing methods were found. Unfortunately, not many potential testers had time for testing and wanted to provide personal data – diabetes self-management history that was required for the alternative testing methods.

Two test persons were found and have been taking part in project testing. One assumption for testing was that the personal data is not transferred elsewhere and is not accessed by anyone except the test persons and the author. Another assumption was that the testers have Android 5.0 mobile platform on their mobile phones, and Diabetesdagboka (Nasjonalt Senter For Samhandling Og Telemedisin, 2015a; Årsand et al., 2010; Årsand et al., 2012) v1.5.3 (ee7a27fb) with inter-process communication (IPC) module.

The time given for the implementation of this master project and conducting the research, was the strictest limitation. Due to the complexity of the project and the research, there is an important part work that was not possible to be done during the timeframe of a master-project. For example, long and extensive testing, as well as deeper research, improvement and calibration of the implemented algorithms, and implementation of new important features and functionality. These are considered to be done in the future. More detailed information about the further work can be found in chapter 8.

1.4. Methods

The methods applied during the project research are as follows.

- Before the start of the project development, and during the whole period of the development, the source and literature search using defined key-words using available search tools was done, in order to gain knowledge and understanding about diabetes, physiological processes, clinical pharmacology – the theoretical information required for the implementation of the project; understand how the engine could be developed and what to base the future algorithms on. The work with literature has been done during the development period as well, in order to improve or change the implemented algorithms or implement new algorithms.
- Technologies, languages and tools used during the development of the prototype, such as Java programming language, Eclipse IDE for Java Developers Version, Android Developer Tools plugin, Android 4.4 and 5.0 mobile operating system.
- Organization of the process of implementation and development of the master-project: work on the project at NST, where it was easy to contact and ask people; direct contact, meetings and frequent face-to-face discussions with supervisors; meetings and discussions with specialists; feedback from the supervisors and test persons.

- Methods used for testing of the implemented prototype with the test persons that were found and involved, and analyzing the testing results. Three testing methods used are: trial and feedback about a new version of prototype; simulation process testing by the test persons, which was telling how the prototype should be possibly calibrated or improved; and sample database testing, during which the author was testing the accuracy of the prototype and analyzing the results, using sample database with diabetes self-management information from the test persons, as well as the additional information that important for testing but could not be recorded in the database.

The full information about the listed methods can be found in sections 3.1 – 3.5. The critique of used methods can be found in section 3.6.

1.5. Significance

In the beginning, the scope of the project was addressed to serious games, and the Diabetes Automata engine supposed to be a component of diabetes-related serious games. During the project development, the scope of the project was revised and extended – from a game-oriented software engine to an engine that could be used in diabetes-related applications and systems, including serious games.

During the search of theoretical information about diabetes, things that affect blood glucose, diabetes self-management, simulation of blood glucose behavior, and making the research of state-of-the-art in order to find out what has been done in the field of blood glucose prediction/simulation and how it was done, the architecture of the prototype of the software engine and the design of the user interface (UI) was created; and the prototype was developed. It was tested with 2 persons that have diabetes type 1, with co-supervisor Eirik Årsand being test person 1. When testing with Eirik was done, it showed quite good results, which are presented in chapter 7. However, the project should be comprehensively tested with people of different gender, age and body types, which is one of the further works presented in chapter 8.

Diabetes Automata is a try of concept in the complex research-field of blood glucose simulation and prediction in experimental medical informatics, an experimental project in software engineering combined with experimental health science. The project integrates together topics such as software system design and development, object-oriented programming, mobile application development, experimental health informatics and electronic health (eHealth) in general and mHealth in particular, and some part of medical biology combined with mathematics.

The project is aimed to develop a prototype of a very important component for diabetes games and simulators. The Diabetes Automata engine is about to open new opportunities for future diabetes-related software. It has potential to be used in diabetes-related serious games and make the game-developers focus more for example on interactivity and gameplay. It has potential to be used in simulators and other diabetes-related programs and applications on various platforms and make the software-developers focus more for example on the usability and design of their programs.

It is a contribution for developers that are creating diabetes-related games, programs and applications for users, which means that the project is also a contribution for people that have

diabetes, for their helping relatives, for healthcare workers, for people that are interested in diabetes disease and learning about it.

The engine itself is a Java-project that it can be imported and used by other projects. The communication with the engine, the simulation control and the engine's functionality is available for the external applications via the provided Application Programming Interface (API).

Moreover, according to the literature review, this project is the first attempt to develop the software engine that has modular system architecture in order to make it easier for the engine developer (the author) to modify the code and run blood glucose calculations in parallel; and is aimed to be used by the external programs and applications on various platforms, and provide blood glucose level simulation over time on chosen time speed and the simulation personalization and control via the provided engine API.

1.6. Organization

The rest of the thesis is organized as follows:

Chapter 2: Theoretical Framework and State-Of-The-Art.
The chapter tells about Diabetes disease and the factors that affect blood glucose. It also contains an overview about games, serious games, diabetes serious games, and state-of-the-art sub-chapter.

Chapter 3: Methods
The chapter describes the methods used for information search, design, development and implementation, testing and evaluation, and critique of chosen methods.

Chapter 4: Requirements Specification.
The chapter contains the description of the project's functional and non-functional requirements.

Chapter 5: Design
The chapter is about the design of the system and testing.

Chapter 6: Implementation
The chapter describes used tools and the process of the development of the model described in the Chapter 5.

Chapter 7: Testing, Results and Discussion
The chapter presents the process of testing, testing results, and discussion.

Chapter 8: Further Works.
The chapter is about the potential future work – possible improvements of the implemented system, as well as possible new features.

Chapter 9: Concluding Remarks
The concluding chapter of the thesis.

Appendix A: Instructions
Contains two versions of the instructions about how to use the prototype – the short one with pictures (the one that was sent to the test persons) and the long one (technical description of the usage of the prototype)

Appendix B: Article For Conference SHI 2015 (15-17 June)
Contains the conference article that was written and presented on the 13th Scandinavian Health Informatics Conference (SHI 2015) that took place at UiT – The Arctic University of Norway 15-17th June 2015.

Appendix C: Prototype Version History
Draft note that the author has been making during the development, in order to remember what changes have been done from version to version, present these changes to the supervisors and testers, and accumulate the version history for this thesis.
References used in this thesis.

Appendix D: Previous Implementation Of Insulin Module
Description of the old implementation of the Insulin module of the developed prototype, which was changed to the implementation presented in section 6.5.6.

References
References used in this thesis.

2. Theoretical Framework and State-Of-The-Art

This chapter contains theoretical information and background of the thesis. It gives overview with relevant information about diabetes disease and the factors that affect blood glucose, blood glucose metabolism models, formulas and equations for diabetes management, as well as the presentation of the example groups of software where Diabetes Automata engine could be potentially used – diabetes-related serious games and existing diabetes-related projects and systems.

2.1. About Diabetes

In this section, we discuss the information that was important to know and understand before starting the prototype development. The information is about diabetes in general, diabetes type 1 in particular, insulin and other factors that affect blood glucose; and is the general theoretical background that the project and the prototype development are based on.

2.1.1. Diabetes Type 1

Diabetes mellitus is a group of metabolic diseases characterized by high blood sugar levels that result from defects in insulin secretion, or its action, or both. The consequences of diabetes are heart disease, nerve damage, blindness, blood vessels damage, coma (MedicineNet, 2014c; Alberti and Zimmet, 1998).

There are two major types of diabetes – diabetes type 1 and 2. This project is focused on diabetes type 1. Type 1 diabetes is characterized by absolute deficiency of insulin production and requires daily administration of insulin. Type 1 occurs in about 10% of cases. However, the cause of type 1 diabetes is not known and it is not preventable with current knowledge (World Health Organization, 2015). According to the WHO (World Health Organization, 2015), type 2 diabetes results from the body's ineffective use of insulin, i.e. excess body weight and physical inactivity. About 90% of people with diabetes around the world have type 2 diabetes (World Health Organization, 2015).

Insulin is a hormone that regulates blood glucose level and is produced by pancreatic β-cells (Alberti and Zimmet, 1998). In healthy body, pancreas produces insulin to keep blood glucose level in normal range (between 4 and 7.8 mmol/l for healthy people (Diabetes.co.uk, 2015c)) when it rises (for example after eating) (MedicineNet, 2014c). Glucose comes from carbohydrates in the food, and when it gets to the bloodstream, pancreas releases insulin into the blood stream in order to make the body cells able to absorb glucose for energy or for future use (Hess-Fischl, 2015). See Figure 1.

In diabetes type 1, pancreas doesn't produce insulin, the β-cells are destroyed. Without insulin, the body cells can't absorb blood glucose. This causes the glucose level to grow and the risk of hyperglycemia, abnormally high blood glucose level. Therefore, insulin injections are required in order to lower the blood glucose level and survive (Hess-Fischl, 2015; MedicineNet, 2014a). See Figure 2.

Figure 1: Insulin released into bloodstream after eating (healthy body).
Source: (Hess-Fischl, 2015)

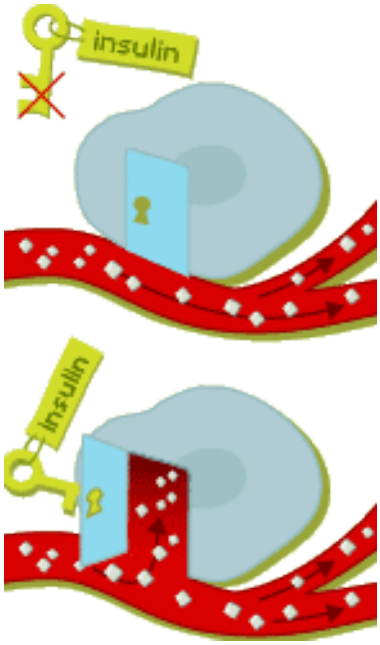


Figure 2: Sugar in the blood of a person with diabetes type 1.
What happens before (upper) and after (lower) insulin is injected.
Source: (Nobelprize.org, 2009)

But if too much insulin is injected, the blood glucose level becomes too low – hypoglycemia (MedicineNet, 2014b). Both hypoglycemia and hyperglycemia are dangerous and can seriously damage health or cause coma (Nobelprize.org, 2009; MedicineNet, 2014c; Alberti and Zimmet, 1998). For people with diabetes, normal blood glucose range is between 4 and 10 mmol/l (Helsedirektoratet, 2009).

In order to self-manage their diabetes, people usually have their personal diabetes diaries (Diabetes.co.uk, 2015b). The contents of the diary depend on the person – the diary can contain records with information about insulin injections, meals, activities, and blood glucose levels after measuring. Such diabetes diaries can be paper-based or electronic. An

example of a self-care diabetes diary is Diabetesdagboka (Nasjonalt Senter For Samhandling Og Telemedisin, 2015a; Årsand et al., 2010; Årsand et al., 2012), a few touch mobile application developed by Norwegian Center for Integrated Care and Telemedicine (NST) (Nasjonalt Senter For Samhandling Og Telemedisin, 2015b). The main screen and the records overview of the application is presented in Figure 3.

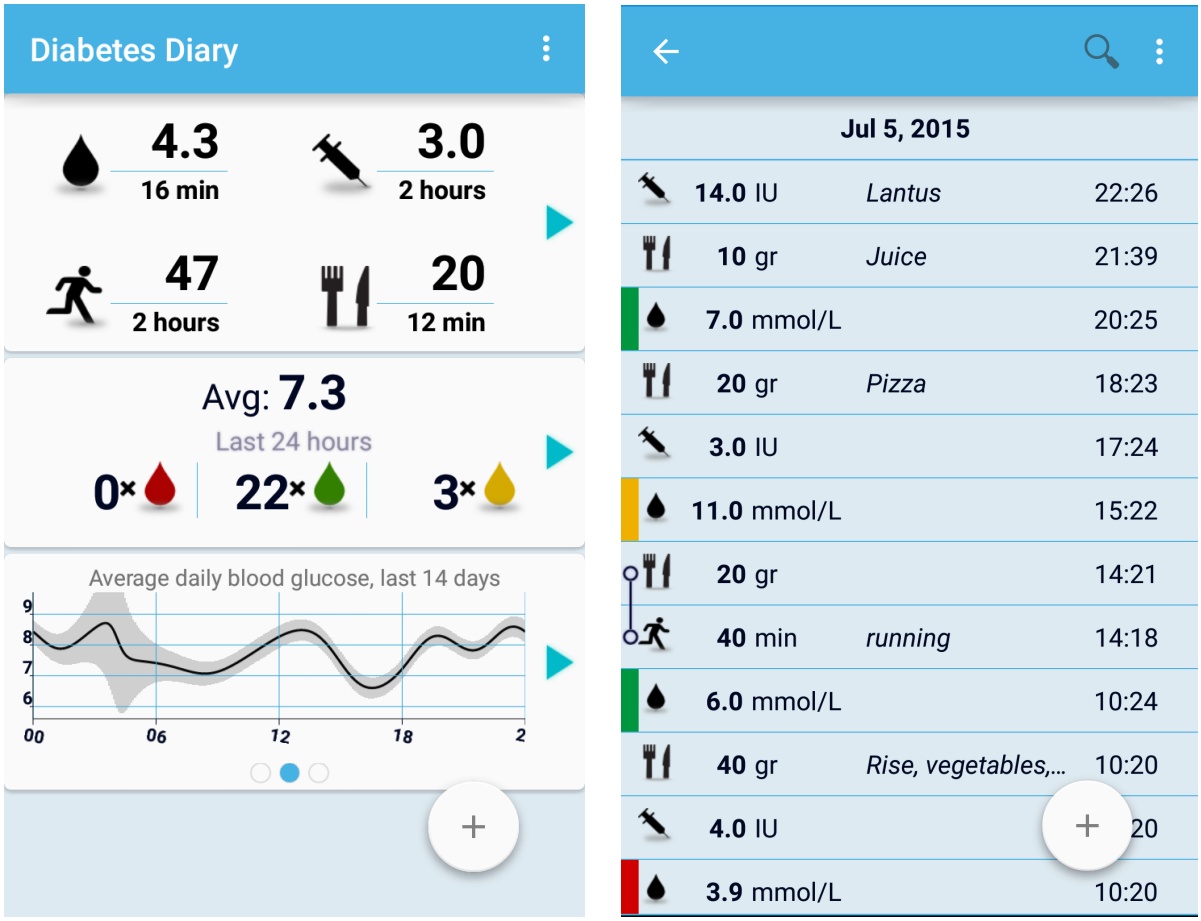


Figure 3: Diabetesdagboka by NST, main screen (left) and records overview (right).

Another reason for presenting this application is because the Diabetes Automata prototype presented in this thesis has integration with Diabetesdagboka. Moreover, Diabetesdagboka is used in testing. For more information, see chapters 6 and 7.

2.1.2. Types Of Insulin And Insulin Activity

There are several major types of insulin: rapid-acting insulin, short-acting insulin, intermediate-acting insulin, and long-acting insulin. (UP Health System Marquette, 2013; Hess-Fischl, 2015; WebMD, 2015; Straight Healthcare, 2014; Diabetes Education Online and UCSF, 2015b; The Diabetes Mall, 2015):

- Rapid-acting insulin is designed to peak and cover meals eaten just after the injection. It is often used with longer-acting insulin. Example: Humalog (Lispro), Novolog (Aspart).
- Short-acting insulin is similar to rapid-acting, but its action is slower. Example: Regular.

- Intermediate-acting insulin is less peaking and has longer action. It covers insulin needs for half the day or overnight. Examples: NPH.
- Long-acting insulin for longest and flat action. It can be called a "background" insulin. It covers about a full day and is often used with rapid-acting or short-acting insulin. Examples: Lantus (insulin glargine), Levemir (insulin detemir).

Rapid-acting and short-acting insulin are called "bolus insulin", and long-acting and intermediate-acting insulin is called "basal insulin" (Diabetes.co.uk, 2015a).

The curves that show the general differences between the actions of different kinds of insulin are represented in Figure 4.

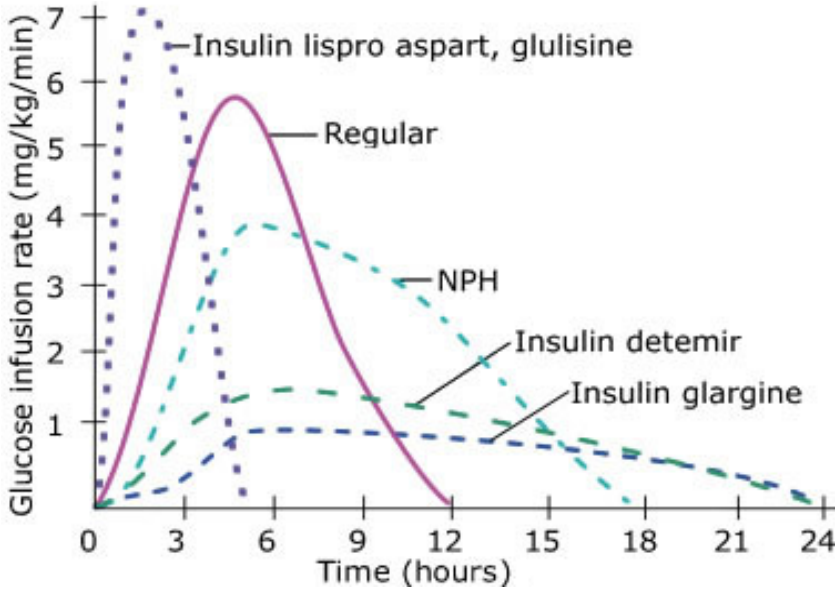


Figure 4: Activity Profiles of different types of insulin.
Rapid-acting insulin Lispro and Aspart. Short-acting insulin Regular. Intermediate-acting insulin NPH. Long-acting insulin Detemir and Glargine.
Source: (Diabetes Education Online and UCSF, 2015b)

The effect over time is represented as glucose infusion rate or glucose utilization rate, and measured in mg/kg/min, where mg is milligrams of glucose, kg per "per kilogram of body weight", and min means "per minute". However, these curves just show the general difference between the behaviors of different types of insulin, and were not used in the development.

The more detailed curves showing the effect over time for each type of insulin, are presented in Figure 5 – Figure 13. These figures are taken from scientific articles and drug documentations, were used in the implementation of algorithms for Humalog, Novolog, Regular, NPH, Levemir and Lantus in the project prototype (see section 6.5.6), and are discussed below.

Different sources provide different information about the peaking effect of insulin in people with diabetes type 1.

In case of Humalog:

Figure 5 (Swan et al., 2009) shows that the maximal effect can be >5 mg/kg/min for 0.2 IU/kg of Humalog.

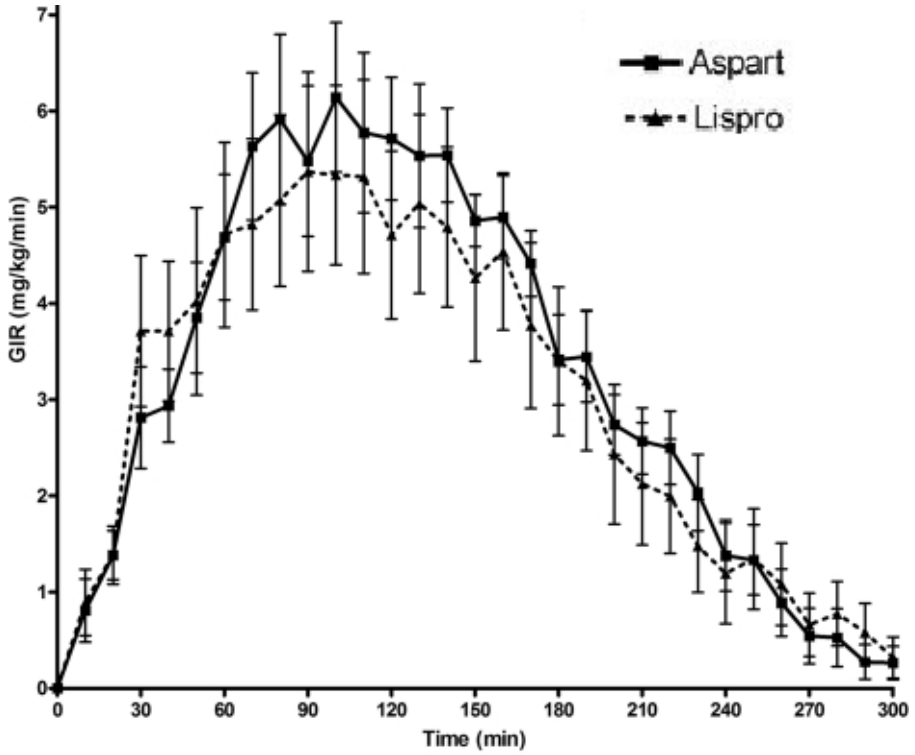


Figure 5: Pharmacodynamic profiles for 0.2 IU/kg of Humalog (Lispro) and Novolog (Aspart). Source: (Swan et al., 2009, Figure 1)

Figure 6 from (Cengiz et al., 2010) shows that the maximal action of 0.2 IU/kg of Humalog used (but not mixed) with 0.4 IU/kg of Lantus is about 6 mg/kg/min.

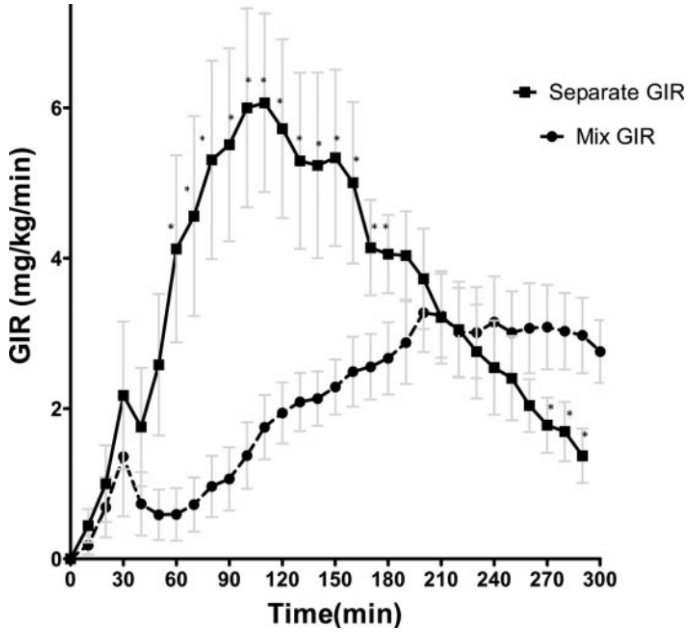


Figure 6: Pharmacodynamic profiles for 0.2 IU/kg of Humalog and 0.4IU/kg of Lantus, mixed and separate usage. Source: (Cengiz et al., 2010, Figure 1).

Figure 7 from the poster-version of (Andersen et al., 2014) shows that the maximal action of 0.2 IU/kg of Humalog is about 7 mg/kg/min.

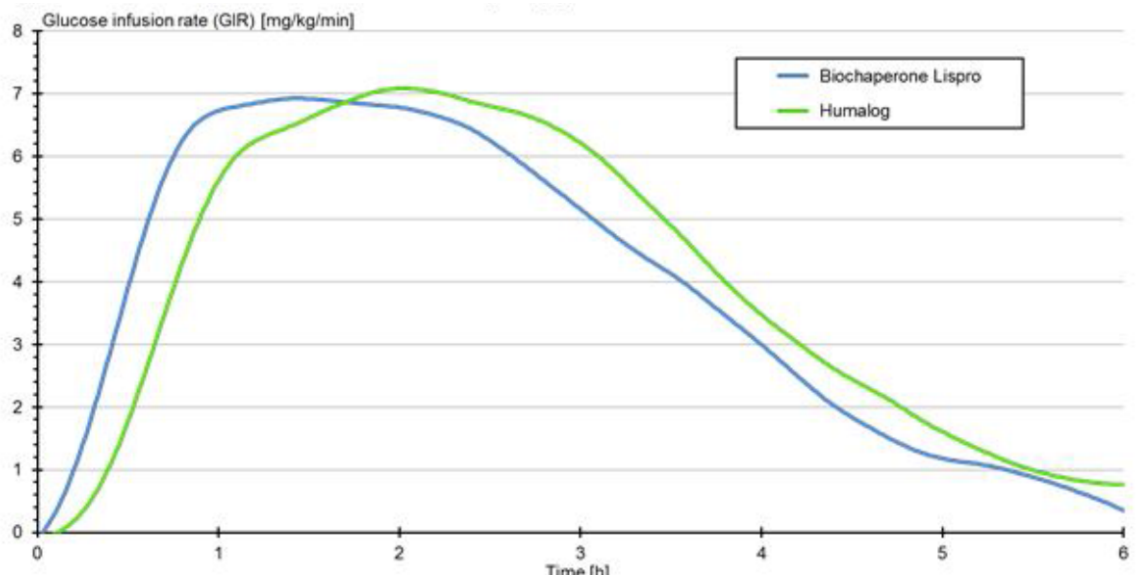


Figure 7: Pharmacodynamic profiles for 0.2 IU/kg of Humalog (Lispro) and 0.2 IU/kg of Ultra-Rapid BioChaperone® Lispro by (Adocia.fr, 2014).
Source: (Andersen et al., 2014, Figure 2).

Figure 8 from (Rave et al., 2001) shows that the maximal action of 0.2 IU/kg of Humalog is about 7 mg/kg/min.

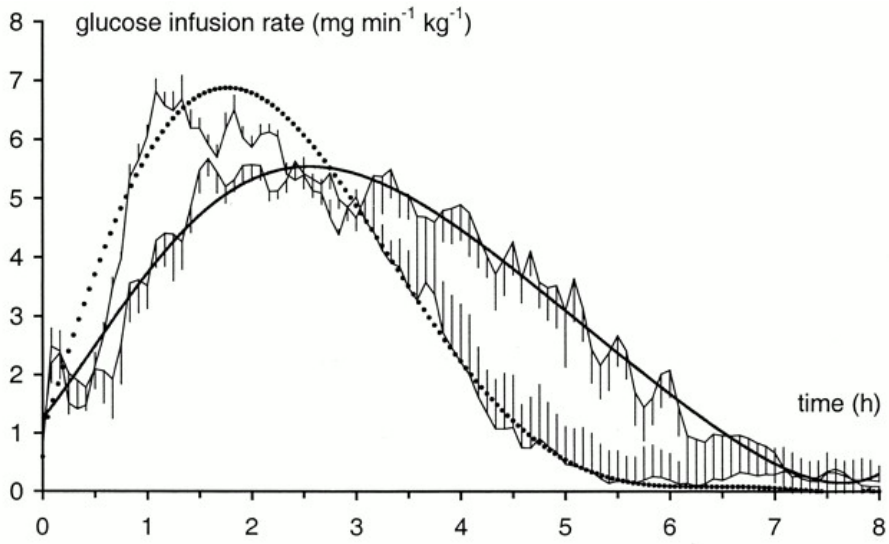


Figure 8: Pharmacodynamic profiles for 0.2 IU/kg of Humalog (Lispro) (dotted line) and 0.2 IU/kg of Regular (heavy black line).
Source: (Rave et al., 2001, Figure 1D).

Figure 9 from (Sanofi, 2013) shows that the maximal action of 0.3 IU/kg of Humalog in obese people is about 5 mg/kg/min.

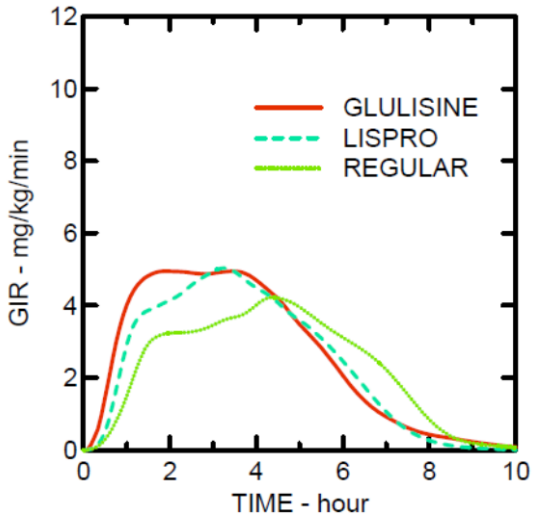


Figure 9: Pharmacodynamic profiles for 0.3 IU/kg of Humalog (Lispro), Regular and Glulisine in obese people.
Source: (Sanofi, 2013, Figure 2).

In case of Novolog:

Figure 5 from (Swan et al., 2009) shows that the maximal effect is between 6 and 7 mg/kg/min for 0.2 IU/kg of Novolog.

Figure 10 from (Heise et al., 2008) shows that the maximal effect is about 8 mg/kg/min for 0.4 IU/kg of Novolog.

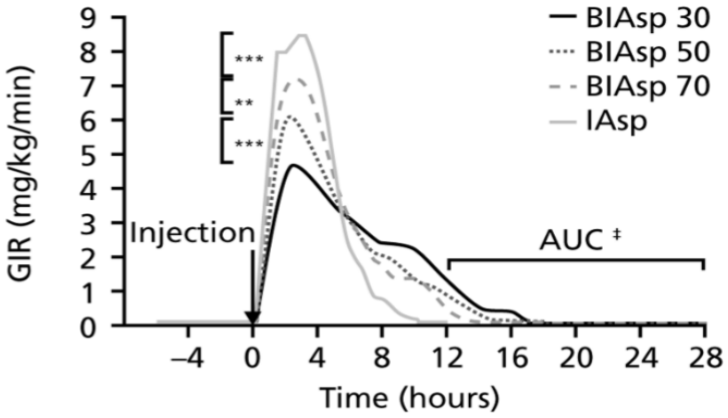


Figure 10: Pharmacodynamic profiles for 0.4 IU/kg of Novolog (Aspart) (grey heavy line).
Source: (Heise et al., 2008, Figure 1A).

In case of Regular:

Figure 8 from (Rave et al., 2001) shows that the maximal action of 0.2 IU/kg of Regular is about 5.7 mg/kg/min.

In case of NPH:

Figure 11 from (Sanofi US, 2015b) shows that the maximal action of 0.3 IU/kg of NPH is 3.4 mg/kg/min. (Sanofi, 2015, Figure 1) and (Lepore et al., 2000, Figure 3) show the same action.

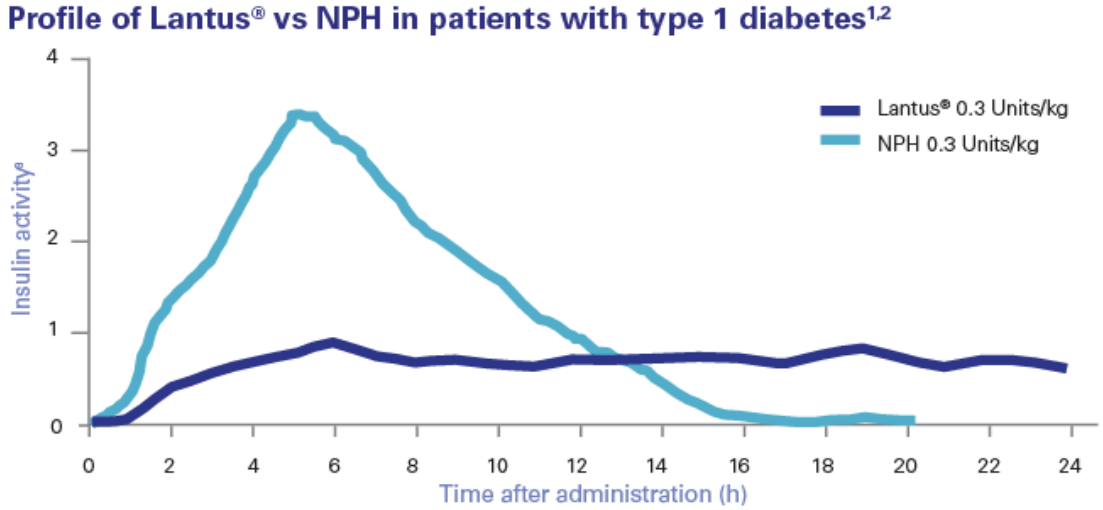


Figure 11: Pharmacodynamic profiles for 0.3 IU/kg of NPH and Lantus.
Source: (Sanofi US, 2015b).

In case of Levemir:

Figure 12 from (Sanofi US, 2015a) shows the action curves and the action duration for 0.2, 0.4, 0.8 and 1.6 IU/kg of Levemir. The same curves (differently represented) are provided in (Plank et al., 2005, Figure 1C). Another curves with peak values close to Figure 12, can be found in (Novo Nordisk, 2015, Figure 2).

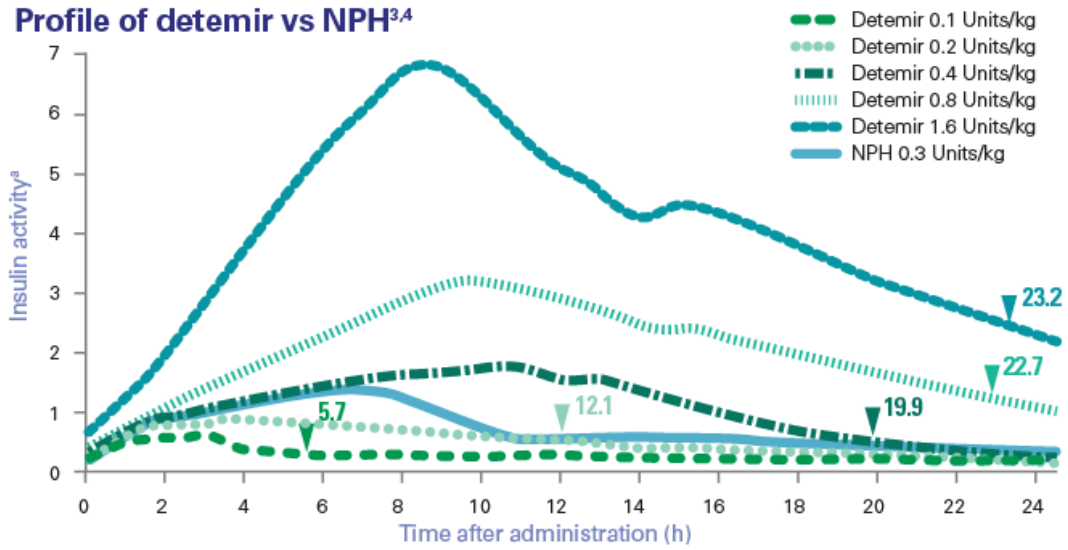


Figure 12: Pharmacodynamic profiles for 0.2, 0.4, 0.8, 1.6 IU/kg of Levemir, with action length.
Source: (Sanofi US, 2015a).

In case of Lantus:

Figure 13 from (Sanofi US, 2015a) shows that the maximal action of 0.35 IU/kg of Lantus is about 1.3 mg/kg/min. Figure 11 from (Sanofi US, 2015b) shows that the maximal action of 0.3 IU/kg of Lantus is about 0.8-0.9 mg/kg/min, and (Sanofi, 2015, Figure 1) and (Lepore et al., 2000, Figure 3) show the same.

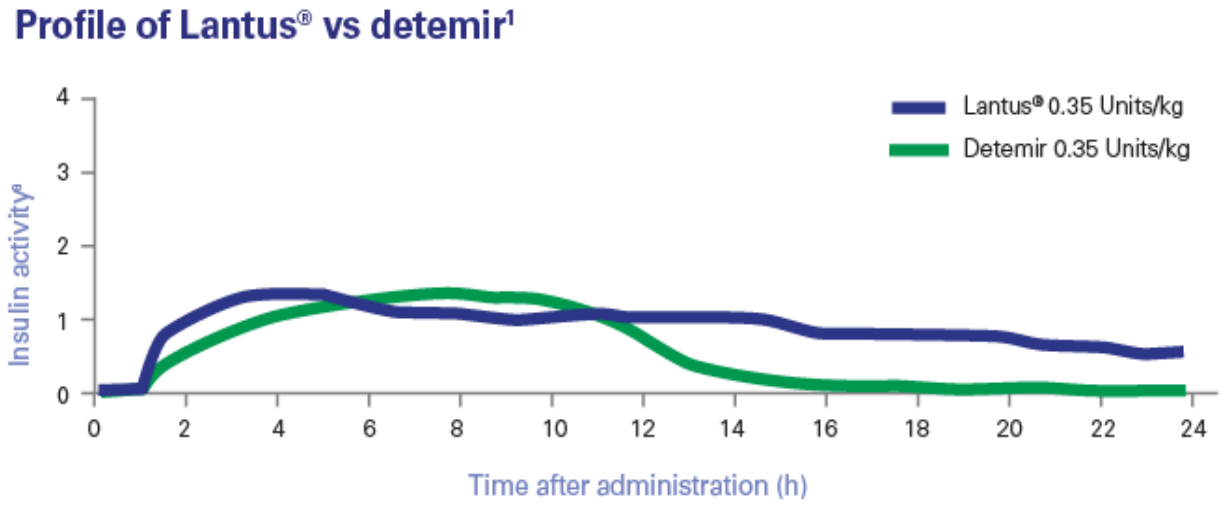


Figure 13: Pharmacodynamic profiles for 0.35 IU/kg of Lantus and Levemir (Detemir). Source: (Sanofi US, 2015a).

Figure 5 – Figure 13 presented above, show the behavior and the action of each type of insulin developed in the system prototype (see section 6.5.6).

Some people with diabetes type 1 use insulin pump instead of long-acting insulin. Insulin pump (American Diabetes Association, 2015) is a small device that delivers insulin via flexible plastic tube and is aimed to mimic the body's normal release of insulin. The amount of the continuous micro-dose delivery can be programed, for different time of the day.

The additional fact to mention is that diseases can affect insulin action. For example, the diagrams and the results presented in (Rave et al., 2001) show that if a person with diabetes type 1 has diabetic nephropathy, the glucose utilization rate is affected and pretty different from what is discussed above.

2.1.3. What Affects Blood Glucose Level

From the section 2.1.1, we already know two factors that affect blood glucose levels in diabetes type 1. They are insulin injections that lower blood glucose level, and carbohydrates in food that raise it.

The information about how much 1 gram of glucose will raise blood glucose level, is presented in Table 1:

Body weight (kg)	16	32	48	64	80	95	111	128	143
1 gram glucose will raise blood glucose by (mmol/L):	1.11	0.56	0.39	0.28	0.22	0.18	0.17	0.14	0.12

Table 1: Rise in blood glucose by 1 g. of glucose depending on body weight. Source: (Bernstein, 2007, Table 20-1).

In addition to the information about carbohydrates, we should add that the blood glucose change can also depend on food glycemic index (GI), which is "a scale used to rank the effect a foods carbohydrates has on raising blood glucose levels after eating" (WhatHealth.com, 2015c). Food with different GI can affect blood glucose level in a different way and during different amount of time: pure glucose (GI 100) or food with high GI (GI >70) affects blood glucose faster (action duration is about 30-60 minutes), and the blood glucose level change is the highest; food with medium GI (GI 56-69) affects blood glucose slower than the food with high GI (action duration is about 45-90 minutes), and the blood glucose level change is the lower than after the food with high GI; and food with low GI (GI <55) affects blood glucose in the slowest way (action duration up to 2 hours), and the blood glucose level change is the lowest in comparison with food with high and medium GI (WhatHealth.com, 2015b; WhatHealth.com, 2015c; GlycemicIndex.com and International GI Database, 2014; Scheiner, 2014; Wilcox, 2010). See Figure 14. The list of GI for common food items can be found in the Glycemic Index List (WhatHealth.com, 2015a).

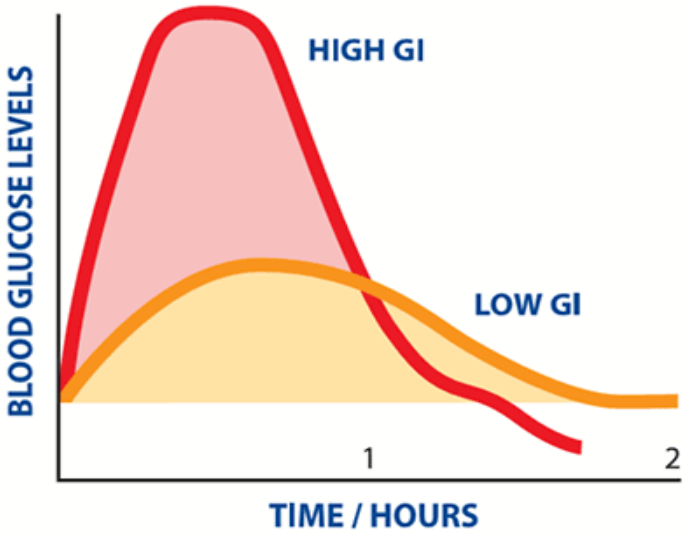


Figure 14: Duration and effect of high GI and low GI, generally. Source: (GlycemicIndex.com and International GI Database, 2014).

The way to predict the blood glucose rise after different types and amounts of food is to calculate Glycemic Load (GL) – it takes into account both the quality and quantity of food,

reflects a specific food glycemic index and amount (WhatHealth.com, 2015d; GlycemicIndex.com, 2014). It can be calculated as follows:

```
GL = GI / 100 * grams of carbohydrate in the eaten food.
```

For example, an apple has GI 40 and contains approximately 15 grams of carbohydrates (GlycemicIndex.com, 2014). GL of apple is 40 / 100 * 15g = 6g. Another example, 50 of pure glucose, which has GI 100, have GL = 100 / 100 * 50g = 50g.

We should not forget that in case of Diabetes type 1 the body doesn't produce insulin. Together with the information about GI, the found sources (GlycemicIndex.com and International GI Database, 2014; WhatHealth.com, 2015c; WhatHealth.com, 2015b) also provide the curves that show approximately how different GIs affect blood glucose in healthy people, not in diabetic. In healthy people insulin is produces by the body when blood glucose is high (see 2.1.1), and the curves show its effect - after peaking, these curves go down. But in people with diabetes type 1, if we assume that there is currently no insulin in the body, the curves would not go down.

The third factor that affect blood glucose levels is physical activity or physical exercise – it can either increase or decrease blood glucose levels. We discuss it in details further in the text.

Another factors are effect of glucagon (when the blood levels are low (Diabetes.co.uk, 2015e) or overnight or between meals, it activates liver glycogenolysis and gluconeogenesis, which results in the release of glucose to the bloodstream by the liver) and its secretion by pancreatic alpha-cells (Hughes and Narendran, 2014; Quesada et al., 2008; Diabetes Education Online and UCSF, 2015a), growth hormone and other hormones (Diabetes Education Online and UCSF, 2015a), and other.

However, in people with diabetes type 1, the secretion of glucagon in the body is reduced during the first years of having the disease, and completely disappears in some years: (The Diabetes Mall, 2014c) says that glucagon secretion is reduced in most people who have Type 1 diabetes within the first two to ten years after onset; (Taborsky, 2010) says that the glucagon response to hypoglycemia caused by insulin is impaired within the first year of T1DM and lost entirely several years later.

Another factor is "dawn phenomenon". All people, diabetic and non-diabetic, can have dawn phenomenon (American Diabetes Association, 2013b), which is a surge of hormones (including growth hormone) that causes blood glucose levels to rise (Perriello et al., 1990; DiabeticTalk and Nutrihand, 2008; Diabetes.co.uk, 2015d; American Diabetes Association, 2013b); it also causes the insulin requirements to become to become 20-30% higher (Perriello et al., 1991). In people with diabetes type 1, glucose production is increased by 65% and glucose utilization is decreased by 50% during the dawn phenomenon (Campbell et al., 1985a). It usually takes place during late night and early morning – 4:00 to 8:00 (DiabeticTalk and Nutrihand, 2008), and in people with diabetes type 1 it takes place between 3:30 and 8:00 (Campbell et al., 1985b). The rise in blood glucose in the morning comparing to the blood glucose levels at bedtime, is 20-100 mg/dl (1.11-5.55 mmol/L), according to (Bernstein, 2007, p. 93); 20-30% rise, according to (Van Cauter et al., 1997, p. 719); 4.6 ± 6.4 mg/dl (0.26 ± 0.355 mmol/L) per hour from 4:00 and 26.6 ± 10.2 mg/dl (1.48 ± 0.566 mmol/L) per hour before 8:00, according to (Freckmann et al., 2008). According to (Spero and Diabetes Self Management, 2013), there are from 25% to 50% of people with diabetes type 1 that have dawn phenomenon.

Another factor is active 24/7-consumption of glucose by brain with no need of insulin. In 2.1.2 we found out that body cells need insulin in order to absorb glucose for energy or to

store it for future use. The brain can't store glucose for future use because it lacks fuel stores (Berg et al., 2002), and moreover, the glucose constantly absorbed by the brain cells via a range of insulin-insensitive glucose transporters (Diapedia, 2014), is the sole fuel for brain, which accounts for ~2% of the body weight and consumes ~20% of glucose-derived energy making it the main consumer of glucose (~5.6mg glucose per 100g human brain tissue per minute) (Mergenthaler et al., 2013). It consumes about 100g (Diapedia, 2014) or 120g (Berg et al., 2002) of glucose per day, which is about 60% of the utilization of glucose by the whole body in the resting state (Berg et al., 2002).

Additional factors are side effects of medication, stress, illness, long-term or short-term pain, alcohol, menstrual periods for women, insufficient sleep or tiredness, nicotine (Diabetes.co.uk, 2015f; American Diabetes Association, 2014; Bornemisza and Suciu, 1980; WebMD, 2011).

Let's take a more detailed look at how exactly different kinds of exercise can affect blood glucose.

Different kinds of activity affect blood glucose level in a different way. Aerobic or cardio activities is a challenging but light-to-moderate activities that require the use of oxygen (active breathing) and raise the heart rate, for example running, swimming, walking, cycling, skiing, games like football, basketball, tennis, and etc. Anaerobic activities or resistance training are high-intensity stressing (short) activities, for example weight lifting, games that "burst" activity like baseball or golf, sprints (during running, swimming, rowing), activities where intensity reaches 80% or more of the maximum capacity activities where winning is the primary objective, and etc. (Scheiner, 2006; Adams, 2013; Diabetes.co.uk, 2015f; American Diabetes Association, 2014; American Diabetes Association, 2013a; Berg, 2010; Turner et al., 2015).

These two groups of activities affect blood glucose level in a different way. When doing aerobic activities, the muscles need to absorb more sugar, and the body becomes more sensitive to insulin. Increased insulin sensitivity causes more blood sugar to be absorbed by the body cells, which means the blood sugar level becoming lower. Depending on the duration and the intensity of the exercise, the increased insulin sensitivity can last between 24 and 72 hours coming after a single exercise session. When doing anaerobic activities, the stressed body produces adrenaline (stress hormone) that causes a short-term blood glucose level rise at the onset of the exercise and immediately after the exercise. The effect remains during 1 to 2-3 hours after exercise for both diabetes type 1 and 2. But in case of diabetes type 1, the blood glucose level gets decreased in 14-20 hours after the exercise. (Scheiner, 2006; Adams, 2013; Diabetes.co.uk, 2015f; American Diabetes Association, 2014; American Diabetes Association, 2013a; Berg, 2010; Turner et al., 2015).

Different sources say different information about how much insulin sensitivity increases after exercising. Aerobic activity increases insulin sensitivity approx. by 30% (Bacchi et al., 2012); or by 20% (Magkos et al., 2010), or by 25% after the first exercise and by 40% after 6 weeks of exercising (Magkos and Sidossis, 2008); or by 23±5% (Landt et al., 1985); or by 25-50% depending on exercise duration (BreakingMuscle.com, 2013); or by 35±14% (lean people) and 59±19% (obese people) (van der Heijden et al., 2009). Anaerobic activity increases insulin sensitivity approx. by 15% (Bacchi et al., 2012).

Different information was found also about how much blood glucose rises after doing anaerobic exercise. The rise in blood glucose is approximately 2 mmol/l (Turner et al., 2014; Turner et al., 2013) or from 2 and up to 4 mmol/l, sometimes even higher (Turner et al., 2015).

2.2. Existing Models

There are several models of glucose kinetics. First, we discuss one of the most famous models – Bergman's Minimal Model. After that, a short overview of other models is presented.

- **Bergman's Minimal Model.**

R.N Bergman and co-workers have developed minimal models for glucose and insulin kinetics in dogs and humans (Bergman et al., 1981; Bergman et al., 1979; Toffolo et al., 1980).

According to (Bergman, 2002), more than 500 minimal-model-related studies can be found in the literature.

The minimal model contains minimal set of parameters, and consists of two models: minimal model of glucose kinetics, and minimal model of insulin kinetics.

The first model consists of 2 differential equations (De Gaetano and Arino, 2000):

$$\frac{dG(t)}{dt} = -[p_1 + X(t)]\,G(t) + p_1 G_b, \qquad G(0) = p_0$$

$$\frac{dX(t)}{dt} = -p_2 X(t) + p_3[I(t) - I_b], \qquad X(0) = 0$$

The second model consists of 1 differential equation (De Gaetano and Arino, 2000):

$$\frac{dI(t)}{dt} = p_4[G(t) - p_5]^+ t - p_6[I(t) - I_b], \qquad I(0) = p_7 + I_b$$

Where:

G(t) [mg/dl] is the blood glucose concentration at time t [min];

I(t) [µUI/ml] is the blood insulin concentration;

X(t) [min^{-1}] is an auxiliary function representing insulin-excitable tissue glucose uptake activity, proportional to insulin concentration in a "distant" compartment;

G_b [mg/dl] is the subject's baseline glycemia;

I_b [µUI/ml] is the subject's baseline insulinemia;

p_0 [mg/dl] is the theoretical glycemia at time 0 after the instantaneous glucose bolus;

p_1 [min^{-1}] is the glucose "mass action" rate constant, i.e. the insulin-independent rate constant of tissue glucose uptake, "glucose effectiveness";

p_2 [min^{-1}] is the rate constant expressing the spontaneous decrease of tissue glucose uptake ability;

p_3 [min^{-2} $(µUI/ml)^{-1}$] is the insulin-dependent increase in tissue glucose uptake ability, per unit of insulin concentration excess over baseline insulin;

p_4 [$(µUI/ml)\ (mg/dl)^{-1}\ min^{-1}$] is the rate of pancreatic release of insulin after the bolus, per minute and per mg/dl of glucose concentration above the "target" glycemia;

p_5 [mg/dl] is the pancreatic "target glycemia";

p_6 [min^{-1}] is the first order decay rate constant for Insulin in plasma;

p_7 [µUI/ml] is the theoretical plasma insulin concentration at time 0, above basal insulinemia, immediately after the glucose bolus.

(De Gaetano and Arino, 2000)

- **Some other models:**
  - *Simple ordinary differential equation model by Bolie (Bolie, 1961):*
    The model was proposed in 1961, and according to (Derouich and Boutayeb, 2002) its author is the pioneer in the field.

$$\frac{dG}{dt} = -a_1 G - a_2 I + p, \qquad \frac{dI}{dt} = -a_3 G - a_4 I$$

    Where "*G = G(t)* represents the glucose concentration, *I = I(t)* represents the insulin and *p; a_1; a_2; a_3; a_4* are the parameters. This model assumes that glucose disappearance is a linear function of both glucose and insulin. The insulin secretion is proportional to glucose and insulin disappears in proportion to the plasma insulin concentration" (Derouich and Boutayeb, 2002).
  - *Bayesian network of blood glucose and exercise (Ewings et al., 2014):*
    A "statistical approach to analysing the effects of everyday physical activity on blood glucose concentration in people with type 1 diabetes. A physiologically-based model of blood glucose dynamics is developed to cope with frequently-sampled data on food, insulin and habitual physical activity; the model is then converted to a Bayesian network to account for measurement error and variability in the physiological processes" (Ewings et al., 2014).
  - *Augmented Minimal Model of glucose metabolism (Markakis et al., 2008):*
    This new model structure can "represent the insulin – glucose dynamics of healthy subjects as well as Type 1 and Type 2 diabetics, with appropriate adjustment in its parameters" (Markakis et al., 2008), and "satisfies these specifications in the following:
      - It models the effects of both insulin and glucagon on blood glucose concentration.
      - It includes time-independent sections for endogenous insulin and glucagon production, in agreement with the approach in [8].
      - It incorporates the effects of exogenous "disturbance" factors on plasma insulin and blood glucose (e.g. exogenous insulin infusions or meals)."

    (Markakis et al., 2008)
    Reference "[8]" in the quotation above is (Van Herpe et al., 2006).
  - *A minimal model for glycemia control in critically ill patients (Van Herpe et al., 2006):*
    Modification of Bergman's minimal model to be used with critically ill patients. Since Bergman's model is "mostly valid in a rather restrictive setting, it might not be suitable to be used in a model predictive controller. Simulations show that the new model exhibits a similar glycemia behaviour but clinically more realistic insulin kinetics. Therefore it is potentially more suitable for glycemia control. The designed model is also estimated on a set of critically ill patients giving promising results." (Van Herpe et al., 2006)
  - *A Nonlinear State Space Model for the Blood Glucose Metabolism of a Diabetic (Briegel and Tresp, 2002).*
  - *Dynamic Modeling of Exercise Effects on Plasma Glucose and Insulin Levels (Roy and Parker, 2007):*
    This is a modification/extension of the Bergman's minimal model – it adds the effects of exercise to the Bergman's model.
    "A minimal model developed previously was extended to include the major effects of exercise on plasma glucose and insulin levels. Differential equations were developed to capture the exercise-induced dynamics of plasma insulin clearance and the elevation of glucose uptake and hepatic glucose production rates. The decreasing liver glucose

output resulting from prolonged exercise was modeled using an equation depending on exercise intensity and duration" (Roy and Parker, 2007).
"The model successfully emulated the physiological effects of exercise on blood glucose and insulin levels" (Roy and Parker, 2007).

- *Model by Derouich and Boutayeb (Derouich and Boutayeb, 2002):*
  This is also a modification/extension of the Bergman's minimal model – it adds the effects of physical exercise to the Bergman's model.
  The authors "introduce the effect of physical activity via parameters of a mathematical model which allows us to compare the behaviour of blood glucose in normal, non-insulin-dependent diabetes and insulin-dependent diabetes people, with and without physical effort. Extreme cases of physical activity leading to hypoglycaemia or aggravating hyperglycaemia are also underlined" (Derouich and Boutayeb, 2002).
- *Computer model for mechanisms underlying ultradian oscillations of insulin and glucose (Sturis et al., 1991):*
  This six-dimensional model "comprises two major negative feedback loops describing the effects of insulin on glucose utilization and glucose production, respectively, and both loops include the stimulatory effect of glucose on insulin secretion" (Sturis et al., 1991).
- *Dynamical model of the glucose-insulin system (De Gaetano and Arino, 2000):*
  A "mathematically more reasonable" (Li et al., 2001) and more realistic delay integro-differential equation model that the authors introduced together with formal mathematical analysis on the Bergman's minimal model (Makroglou et al., 2005).
- *A model by (Li et al., 2001):*
  A more generic model than the previous one (Makroglou et al., 2005).
  The authors "generalize the dynamical model to allow more general functions and an alternative way of incorporating time delay" (Li et al., 2001).

And other models.
More detailed overview of some models can be found for example in (Makroglou et al., 2005) or (Boutayeb and Chetouani, 2006).

2.3. Formulas, Rules, Equations

This section presents some simple formulas, which either are or were used in the development of the prototype (see section 6.5 for more information about how some of the formulas were used).

- **Total body water (TBW) formulas.**
  There are several formulas for TBW calculation.

  - One of such formulas is Watson formula (Watson et al., 1980). The researchers collected data from 458 adult males and 265 adult females together with their height, weight and age, and developed TBW formula.

    For adult males, the formula is:

```
TBW (liters) =
2.447 - 0.09156 * age (years) + 0.1074 * height (cm) +
0.3362 * weight (kg)
```

For adult females, the formula is:

```
TBW (liters) =
-2.097 + 0.1069 x height (cm) + 0.2466 * weight (kg)
```

❖ Another formula is Hume-Weyers formula (Hume and Weyers, 1971). The researchers collected data from 30 adult males and 30 adult females together with their height, weight and other information.

For adult males, TBW was calculated as:

```
TBW (liters) =
0.194786 * height (cm) + 0.296785 * weight (kg) -
14.012934
```

For adult females, TBW was calculated as:

```
TBW (liters) =
0.34454 * height (cm) + 0.183809 * weight (kg) - 35.270121
```

❖ Another formula is Mellits-Cheek formula for kids (Mellits and Cheek, 1970). The researchers collected data from 168 males and 83 females of different age, and developed TWB formula.

For males with height < 132.7 cm, the formula is:

```
TBW (liters) =
-1.927 + 0.465 x weight (kg) + 0.045 x height (cm)
```

For males with height > 132.7 cm, the formula is:

```
TBW (liters) =
-21.993 + 0.406 x weight (kg) + 0.209 x height (cm)
```

For females with height < 110.8 cm, the formula is:

```
TBW (liters) =
0.076 + 0.507 x weight (kg) + 0.013 x height (cm)
```

For females with height > 110.8 cm, the formula is:

```
TBW (liters) =
-10.313 + 0.252 x weight (kg) + 0.154 x height (cm)
```

- **Nadler's blood volume formula (Nadler et al., 1962).**
  This formula was tested with 155 men and women and is used to calculate the blood volume in liters. The formula for men is:

  $$\text{Volume (L)} = (0.3669 * \text{height (m)})^3 + 0.03219 * \text{weight (kg)} + 0.6041$$

  The formula for women is:

  $$\text{Volume (L)} = (0.3561 * \text{height (m)})^3 + 0.03308 * \text{weight (kg)} + 0.1833$$

- **Carb Factor: The 500 Rule (The Diabetes Mall, 2014a).**
  This rule is used to estimate the number of grams of carbohydrates per unit of insulin. The formula is:

  $$\text{Number of grams of carbohydrates per insulin unit} = 500 / \text{Total daily dose of insulin (units)}.$$

  However, the authors of (Kuroda et al., 2012) say that the formula is not the best for patients that use insulin pump and 300 (breakfast) / 400 (lunch and supper) should be used instead of 500.

  The formula becomes (breakfast):

  $$\text{Number of grams of carbohydrates per insulin unit} = 300 / \text{Total daily dose of insulin (units)}.$$

  And (lunch/supper):

  $$\text{Number of grams of carbohydrates per insulin unit} = 400 / \text{Total daily dose of insulin (units)}.$$

- **Correction Factor: The 1800 rule for mg/dl (The Diabetes Mall, 2014b) and the 100 rule for mmol/l (University College London Hospital NHS, 2013).**
  Both rules estimate the drop of blood glucose level per unit of insulin. The difference between them is the units used for blood glucose levels – mg/dl (milligrams per 100 milliliters) and mmol/l (millimoles per liter).
  (1 mmol/l = 18 mg/dl (Diabetapedia, 2014)).

  The 1800 rule is used when blood glucose level is measured in mg/dl. The formula is:

  $$\text{Number of mg/dl} = 1800 / \text{Total daily dose of insulin (units)}.$$

  The 100 rule is used when blood glucose level is measured in mmol/l. The formula is:

  $$\text{Number of mmol/l} = 100 / \text{Total daily dose of insulin (units)}.$$

In Norway (since 1977) blood glucose unit of measure is mmol/l (Diabetesforbundet, 2008). The full list of which country uses which units, can be found here (Abbott Diabetes Care, 2005).

- **The 0.12 Formula for the Management of Hypoglycemia and Hyperglycemia in Children with Type 1 Diabetes (Yazbeck et al., 2013a; Yazbeck et al., 2013b).**
  How it works:

  Child's total body water is about 65%.

  ```
  1 mmol/l of glucose = 18 mg/dl = 0.18 g/l.
  0.18 * 0.65 = 0.12.
  ```

  Glucose sensitivity gives the number of grams of carbohydrates that will increase blood glucose level by 1 mmol/l.

  ```
  Glucose sensitivity (mmol/l) = 0.12 * Weight (kg)
  ```

  Insulin sensitivity gives the number of mmol/l that are dropped by 1 unit of insulin.

  ```
  Insulin sensitivity = Insulin/Carbo ratio / Glucose sensitivity.
  ```

  Instead of using The 500 Rule to find Insulin/Carbohydrate ratio, the authors use The 300 Formula:

  ```
  Number of grams of carbohydrates per insulin unit =
  300 / Total daily dose of insulin (units),
  ```

  assuming that the average child consumes about 300 grams of carbohydrates per day.

2.4. Diabetes-Related Serious Games

Initially, the motivation behind the Diabetes Automata project was to create serious game oriented software engine for blood glucose levels simulation. The engine developed in this project, supposed to be used in a diabetes-related game (see section 3.6 for more information) with avatars having diabetes, and was aimed to make the avatars behave as realistic as possible. But what is a serious game? What are the examples of existing diabetes-related games? And how exactly the Diabetes Automata engine could be used in diabetes-related games? This section provides answers to these questions.

2.4.1. Background

A game is defined as a contest with some objective, which has a set of defined rules and is specified by gameplay, challenge, interaction and objective (Wattanasoontorn et al., 2013). Usual games contain only explicit objective, which is making it fun and interesting to play.

However, serious games also have an implicit objective, which is increasing knowledge, experience and skills (Wattanasoontorn et al., 2013). See Figure 15. In other words, the use of serious games actually is “game-based learning”, which describes the application of games for teaching competences and skills in a selected area of knowledge or the informal learning while playing a game (Michel, 2013).

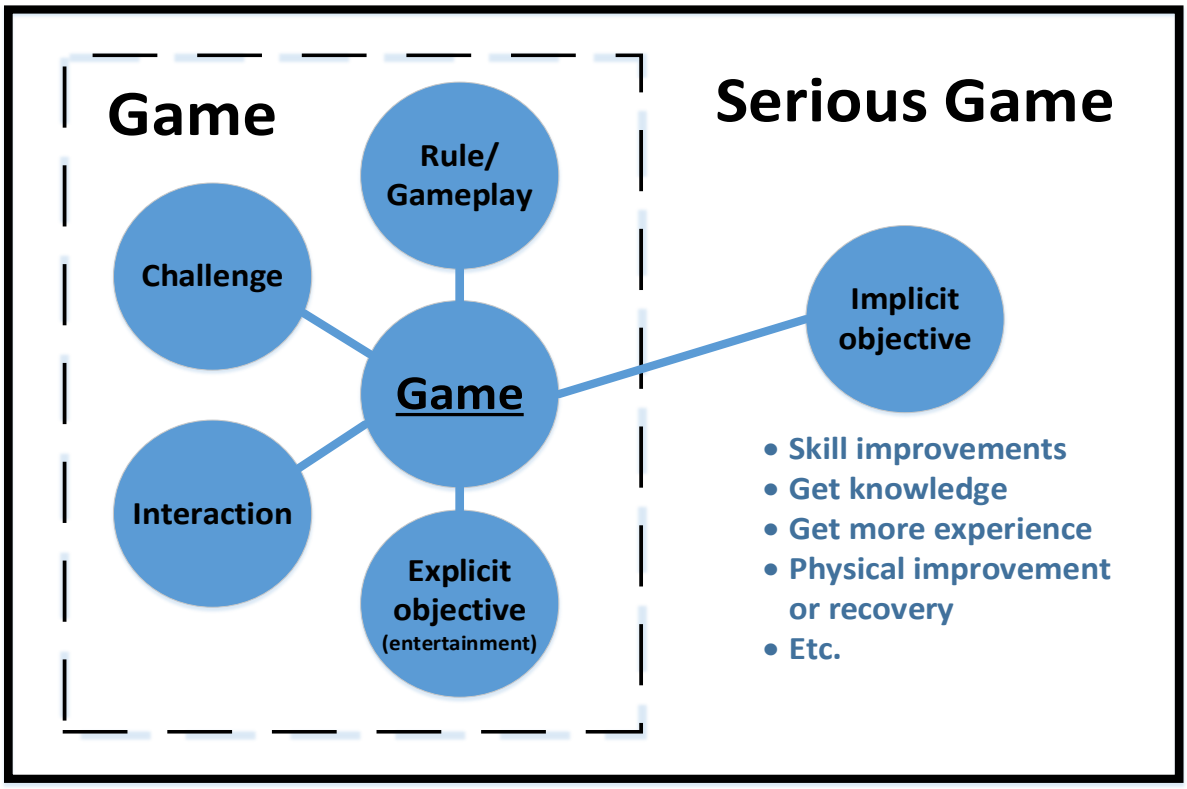


Figure 15: Games and Serious Games.
Based on: (Wattanasoontorn et al., 2013, Figure 2).

In addition, story, feedback mechanisms, points and transfer to real-life settings, are the other issues of a serious game (Baranowski et al., 2013).

Serious games are motivated by a need to educate, train or inform regarding particular problem. The big advantage that they have - the ability to balance interactivity, entertainment and re-playability (Pereira et al., 2012) - makes them a very effective educating and helping tool for gaining skills, knowledge and experience in different branches, for example in military, education, and of course in the branch that this paper is dedicated to - healthcare and medicine (Wattanasoontorn et al., 2013).

There are three main reasons for the ever-increasing the use of serious games in education (Wrzesien and Raya, 2010):

1) they use actions rather than explanations and create personal motivation and satisfaction;

2) they accommodate multiple learning styles and abilities;

3) they foster decision-making and problem-solving activities in a virtual setting

In healthcare, serious games are emerging as a new pedagogical approach (Lancaster, 2014) that, if we compare it with classical way of educating, has brought three big changes discussed in (Guill and Aleson-Carbonell, 2012). They are:

1) the shift from a teacher centered approach to a learner centered approach;

2) the shift from a model of instruction based on listening to a model of instruction based on doing and interaction;

3) the shift from a concept of learning based on memory to a concept of learning based on the capacity to find and use information.

Nowadays there are various serious games with focus on health care, physical and mental fitness (Fuchslocher et al., 2011):

- For patients with different diseases or problems: diabetes, elderly people, cancer, flu, respiratory illness, stroke, autism, and many others. The main objective is to increase motivation in learning about their condition and the way to treat, using games as a tool against anxiety (for example for pre-surgery) and pain (as an example, distraction, by means of engaging burn-injured patients in a virtual game activity, has showed to significantly reduce perceived pain during bandage change (Sharar et al., 2007)), and continuing the treatment as long as needed (Carolyn, 2006).
- For usual people (not patients): HIV, relation and sex education, aggression and emotion control, anti-smoking, anti-narcotic, and many others. The main objective is either to increase motivation for healthy way of life and think more about being careful with personal health, or to get rid of a starting disease or problem or an ongoing unhealthy dependency.
- Moreover, health related games can go even beyond the educational phase, for example, games for long term support of people with chronic conditions (Carolyn, 2006).
- Serious games can be used not only for patients or usual people, but also for healthcare personnel, for example games for training nursing skills, or surgeon training, patient care training (Wattanasoontorn et al., 2013), etc., in order to provide high-level professional help and avoid medical errors.

The paper (Wattanasoontorn et al., 2013) tells that considering the subjects related to a serious game for health, which are “serious game”, “health”, and “player”, the games can be classified by:

1) stage of disease;
2) game purpose;
3) functionality;
4) players wellness.

The first classification divides games into the games focused on:

a) entertainment, b) health, and c) health acquisition and medical skills.

The second classification makes the games sub-classified by:

a) application area, b) interaction technology, c) game interface, d) number of players, e) game genre, f) game hardware, g) game engine, h) platform, i) whether the game is able to communicate via internet or not; and some other.

The third classification subdivides the games into games for patients and non-patients. Games for patients can be for:

a) health-monitoring, b) detection, c) treatment, d) rehabilitation, and e) education. Games for non-patients can be for a) health and wellness, b) training for professionals, c) training for non-professionals.

Finally, the forth classification sub-classifies into four different stages:
a) susceptibility, b) pre-symptomatic, c) clinical disease, and d) recovery, disability stage.

For having a visual view over this healthcare serious games classification model, we can take a look at Figure 16, which represents all classifications presented above.

Serious games for healthcare cover a huge area in the intersection of informatics (especially game/application development) and healthcare. Serious games in general have a great potential, combining the advantages of games and purposes of education. Some researchers are convinced that SGs are the educational tools of the future (Girard et al., 2013).

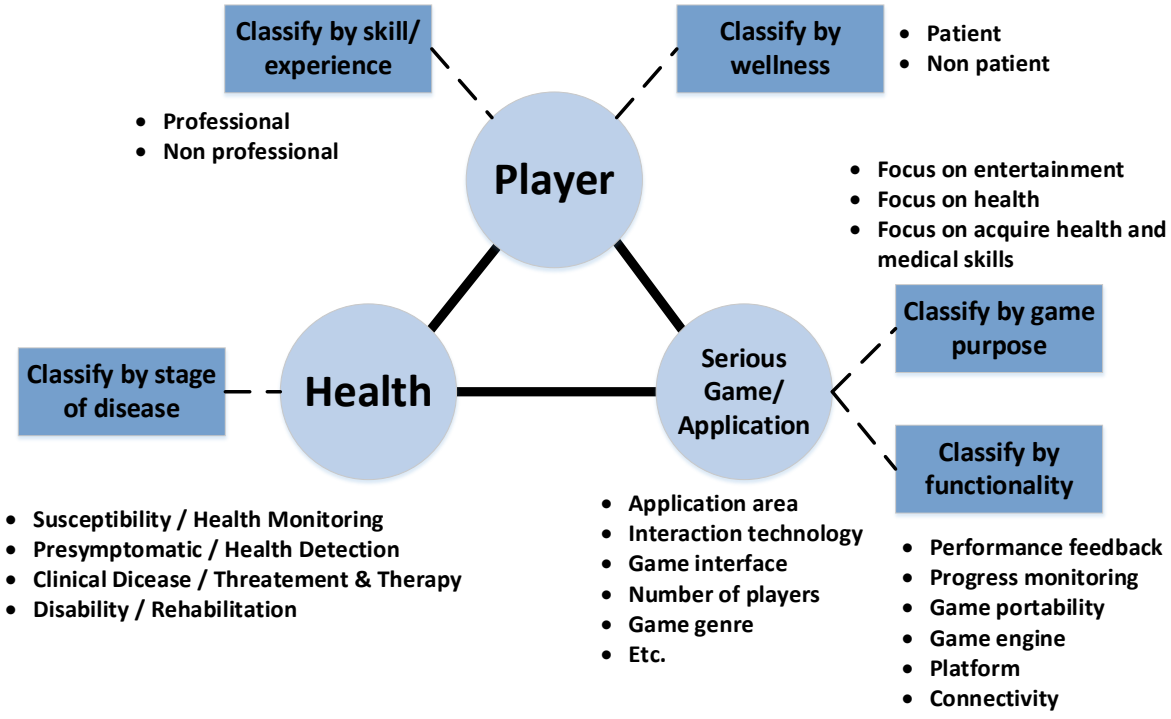


Figure 16: Classification of healthcare serious games.
Based on: (Wattanasoontorn et al., 2013, Figure 5)

Where in the presented classifications can Diabetes Automata software engine be placed? The project has a wide potential: in the classification by skill and experience, it can be both professional and non-professional, since it can be used in health-related applications for both healthcare specialists and those who don't work in health sector. In the classification by wellness, the engine can be used by both patients (that want to increase diabetes self-management skills) and non-patients (for example friends or relatives of a person with diabetes, or a person that is interested in diabetes disease and wants to increase knowledge about the disease). In the classification by purpose, the engine can be used in games of all three presented types – games focused on entertainment, games focused on health, and games focused on acquiring health and medical skills. In the classification by functionality, the

project can be classified as a portable engine that provides progress monitoring and can be used on various platforms.

2.4.2. Examples Of Diabetes-Related Games

This section presents some examples of existing diabetes-related serious games. Most of the presented games were found in review articles (Lehmann, 1997; Lieberman, 2012). The game "DiaSpill" (Makhlysheva, 2013) was developed by one of the author's supervisors.

- "BG Pilot" (Gentry, 1991) (Lehmann, 1997):
  BG Pilot is one of the first diabetes-related games. It is an old cartoon-like pc-game (see Figure 17) that is aimed to teach concepts about diabetes and about living with diabetes better. In this game simulator the player can test his/her knowledge. The goal of the game is simple: the player has to control the flying plane's attitude by food, insulin (short-acting and intermediate-acting) and activity. The duration of the journey is 3 days. The challenge of overnight blood glucose level control is included to the gameplay. If the plane is out of the normal attitude range, it can hit the obstacles at the top or at the bottom of the game screen. The mathematical model of the game is partially based on the Bergman's Minimal Modelling (see section 2.2 for more details).

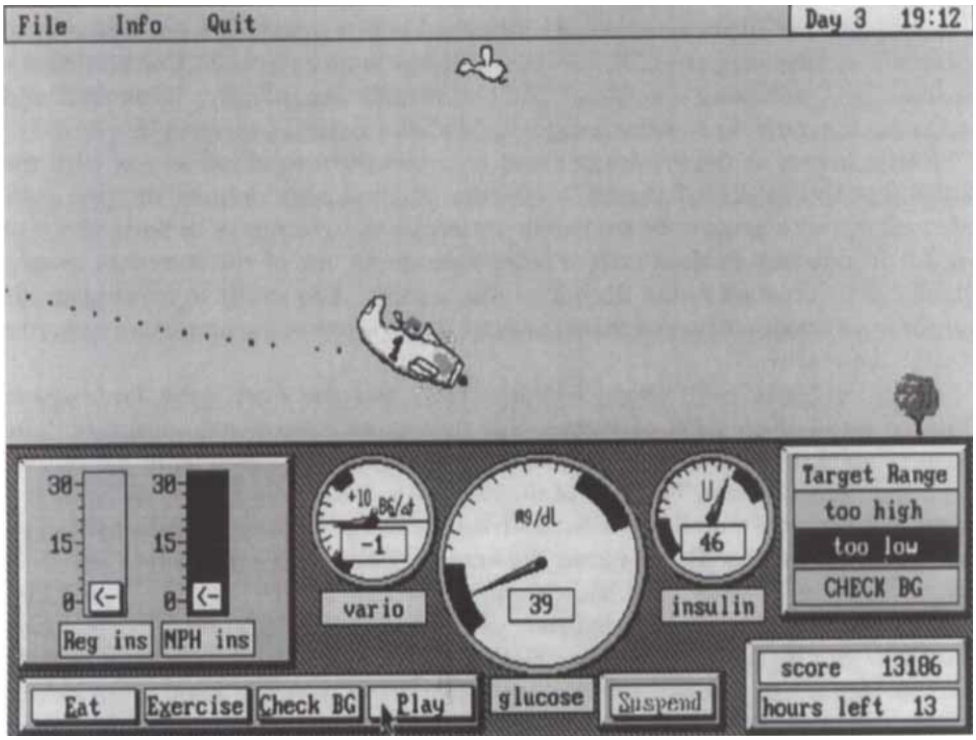


Figure 17: BG Pilot Gameplay.
Source: (Lehmann, 1997, Figure 8)

- "Balance" (Fuchslocher et al., 2011):
  Balance is a pc-game that is aimed to gain skills in self-management for teenagers with diabetes type 1. It is a jump-n-run kind of game, in which the player has to go through

different locations, search for the friends that were captured by strangers and make them free. During the gameplay, the player faces some obstacles and has to control the blood glucose level by eating and using insulin. If the blood glucose level is too high or too low, the player is not able to finish all given tasks. The game challenges the player to find the balance between eating and using insulin. While getting more experienced during the game, the difficulty level increases.

In addition to the regular version of the game, the creators developed an additional version with no diabetes content. They studied both versions and concluded that the version with diabetes content was making bigger enjoyment for players.

- "Captain Novocare" (Hayes, 1994; Lehmann, 1997):

  Captain Novocare is a pc-game for diabetic children. It is based on a compartmental model (see section 2.3.3 for more details), like "PC Pilot". In this game, the player is a superhero with diabetes that has to improve self-care skills and control the illness by taking insulin and eating while having increased level of physical activity at the same time. The Captain has to fight with sugar aliens and rescue the major (that also has diabetes) before the supplies finish. The gameplay reflects a typical diabetes meal plan by simulating 2 days divided into stages. The game gains children's experience about diabetes in an interactive and challenging way, and improves communication between the child, family and friends.
- "Captain Novolin" (Diabetes Health, 1992; Lehmann, 1997):

  This game is a Super Nintendo version of Capitan Novocare with altered compartmental model in the heart of the game. The game was successful – more than 15000 copied of the game were distributed during the first year.
- "DiaSpill" (Makhlysheva, 2013):

  DiaSpill is Alexandra Makhlysheva's master project Android-based mobile phone game for children with diabetes. It is developed in Unity 3D game development framework. The game is aimed to improve children's diabetes self-management skills in an interactive and unobtrusive way.

  The game consists of several levels. The goal is to complete them and free the world attacked by the enemies that can make the whole population fat. At the same time, the player has to monitor and control the main character's blood glucose level by using insulin, eating healthy food and doing activities. During the game, the player has to run and jump over the obstacles, fight enemies, avoid unhealthy food and collect insulin pens and orange juice boxes (as a weapon against the enemies' boss).

  After the short testing period, the children and their parents were given questionnaires. The feedback result was that "the game was recognized as attractive and moderately difficult with remarkable characters' choice, backstory, colors and sounds, real-life bonuses, and the most impressive feature of user's data influence on the gameplay: a game character, enemies and the rewards" (Makhlysheva, 2013).
- Diabetes educational game prototype (Diehl et al., 2011):

  This serious pc-game prototype is an Adobe Flash game that is aimed to gain knowledge of medical doctors and students on insulin management and diabetes treatment. This adventure game presents several clinical scenarios that require player's decision about the best therapeutic and diagnostic choices. After each player's decision, the decision is compared with recommendations from high-quality literature. The player also has access to additional resources of education.
- Diabetes Education for Kids CD-ROM (Dbaza, 2012; Farlex, 2003):

  This is a diabetes pc-game with the aim to help children with diabetes type 1 and their families to improve knowledge about diabetes and self-management. It presents an animated storyline and provides opportunities to apply information in an interactive way.

It consists of several chapters, covering different topics. Each chapter consists of exercises, skill practices and quizzes. The game was tested with 83 children, pre-tests and post-tests were made, and the results showed children's significant knowledge gain and positive feedback.

- The Diabetic Dog Game (The Nobel Foundation, 2015; Lieberman, 2012):
  This game is an online pet simulator, in which the player has to nurture the pet dog with diabetes. The player is challenged to manage and control the dog's blood glucose through insulin intake and nutrition.
- "Power Defense" (Bassilious et al., 2011):
  Power Defense is a highly interactive "tower defense" real-time strategy pc-game developed in Unity Pro game development framework. It is developed for adolescents with diabetes type 1 and is aimed at improving player's numeracy and self-management. The gameplay consists of surviving as many energy attacks as possible, and controlling a diabetic reactor-based power station, in which energy and power output meter represents blood glucose level, real-time coolant represents short-acting insulin, daily super coolant represents long acting insulin. The attacking food represented by energy entities tries to reach the base station, and the player's mission is to place a number of towels at good locations in order to control the number of entities that reach the base, keep the base's power level in the acceptable range, and use coolants when the level is too high. If the level is out of the acceptable range, the station shuts down.
- GameMetrix (GameMetrix, 2013b):
  GameMetrix is a web-based game platform for cloud-based games. "The platform delivers a customized, embedded game experience, customer content and data analytics, unique user insights, and engagement tools centered on games and game mechanics" (GameMetrix, 2013b). Their online quiz game of diabetes knowledge and health "Triviality" is available online (GameMetrix, 2013a).
- "ISULOT" (Aoki et al., 2005):
  ISULOT is a mobile phone game, and "edutainment" learning tool developed to teach children with diabetes type 1 about how insulin and carbohydrates dosage affects blood glucose level. The game is developed in Java 2 Micro Edition. "INSULOT is a special, three-window slot machine designed to teach the relationships among plasma glucose level, food (carbohydrate grams), and insulin dosage" (Aoki et al., 2005). The player uses slot machines as the game interface to estimate the amount of insulin to be used in order to keep the blood glucose level in an acceptable range. The algorithms used to simulate glucose level, calculates the number of carbohydrate grams in each food, the number of carbohydrate grams covered by one unit of insulin, and finally the subtraction of carbohydrates absorbed by insulin from the number of eaten carbohydrate grams. The game was evaluated by 30 patients from 12 to 24 years old, and received a positive feedback.

And other games. More games and/or more detailed overview of them can be found for example in (Lehmann, 1997) or (Lieberman, 2012). Table 2 summarizes information about the presented game examples.

Name	Author	Year	Platform	Type/Goal
BG Pilot	Todd Ramming, Raya Systems Inc.	1989	PC	Plane simulator
Balance	Alberto Fuchslocher, Jörg Niesenhaus, Nicole Krämer	2011	PC	Jump-n-run
Captain Novocare	Susanne Hayes, Raya Systems Inc.	1994	PC	Fight with enemies, rescue the major
Captain Novolin	Raya Systems, Novo Nordisk	1992	Super Nintendo	Fight with enemies, rescue the major
DiaSpill	Alexandra Makhlysheva	2013	Mobile	Jump-n-run, fight with enemies
Diabetes educational game prototype	Leandro A. Diehl, Eldon Lehmann, Rodrigo M. Souza, Juliano B. Alves, Roberto Z. Esteves, Pedro A. Gordan	2011	PC	Adventure game that presents several clinical scenarios
Diabetes Education for Kids CD-ROM	Dbaza	2003	PC	Animated and interactive storyline with exercises, practices and quizzes
The Diabetic Dog Game	The Nobel Foundation	?	Web	Online pet simulator
Power Defense	Ereny Bassilious, Aaron DeChamplain, Ian McCabe, Matthew Stephan, Bill Kapralos, Farid Mahmud, Adam Dubrowski	2011	PC	Real-time strategy game
GameMetrix	GameMetrix	2013	Web, Cloud	Platform for cloud-based games
ISULOT	Noriaki Aoki, Sachiko Ohta, Taisuke Okada, Mariko Oishi, Tsuguya Fukui,	2005	Mobile	“Edutainment” learning game about the effect of different doses of insulin/carb.

Table 2: Summary of game solutions presented in section 2.4.2.

2.4.3. Potential Diabetes Automata Engine Usage

Diabetes-related serious games is one of the examples where the Diabetes Automata engine could potentially be used. During the gameplay, a diabetes-related game could run the blood glucose simulation at the game developer defined speed; input insulin, carbohydrates (if there are different types of food in the game, the game developer could implement a mapping of used types of food into GI, in order to set them into the engine), activity; parametrize the simulation results with the players biometry, and continuously get blood glucose output levels that the game would use, as well as other simulation data provided by the engine (see 6.5.5.3 and 6.5.3). When the game is paused or stopped, it would pause or stop the simulation

currently running in the engine process. If the player wants to save the current game process and load it later, the game would request and get the engine state, save it, and load it into the engine later. The engine keeps track on how much simulation time (or game time) it is. In other words, the blood glucose level simulation would be an essential and integral part of the gameplay.

All implemented engine‘s features and functionality, as well as the information about how it works and how it was designed and implemented, can be found in sections 5.1 and 6.5.

2.5. Existing Systems And Projects

Besides serious games, Diabetes Automata could be used in other diabetes-related applications and systems. This section presents some examples of such systems, as well as the examples of the blood glucose levels simulation solutions that already exist. The examples were found during the discussions with supervisors, as well as from the literature sources (Lehmann, 1997; Makroglou et al., 2005). In the end of the section, the summarizing table is presented.

- *DIAS (Hejlesen et al., 1997; Hejlesen, 1998):*
  Diabetes Advisory System (DIAS) is a decision support system for the management of diabetes. It is used by clinicians to provide better blood glucose management and advices to patients.
  “The core of the system is a compartment model of the human carbohydrate metabolism implemented as a causal probabilistic network (CPN or Bayesian network), which gives it the ability to handle the uncertainty, for example, in blood glucose measurements or physiological variations in glucose metabolism” (Hejlesen et al., 1997).
  The system has three modes: the learning mode, the prediction mode, the advisory mode. In the learning mode, the system uses standard blood glucose, insulin and carbohydrates data for making an estimation of adjustable parameters. The prediction mode is used for predicting blood glucose concentration, unrecognized hypoglycemia and possible effects of changes made in insulin usage and food. The advisory mode can be called as a special case of the prediction mode, and is used to calculate the intake with the smallest risk of hypoglycemia and hyperglycemia (in the prediction mode, the insulin usage changes are suggested manually, in the advisory mode, they are calculated automatically). (Hejlesen et al., 1997)
  The evaluation results that the authors of the paper present, suggests that the system works safely and that the quality of generated advices is at least as good as the quality of the advice of an experienced specialist in maintaining good blood glucose control. (Hejlesen et al., 1997)
  The detailed outline of the system can be found in (Hejlesen et al., 1997), the presentation of technical and physiological aspects can be found in (Hejlesen, 1998), the detailed evaluation information can be found in both.
- *DiasNet (Hejlesen et al., 2000; Plougmann et al., 2001):*
  DiasNet system is based on DIAS, implemented in Java and is a client-server system. This web-based IT system has a wider scope than DIAS - it is used not only by clinicians to provide better blood glucose management and advices to patients, but also by patients as a tool that provides communication and education. “In DiasNet the patients can experiment with their own data, adjusting insulin doses or meals sizes, and thereby learning how to

cope with various situations“ (Hejlesen et al., 2000). “In this way different therapeutic and dietary alternatives can be tried out, allowing the patient to gain experience in achieving glycaemic control” (Plougmann et al., 2001).
The physical structure of the system is shown in Figure 18:

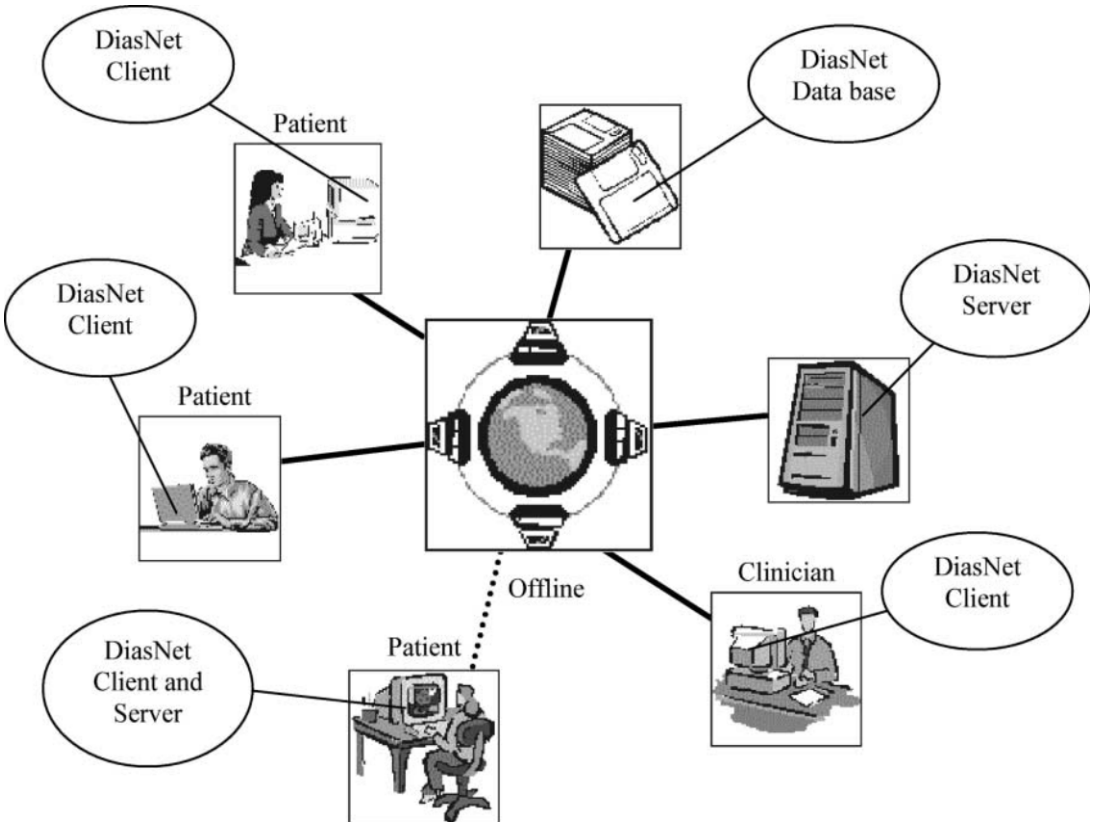


Figure 18: Physical parts of DiasNet.
Source: (Plougmann et al., 2001, Figure 1)

“Utilising the client/server principle a server is used for central data storage and model computations allowing relatively thin clients. The patient and clinicians can access the DiasNet via Internet from a platform supporting *JAVA*. In case of no Internet connection a virtual server can be executed on the client computer” (Plougmann et al., 2001).
Evaluation of DiasNet (Dinesen et al., 2004; Dinesen and Andersen, 2006) was lasting 6 months, having 3 patient-participants with diabetes, and had purpose to find out the observations, experiences and consequences of using the system both by diabetes team and diabetic patients. The authors found several such characteristics, unique and typical, both for patients and the members of diabetes team. For patients, in short words, the results were that patients confidence was increased, personal rapport with diabetes team was improved, patients experienced better self-control, the awareness of carbohydrates with regard to blood glucose regulation was increased, and some other observations and results. The full table of the characteristics and the full evaluation information and report can be found in (Dinesen et al., 2004) and (Dinesen and Andersen, 2006).

- *AIDA (Lehmann, 1998):*

  AIDA is a freeware simulator of interactions between glucose and insulin, and of insulin dosage and dietary adjustments in diabetes type 1. The simulator is developed for educational, teaching (recommended guidelines and training requirements for teaching can be found in (Tatti and Lehmann, 2002) and (Lehmann and Tatti, 2002) respectively) and demonstration purposes. AIDA uses a compartmental model of insulin-glucose interaction in the human body (Lehmann, 1998), "that provides a description of typical glucose-insulin interactions in patients who completely lack endogenous insulin secretion (ie, those with type 1 diabetes)" (Lehmann, 1998; AIDA, 2000). More information about the AIDA model can be found in (Lehmann et al., 1994).

  The first version of AIDA was developed in 1991 (Lehmann and Deutsch, 1991). It has gone through several development cycles and revisions, and in 1996 AIDA v4.0 was released and publicly published online as freeware software. Current web-address can be found in (AIDA, 2000). The web site contains many different links to different kinds of information; offers download links for various operating systems, online-simulation, feedback; etc. The web-page that describes the online-statistics (AIDA Logstats, 2000) shows the following information for March 1996 - April 2012:

  "There have been over *2.5 million* visits logged to the AIDA Websites since March 1996 and over *400,000* copies of AIDA have been downloaded since then. Over *580,000* simulations have been run at 'AIDA on-line' since August 1998" (AIDA Logstats, 2000).

- *DIABLOG (Biermann and Mehnert, 1990):*

  DIABLOG is an educating computer simulator for patients with diabetes type 1. It can simulate glucose and insulin profiles of 24 hours using mathematical modelling of glucose-insulin processes in a human body. The simulation is displayed as curves. The variables are: the insulin injection time and dose (for both short-acting and intermediate-acting insulin), carbohydrates in meals; and also, user can switch to insulin pump therapy. The simulator was evaluated with 22 patients and was well-accepted, even by patients that had no experience with computers. (Biermann and Mehnert, 1990)

- *Glucosim (Erzen et al., 2001) (Agar et al., 2005):*

  Glucosim is a web-based educational simulation package for glucose and insulin levels in an average human body. The simulator can be used in educational purposes, for performing virtual simulations and experiments with different characteristics, diet and exercise conditions (Erzen et al., 2001). The simulator is available at (GlucoSim).

- *DIABTel (Gomez Aguilera et al., 1989; Gomez et al., 1996):*

  DIABTel is a telemedicine system that complements the daily care of patients with diabetes. The system is based on the compartment glucose/insulin "dynamic model" described in (Gomez Aguilera et al., 1989), and consists of 2 units: palmtop-based patient unit, and PC-based medical unit for healthcare specialists/physicials/nurses. The architecture of the system is represented in Figure 19.

  The system has the following purposes:

  "(1) to improve communication of the patient with the hospital-based diabetologist, in between the patient's visits to the clinic; (2) to allow doctors to assess the patient's condition on a frequent basis (every week); (3) to help patients with management in the daily care of diabetes, and (4) to provide patients with a service of 'supervised autonomy', to increase patient's independence without decreasing the necessary continual support and supervision from the doctor" (Gomez et al., 1996).

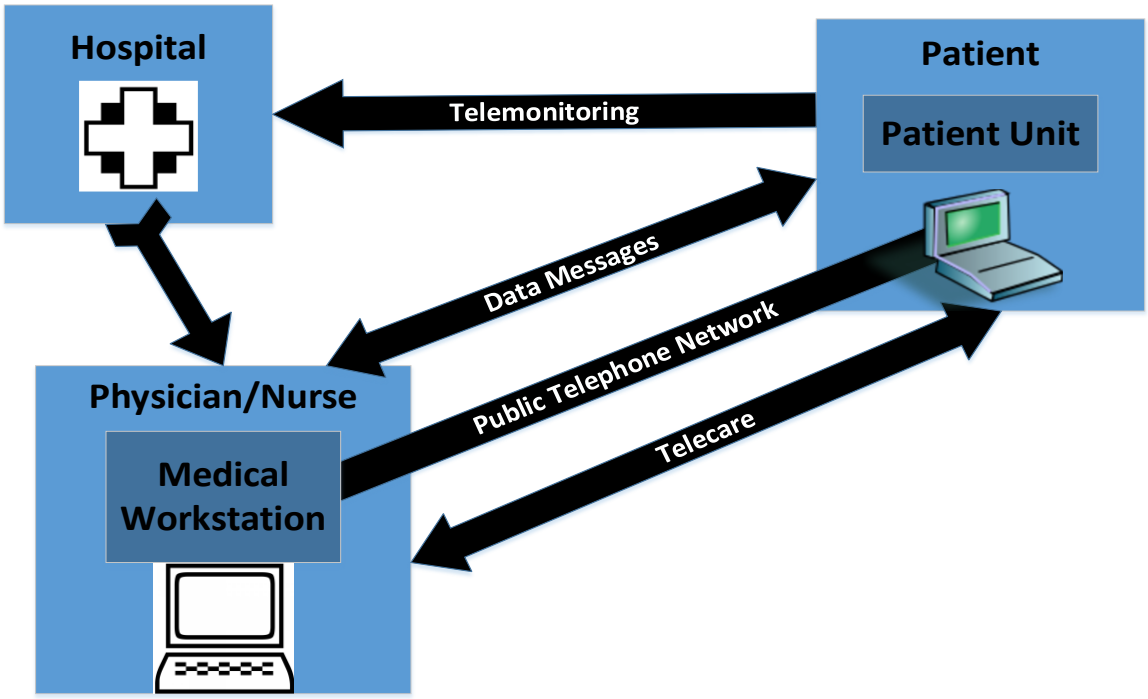


Figure 19: The DIABTel architecture, medical unit and patient unit. Based on: (Gomez et al., 1996, Figure 1).

- *WinSAAM (Stefanovski et al., 2003) (Novotny et al., 2003):*
  WinSAAM (simulation, analysis and modelling) is a freely available (WinSAAM, 2007) interactive biological modelling software (for Windows operating system). It can be used for modelling glucose and insulin kinetics in particular. The history of the software has begun with the original SAAM more than 50 years ago (Berman et al., 1962) (Berman et al., 1967). The SAAM developers at National Institutes of Health (NIH, 2015) developed SAAM into CONSAAM (Boston et al., 1981), then SAAM II (Barrett et al., 1998) and finally WinSAAM. WinSAAM includes "capability to (1) simulate systems, (2) fit models to data, (3) provide measures of uncertainty of parameter estimates, and (4) to be user-friendly" (Stefanovski et al., 2003). The description of the examples for glucose minimal models, glucose and insulin kinetics, can be found in (Barrett et al., 1998), (Stefanovski et al., 2003).
- *Mlab (Civilized Software):*
  Mlab is a commercial advanced mathematical modelling software for solving simulation and modelling problems. The software is developed for various PC (Windows, Linux) systems, Mac and others. The example of the implementation of "Minimal Models for Glucose and Insulin Kinetics" can be found at (Civilized Software).
- *A Matlab suit by Natal Van Riel (Van Riel, 2004):*
  This Matlab suit of routines for a frequently sampled intravenous glucose tolerance test by Natal van Riel (Eindhoven University of Technology), implements minimal models for glucose and insulin kinetics in Matlab.

And other systems and projects. More projects and/or more detailed overview of some systems and projects can be found for example in (Lehmann, 1997) or (Makroglou et al., 2005).

The table that summarizes information about the presented systems and projects, is presented in Table 3.

Name	Author	Year	Platform	Description
DIAS	Ole Hejlesen	1997	PC	Decision support system for the management of diabetes
DiasNet	Ole Hejlesen	2000	PC	Web-based system for decision support, communication and education
AIDA	Eldon D. Lehmann	1998	PC	Simulator of interactions between glucose and insulin
DIABTel	Enrique J. Gómez Aguilera	1996	PC	Telemedicine system for daily care of patients with diabetes
DIABLOG	Berhard Biermann, H. Mehnert	1990	PC	Educating simulator of glucose and insulin profiles and glucose-insulin processes
Glucosim	Meriyan Oruklu, Baris U. Agar, F. Ceylan Erzen, Ali Cinar	2001	PC, Web	Web-based educational simulation package for glucose and insulin levels
WinSAAM	Darko Stefanovski, Peter J. Moate, Raymond C. Boston	2003	PC	Interactive biological modelling software
Mlab	Civilized Software	?	PC	Advanced mathematical software for simulation and modelling
A Matlab suit by Natal Van Riel	Natal Van Riel	2004	PC	Suit of routines for intravenous glucose tolerance test

Table 3: Summary of the projects and systems presented in section 2.5.

2.6. Summary

In this chapter, we have discussed theoretical background and state of the art for the project – about diabetes type 1 and what affects blood glucose levels, serious games, formulas and equations, and existing models and projects.

In people with diabetes type 1, body doesn't produce insulin to keep the blood glucose levels in acceptable range, between 4 and 10 mmol/L, so they have to inject insulin by themselves. There are several types of insulin: rapid-, short-, intermediate-, and long-acting. If too much insulin is injected, the risk of hypoglycemia occurs. If not enough insulin is injected in order to cover the glucose that appears in blood from eaten food, the risk of hyperglycemia

occurs. Both hypoglycemia and hyperglycemia are dangerous and can seriously damage health or cause coma. The lower glycemic index of eaten food is – the less it will rise blood glucose levels and the longer the raising effect will last.

Besides different types of insulin and carbohydrates, there are other things that affect blood glucose. Different types of activity, dawn phenomenon and constant blood glucose consumption by brain are among them.

After that we have discussed what serious games are, what advantages they have, how they can be classified, and what the examples of diabetes-related serious games are.

Finally, we have discussed some simple formulas that either are or were used in the project; and the examples of existing mathematical models, software systems and projects in the field of blood glucose simulation/prediction.

3. Methods

As mentioned in section 1.5, Diabetes Automata is an experimental project in software engineering combined with experimental health science, in which the choice of methods and the source and literature sources plays important role.

This chapter describes the methods used for information search, design, development and implementation, testing, and results' evaluation. First, we will look at the search of the related literature and work. Then we will shortly discuss tools, methods for design & implementation, testing and evaluation. The chapter ends with critique of chosen methods and summary.

3.1. Literature Review

Before the start of the project development, theoretical and literature search were done, in order to gain knowledge and understanding about diabetes, physiological processes, clinical pharmacology, find out what models and solutions already exist, understand how the engine could be developed and what to base the future algorithms on. The work with literature has been done during the development period as well, in order to improve or change the implemented algorithms or implement new algorithms.

The keywords for literature search were various combinations of the example words presented below:
Diabetes, type 1, type 2, serious games, serious games for health, diabetes games, blood glucose, blood sugar, simulation, minimal model, compartment model, insulin, carbohydrates, activity, aerobic activity, anaerobic activity, resistance exercise, glycemic index, glycemic load, blood glucose metabolism, glucose sensitivity, glucose infusion, glucose utilization, insulin sensitivity, stress hormones, growth hormone, dawn phenomenon, pharmacodynamics, brain, metabolic profile, action profile, total body water, blood volume, blood glucose modelling, blood glucose simulator, simulation, blood glucose prediction, hyperglycemia, hypoglycemia, basal, bolus, long-acting insulin, intermediate-acting insulin, short-acting insulin, rapid-acting insulin, object-oriented programming, java, android, fluid volume, blood components, plasma volume, interstitial volume, extracellular volume, intracellular volume, molecular weight, carbohydrate factor, correlation factor, hypoglycemia unawareness, mobile health applications, glucose consumption, factors affecting blood glucose, α-cell, β-cell, glucagon, blood glucose monitoring, diabetes self-management, glucose uptake, metabolic effect, glucose-insulin interaction, system architecture; names like Humalog, Novolog, Regular, NPH, Levemir, Lantus, DiasNet, etc.; and other.

The tools used for the information and literature search are Endnote, Google Search, and online libraries such as PubMed. The total number of articles found in Endnote during the whole thesis-writing period is 3956. The total number of references used in this thesis (including articles, books, web-pages, posters) is 213.

3.2. Tools, Languages, Technologies, Frameworks

The programming language used for the project development is Java. The tools and frameworks are Eclipse IDE for Java Developers Version: Luna Release (4.4.0) and Android Developer Tools plugin. The mobile platform for the simulator prototype development was Android 4.4 and 5.0. For more information see chapter 6.

3.3. Development Organization And Discussions

The place of project development was NST. It was providing opportunities for direct contact with supervisors, face-to-face discussions, help and fast feedback. The feedback from the supervisors and test persons was playing one of the key roles, since it was providing information about what should be fixed, changed or improved in the prototype.

External help played a big role as well – one of the external contact persons, Ole Hejlesen (the author of Dias and DiasNet systems which are described in section 2.5), helped with understanding of questions in physiology and clinical pharmacology that occurred during the development and in particular while re-implementing one of the engine components (see section 6.5.6 and Appendix D: Previous Implementation Of Insulin Module). Another contact person, Georg Sager (professor at the Department of Medical Biology, UiT), helped with problems that occurred while testing the prototype with one of the test persons. See chapter 7 for more information.

Solutions for most of the problems that were occurring during the development and programming, were found on StackOverflow website (StackOverflow.com, 2015).

3.4. Testing

Three testing methods were done. The first method is feedback from testers during the development. It consists of sending of new prototype versions to the testers, getting feedback about usability, accuracy of the algorithms and bugs, make corrections and improvements of the prototype and repeat the same steps.

The second testing method is simulation trial by testers, which means that test persons run the simulator prototype, continuously register information about insulin, carbohydrates and activity, and see how close the resulting blood glucose level is to the real level.

The third testing method is testing of the engine algorithms with testers' private data. During this testing, the testers provide a 2-day sample database from the Diabetesdagboka together with collected detailed information about insulin, carbohydrates and activity; they are used in order to hardcode the information into the simulator, run it with the provided database, and see the deviations between the measured blood glucose level and the simulated level.

The developed prototype is used as a testing tool for the second and the third testing methods.

3.5. Evaluation And Its Analysis

Evaluation of the testing results consists of the comparison of the real measured blood glucose levels and the calculated levels, seeing how much they differ from each other. The feedback from the testing persons, their opinions about the prototype and about how good it works, is also an important part of testing and evaluation.

3.6. Critique Of Chosen Methods

The project could be based on one of the mathematical models presented in 2.2. However, the author's level of knowledge in mathematic was Calculus 1, and in order to understand those models and implement them as programming code the knowledge in mathematics must be higher. So the author was searching and found alternative ways and methods to implement the prototype. See sections 2.1, 2.3 and 6.5 for more information about what the software model of the project is based on, and how it was implemented.

About the development platforms – it would be better to start the development on the current newest version of Android – 5. It would save more time, since the development was going on Android 4.4 most of the time, the testers were having Android 5, and most of crashes that were happening on Android 5, were not happening on the authors mobile phone for testing.

The development framework could be the newest official Android Studio (Developer.Android.com, 2015b) rather than Eclipse with plugins. However, the development was started in Eclipse, so the author decided to continue and finish in Eclipse.

Testing results could be compared with the results from the testing and evaluation of of the models and projects presented in 2.2 and 2.5. However, the testing methods, the number of test persons, and the prototype accuracy, were different. In addition, the time for testing was limited. See chapters 7 and 7.3 for more information and discussion about testing. In order to test and evaluate the prototype in the way the presented models and systems were tested and evaluated, and compare the results, testing should have involved more people with different biometric information and take much longer time. The complexity of chosen field of research, as well as the complexity of potential comprehensive long-term development and testing, would take much longer time than one study year that was spend for this master project.

Initially, the project supposed to be a part of the project "Spill og Lær med Diabetesvenner" by NST, aimed to develop a diabetes-related game. This could provide opportunities for a good comprehensive testing in practice – in an existing external game that would use Diabetes Automata via its API in order to input the necessary information and get blood glucose levels over time. However, in the middle of the development of Diabetes Automata, plans for "Spill og Lær med Diabetesvenner" project had been changed and Diabetes Automata became no longer a part of it, so the opportunities for the comprehensive testing in practice disappeared. However, alternative testing and evaluation methods mentioned in sections 3.4 and 3.5 were found and conducted.

More information about the alternative testing and evaluation, as well as more critique and discussion of testing and results, are presented in chapter 7.

3.7. Summary

In this chapter, we have discussed the methods used in different phases of the master-project. They consist of literature and source search, in order to gain knowledge in the field of the project, find out what models and solutions already exist, and find the information to base the software model of the Diabetes Automata engine on. The examples of key-words and tools used for the information and literature search, as well as the number of sources found and used, were also presented.

Other methods discussed are the tools, languages and technologies used during the development of the prototype, as well as the organization of the development process. After these, we have looked at the methods used for testing organization, process, and result evaluation.

Finally, the critique of used methods was presented. More critique and discussion of the testing and its results can be found in chapter 7.

4. Requirements Specification

This chapter describes the project's functional and non-functional requirements for the project. As said in 1.2, the project prototype implementation consists of two parts: the Engine itself and the Demonstrator (see chapters 5 and 6). So the functional requirements are also divided into the requirements for the Engine and the requirements for the Demonstrator. The requirements have been created during the theoretical search, and well as during the project architecture creation and project development. They have been defined by the author, while constructing the project architecture and developing the prototype; and his supervisors – Gunnar, Eirik, Håvard, Alexandra, while having group meetings and discussions, and providing information and feedback to the author.

4.1. Functional Requirements

In this section we discuss functional requirements for the Engine and the Demonstrator, or in other words, what exactly they shall do.

4.1.1. Functional Requirements For Engine:

1. The Engine shall continuously simulate the blood glucose level and the blood glucose influence by insulin, carbohydrates and activity;
2. The Engine shall support various kind of insulin, carbohydrate and activity (for example insulin types for insulin, different GI level for carbohydrate, different kinds of activity for activity);
3. The calculations shall be customized for user and depend on the user's biometry;
4. The Engine shall provide API for external programs or applications;
5. It shall be possible to start/stop and pause/continue the simulation in a simple way;
6. The Engine shall contain an "infinite loop" that runs necessary procedures with given frequency;
7. The Engine shall have implemented simulation time/speed management, so that it would be possible to speed up the simulation;
8. The Engine shall have implemented State data structure that holds the simulation/biometric settings, simulation data and currently active insulin/carbohydrate/activity records, providing the opportunity to keep the data and continue the previously started simulation if the simulation was paused;
9. The Engine shall be able to reset the simulation, the settings and the state in a simple way;
10. The Engine shall be language-independent;
11. It shall be possible to set in a new blood glucose level value "on-the-go" (while simulation is running);
12. The Engine shall have default settings and be able to reset the settings using them.
13. The Engine shall handle incorrect parameters that external applications could pass as arguments for the API-methods;

14. The Engine shall have Database module that holds the records that are no longer affecting blood glucose levels; the priority of this requirement was set to low, since the Engine can function and be tested without the Database module; the fulfillment of this requirement depends on there is enough time for the development of the Database module.

4.1.2. Functional Requirements For Demonstrator:

1. The Demonstrator shall interact with the Engine via the API by the Engine;
2. The Demonstrator shall demonstrate the Engine's features and functionality;
3. The Demonstrator shall contain the following minimal set of screens: main screen, new simulation, settings;
4. The Demonstrator's main screen shall have clickable buttons to: set the simulation time speed, make new insulin/carbohydrate/activity records, change/correct blood glucose level "on-the-go", launch "new simulation" screen if there are no ongoing simulations, pause/continue/stop the ongoing simulation;
5. The Demonstrator's main screen shall display the following: current simulation time and speed, summary information for the last 24 simulated hours, information about the current blood glucose influence of insulin/carbohydrate/activity, current blood glucose level, graph with the curve representing the levels of blood glucose level over the simulated time, necessary control buttons like start/stop/pause/continue;
6. While a simulation is running, the Demonstrator's main screen with all its changing information shall be automatically updated with application-defined frequency;
7. When the ongoing simulation is paused, all buttons of "add a new record" type should become not clickable, and when the simulation is continued they should become clickable again;
8. The Demonstrator's settings screen shall contain editable application and engine settings, including biometric values;
9. If the application is launched the first time after installation, or if the application settings are corrupted, it shall import the standard settings from the Engine;
10. Otherwise it shall update the Engine's settings using its settings;
11. The Demonstrator shall be able to restore the default settings with one click;
12. The Demonstrator shall have the "default settings" containing various editable default settings;
13. The Demonstrator's graph shall have changeable zoom and should move automatically along the simulation time;
14. When the simulation is continued from the pause state, the graph shall be set back to the last point of the simulation time and application-defined zoom;
15. When the ongoing simulation is paused, it shall be possible to zoom in/out and move the graph manually;
16. The Demonstrator shall allow only correctly formatted data to be set in the editable fields (for example in Settings) in order to avoid errors;
17. The Demonstrator (with help of the API) shall be able to save the current simulation/engine state and restore it later.
18. The Demonstrator shall show dialogs (for example when setting in a new record) or pop-up messages (for example "new insulin record was added, insulin type x, value y") when necessary.

4.2. Non-Functional Requirements

In this section we discuss non-functional requirements of the Engine and the Demonstrator, or in other words, how they shall be working. These requirements were defined in order to create a well-functioning engine and demonstrator.

4.2.1. Non-Functional Requirements For Engine:

1. The Engine shall be a project or a library that can be imported and used by external applications (for example games, simulators, demonstrators, etc.) on various platforms or become a part of another project;
2. The Engine shall have modular architecture that is easy to update/change/modify and run some modules in parallel;
3. The Engine shall work stably, responsively and reliably, without crashes;
4. The Engine shall be moderately accessible, providing access to the methods and data required for simulation parametrization and control, but not to other methods and data that should be visible only for the internal code and be accessed from outside the Engine;
5. The Engine consists of multiple loops, iterations and calculations, which will consume CPU-time.The requirement is that the Engine shall not consume too much CPU-time.

4.2.2. Non-Functional Requirements For Demonstrator:

1. The Demonstrator shall work stably, responsively and reliably; this requirement is excluded from the integration with Diabetesdagboka, since this part is very experimental (see section 6.6 for more information);
2. The Demonstrator shall be simple-looking, user-friendly and easy-to-use;
3. Since the Demonstrator uses the Engine, it will also consume CPU-time. The requirement is that the Demonstrator shall not consume too much CPU-time;
4. The Demonstrator shall be informative, since it is aimed to demonstrate the functionality and features of the Engine.

5. Design

This chapter presents the history of how the project prototype was designed. It consists of 2 separated parts – the engine itself and the demonstrator that shows how the engine works and what functionality it has.

5.1. Engine Architecture

This section presents the design of the Diabetes Automata Engine.

5.1.1. Architecture Overview

The engine architecture consists of modules: Main Module, Insulin Module, Carbohydrate Module, Activity Module, Blood Glucose Module, State Module, Database Module and Collection of formulas and rules. The architecture is represented in Figure 20:

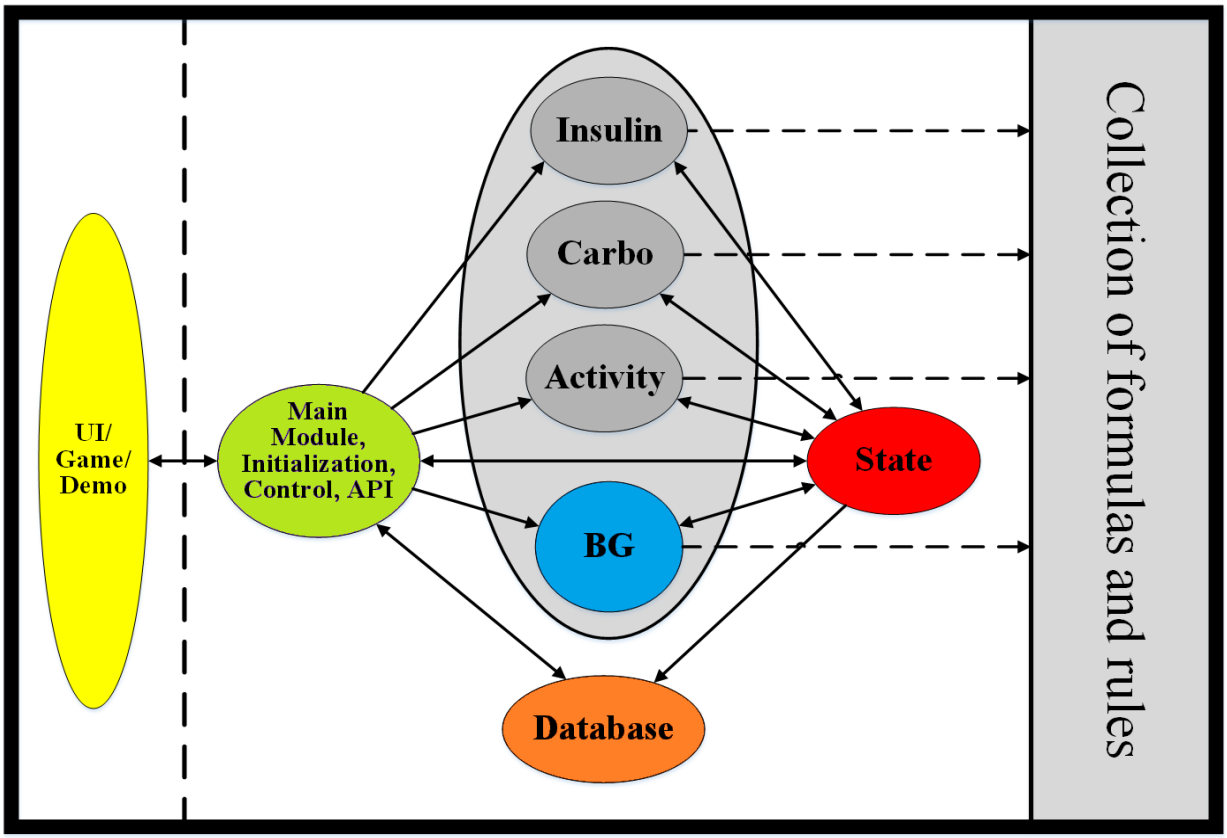


Figure 20: The Engine Architecture.

Main module is responsible for initialization, simulation process, communication and interaction with external UI via API. State module keeps the current state of the engine – program and personalization settings, simulation parameters and state, current blood glucose level, currently active effects and progress of insulin, carbohydrate and activity modules. Database module is designed to keep the information that the State module doesn't need anymore – the information and details about the effects that are finished.

The "action group" consists of 4 modules. Insulin/Carbohydrate/Activity modules calculate how currently active insulin/carbohydrates/activity affects the current blood glucose level. Blood glucose module updates the blood glucose value after Insulin/Carbohydrate/Activity calculations and after the calculations of glucose consumption by brain and dawn phenomenon influence in the module. The calculation methods used by the "action group" modules are collected in one place.

5.1.2. How It Works

When the main module is called by the simulator/game UI, the main module starts its main loop that runs until the simulation is stopped or paused. If the ongoing simulation is paused, the main module doesn't reset the engine state and just stops the execution. When the simulation continues, the loop starts executing again from the same state as when the simulation was paused. If the ongoing simulation is stopped, the engine stops the loop and resets the simulation state.

In the main loop, the module calls the four "action group" modules in a sequence. Insulin, Carbohydrate and Activity modules peak each active insulin/carbohydrate/activity record stored in the State, and calculate the current blood glucose change in the module for each record by using specified sequence of formulas from the Collection. The final result in the sum of calculated blood glucose level changes from each record. The results are saved in the State module when the calculations are complete. The Blood Glucose module reads these results and updates the current State blood glucose level.

If a record becomes inactive and is not affecting the blood glucose level anymore, it is deleted from the State and moved to the Database module. The State and the Database should be stored on disk and be frequently updated.

The loop repetition frequency, the pause/running/stopped status, and all other simulation parameters are saved in the State. The simulation speed (the number of simulated seconds/minutes per second) can be changed by the UI.

5.1.3. Why This Architecture?

With such functionality separation and module architecture, the development process is expected to be easier and clearer. Moreover, if blood glucose simulation becomes too complex and has a huge number of calculations to be done during a short amount of time, it will be easy to modify the code and make each "active group" module run in a separate thread – it can give performance gain in case if the device that runs the engine has two or more processing cores.

5.2. UI

The overall idea of the UI is to look simple and to demonstrate how the engine works and which functionality it has. The UI demonstrator is a mobile application (see section 6.6 for more details). First, we will look at the UI mockups, then we will look at the previous implemented UI versions. The last version of the Demonstrator and the information about how to use it, is presented in section 6.6.

5.2.1. Mockups

In the beginning of the project development, some UI mockups were made. Let us take a look at them. The screenshots from the final version of the implemented UI are presented in section 6.5. The main screen of the simulator was initially designed to look close to the representation in Figure 21:

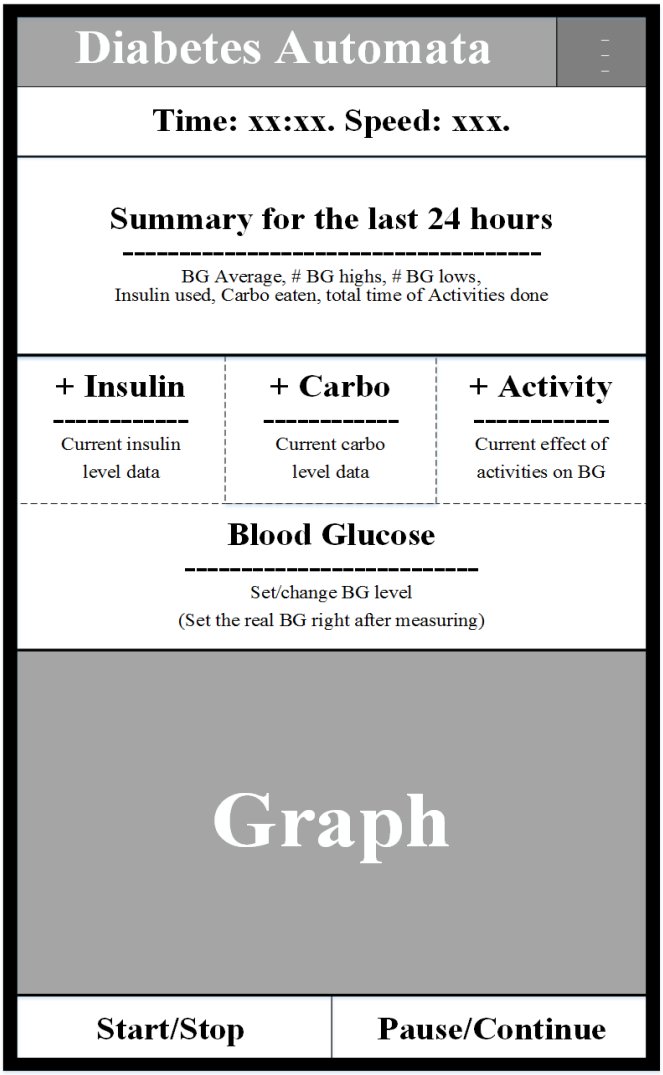


Figure 21: The mockup of the Main screen of the Demonstrator.

The “time” displays how much simulated time has already passed, and the “speed” displays the current simulation time speed that could be changed by clicking on the parent rectangle.

The “summary” field displays some information about what happened during the last 24 simulated hours, for example the number of blood glucose highs/lows, the average blood glucose level, injected insulin, consumed carbohydrates and done activities.

“Insulin”, “Carbo” and “Activity” fields display information about how each of the related modules are currently affecting the blood glucose level. They would also be buttons that could be clicked in order to add a new specific record to the simulation.

“Blood glucose” would display the current blood glucose level and be a button that can give an opportunity to change the current blood glucose manually.

“Graph” would display graphical representation of the blood glucose levels as a curve. It may also display the same kind of information for insulin, carbohydrate and activity.

The button “pause/continue” pauses or continues the current simulation. The button “start/stop” would open the “New Simulation” screen if the simulation is not currently running, otherwise it would stop the current simulation.

The “New Simulation” screen was initially designed to look close to the representation in Figure 22:

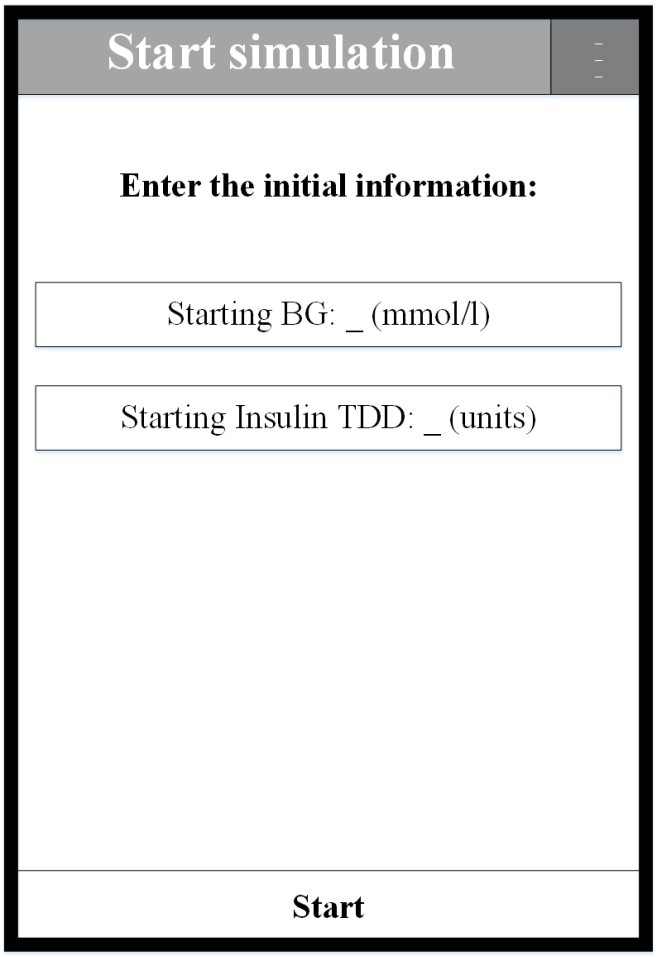


Figure 22: The mockup of the Start Simulation screen of the Demonstrator.

A new simulation is started by pressing the “Start” button. The simulation accuracy depends on several parameters that user should update if necessary. Two of those parameters can be changed from this screen, they are: the starting blood glucose level (which value to start the simulation with) and the initial TTD (total amount of insulin consumed per 24 hours). The TTD was used in the older versions of the prototype and is no longer used in the newer ones (for more information, see section 6.5.6 and Appendix D: Previous Implementation Of Insulin Module).

Other parameters are accessible in the Settings screen that was initially designed to look close to the representation in Figure 23. The Settings screen gives opportunity to change the personal parameters and customize the simulation by setting the user’s biometric information, which is used in engine’s calculations. The button “Use Default Values” shall restore the default settings in one click. The Default Settings screen was initially designed to look close to the representation in Figure 24.

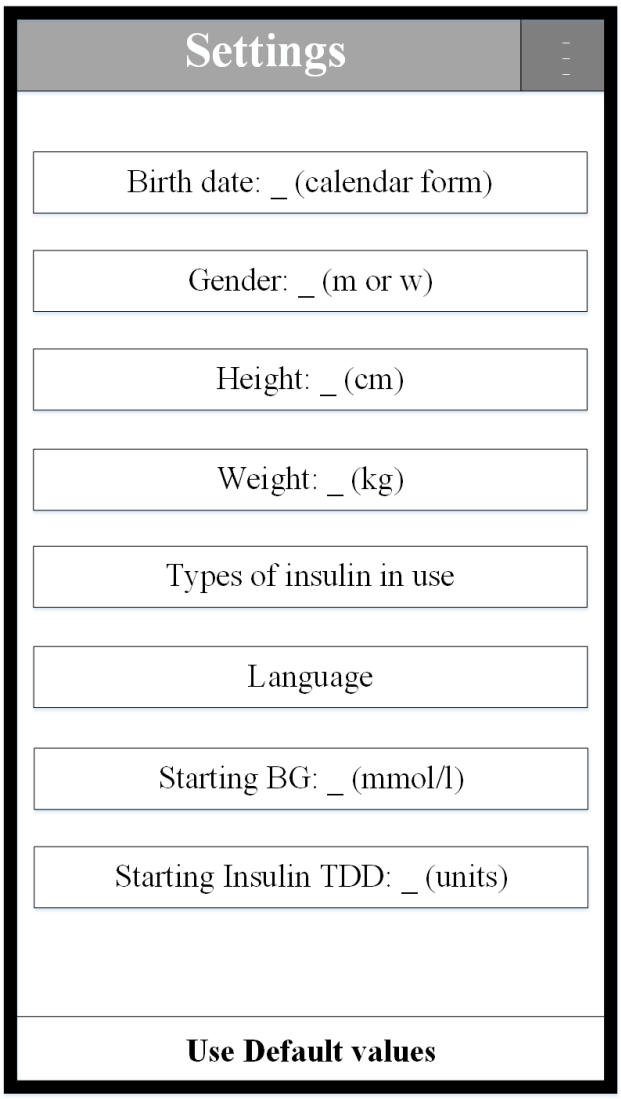


Figure 23: The mockup of the Settings screen of the Demonstrator.

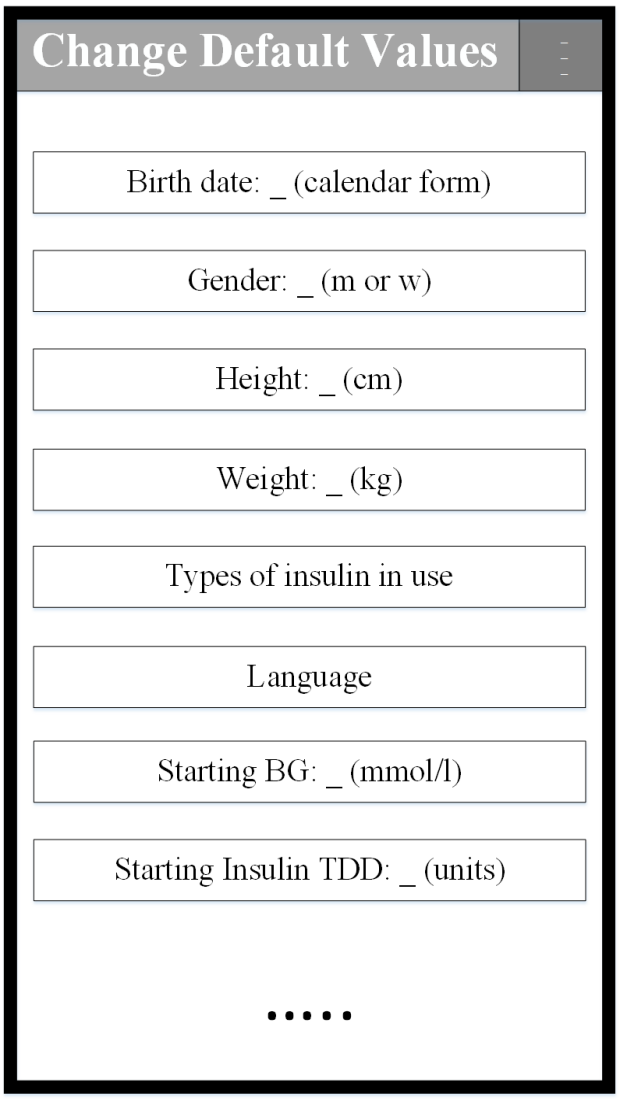


Figure 24: The mockup of the Default Settings screen of the Demonstrator.

It provides opportunity to change the default values, a part of which is the same parameters as in “Settings”, together with additional settings that have been defined later during the development.

5.2.2. Previous Versions Of The Demonstrator

The screenshots of the previous prototype versions are presented in Figure 25 – Figure 66, which demonstrate what changes were done from version to version. Since the project is much more about algorithms than UI-design, there were not many changes in the UI during the project implementation. However, the presented screenshots demonstrate the progress in the development of the UI, and partially demonstrate the progress in the development of the project.

In particular, the prototype version v0.001a is shown in Figure 25 – Figure 36, the changes made in versions v0.002a-v0.003a in comparison to v0.001a, such as new Activity dialog, changed Insulin dialog, representation of insulin, carbohydrates and activity with different colors on the graph, are represented in Figure 37 – Figure 39.

The changes made in version v0.004a in comparison to v0.003a, such as new elements in the options menu when the simulation is not running and running, changed New Simulation Screen, Blood Glucose field having green/yellow/red colors when the blood glucose level is normal/high/low respectively, implementation of notification, implementation of manual and interrupted simulation save/load, are shown in Figure 40 – Figure 48.

The changes made in version v0.006b in comparison to v0.004a, such as changed New Simulation Screen, new simulation mode and new simulation data representation on the graph, new elements and simulation parameters in the options menu, new elements on the Main Screen, are shown in Figure 49 – Figure 52.

The changes made in version v0.007d in comparison to v0.006b, such as changed New Simulation Screen and new simulation mode, changed and new elements on the Main Screen, implemented different graph sizes, “Please Wait…” mini-windows, new graph mode, are presented in Figure 53 – Figure 58.

The changes made in version v0.008g in comparison to v0.007b, such as new elements on the Main Screen, changed elements in the options menu, Main Screen in the landscape mode, new representation of all clickable elements of the Main Screen that makes them look like buttons, are presented in Figure 59 – Figure 66.

The screenshots of the last version of the prototype, together with description of each element of its screens, can be found in section 6.6.

The full overview of the version history is presented in “Appendix C: Prototype Version History”.

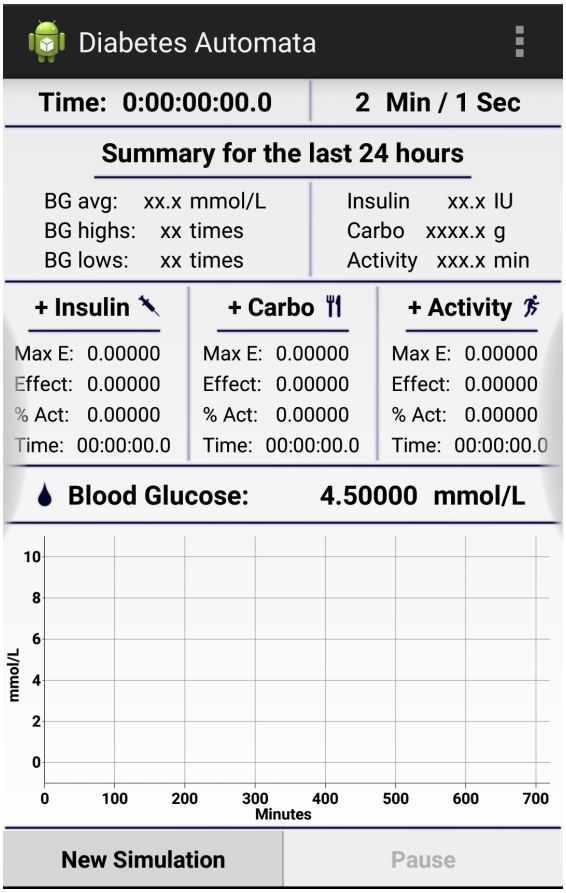


Figure 25: The Main screen of the prototype v0.001a.

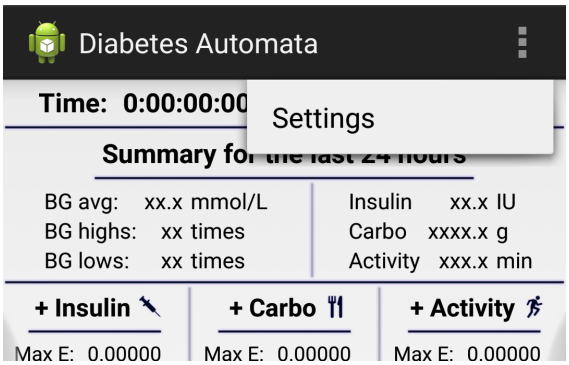


Figure 26: Pressed options menu of the Main screen of the prototype v0.001a.

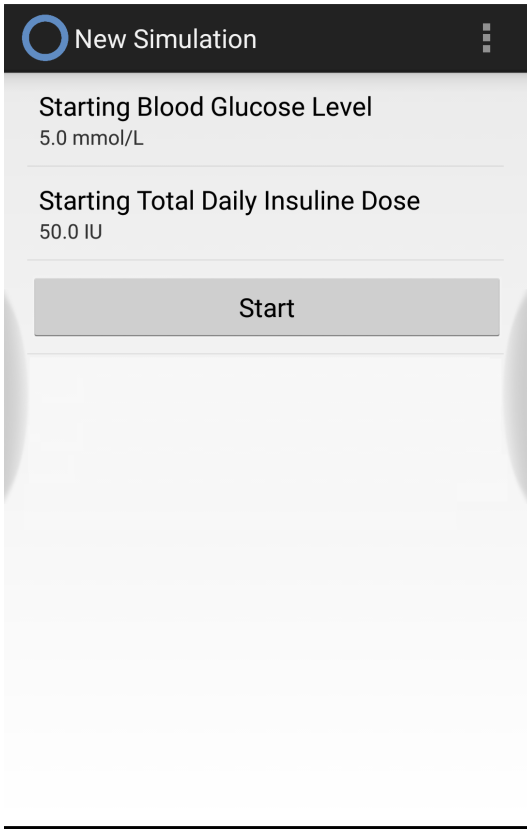


Figure 27: New Simulation screen of the prototype v0.001a.

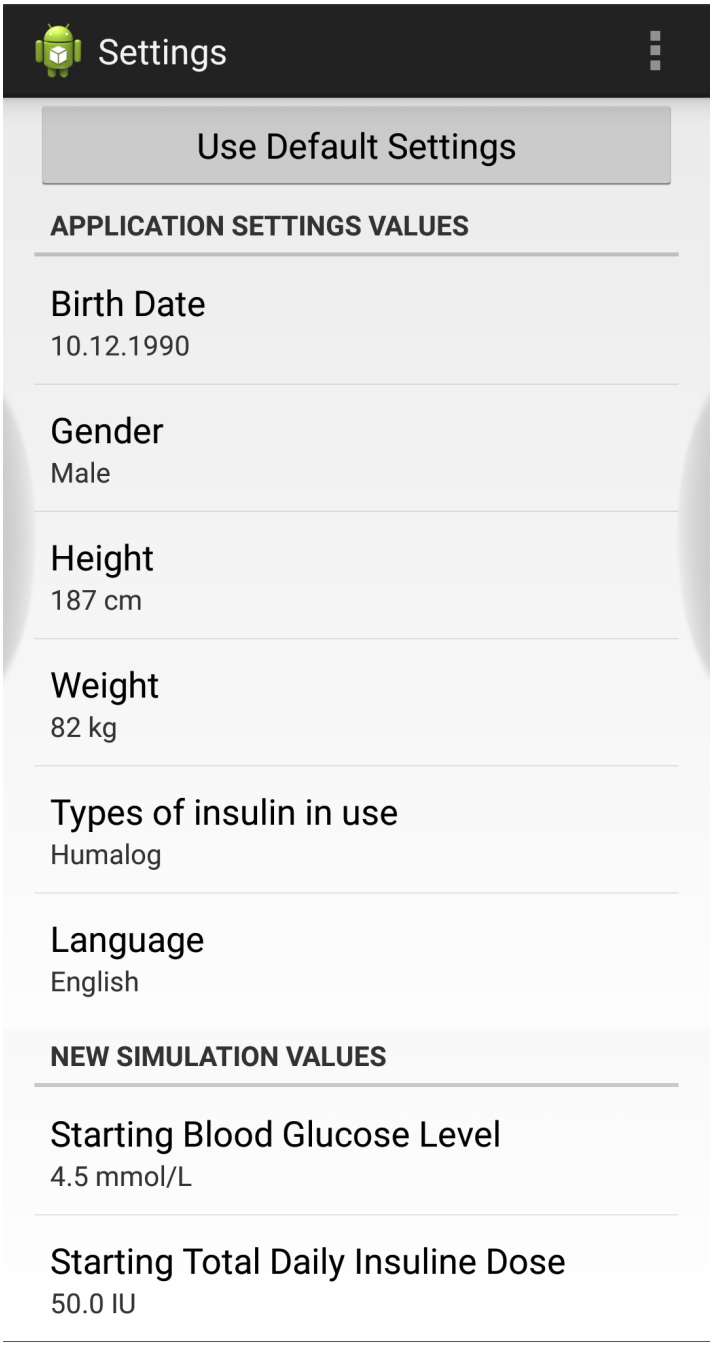


Figure 28: Settings screen of the prototype v0.001a.

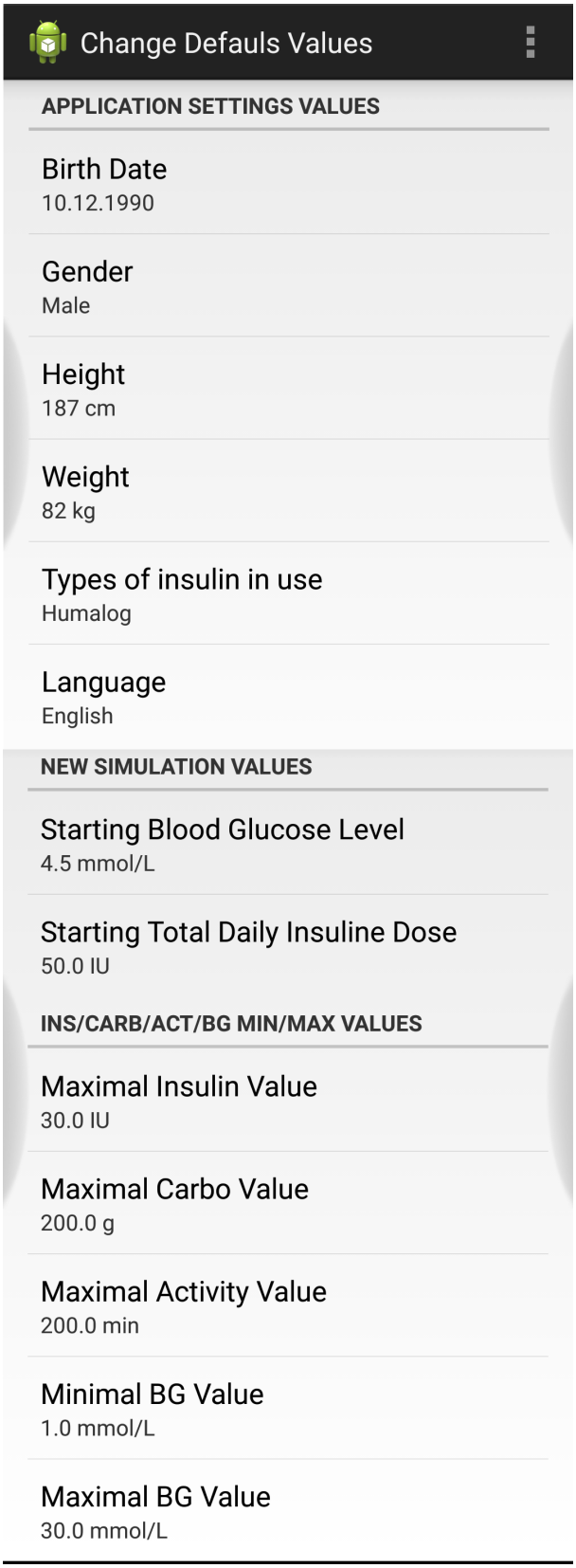


Figure 29: Default Settings screen of the prototype v0.001a.

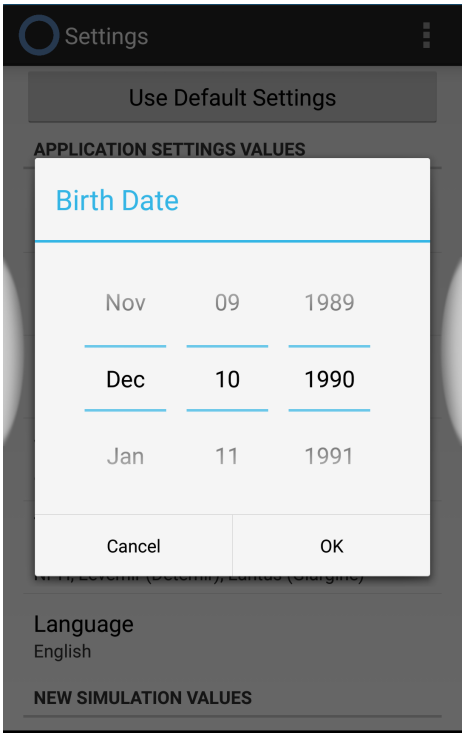


Figure 30: Setting user's birthday on the Settings screen of the prototype v0.001a.

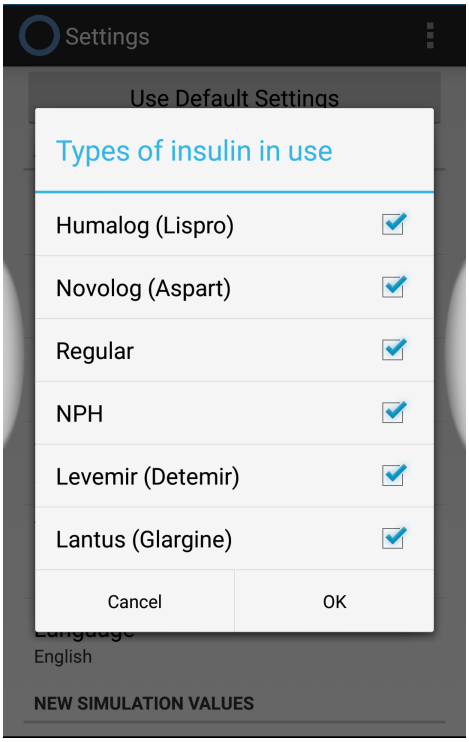


Figure 31: Setting types of insulin that user uses on the Settings screen of the prototype v0.001a.

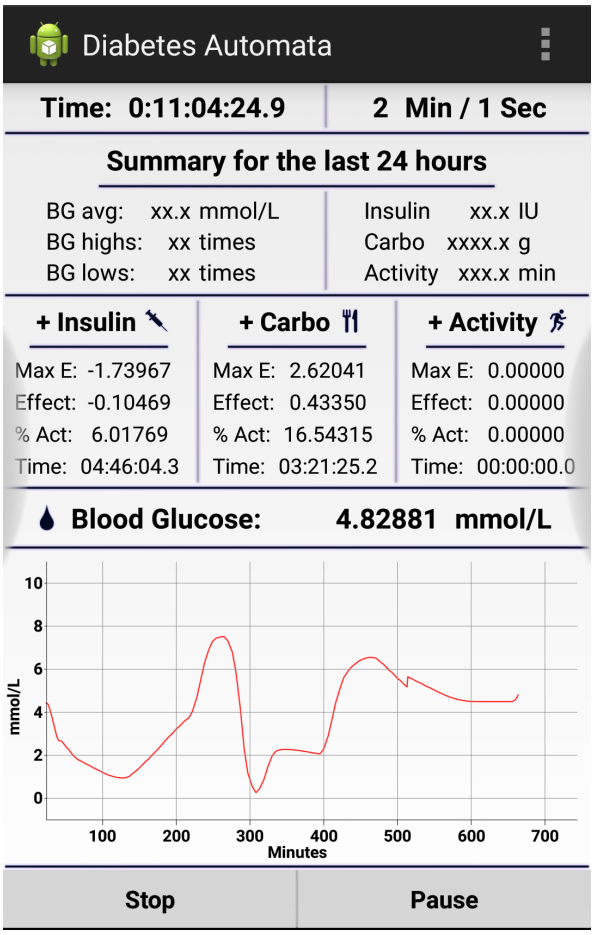


Figure 32: The Main screen of the prototype v0.001a with running simulation.

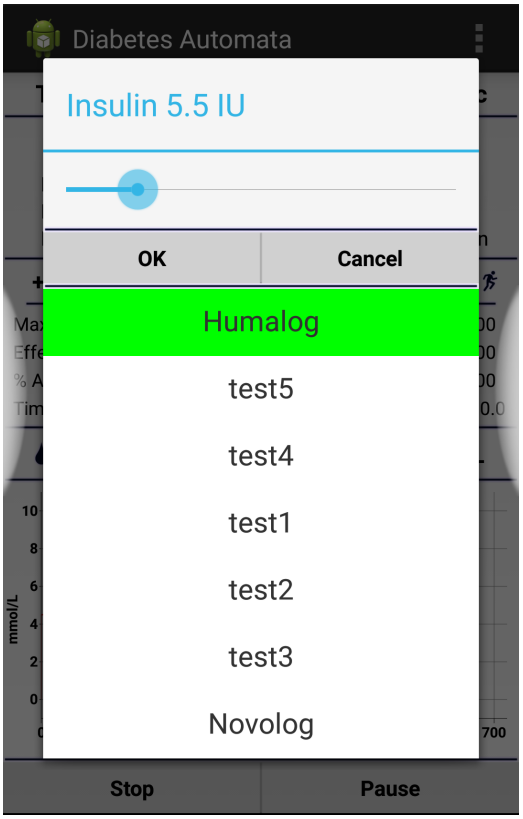


Figure 33: New Insulin Record dialog on the Main screen of the prototype v0.001a.

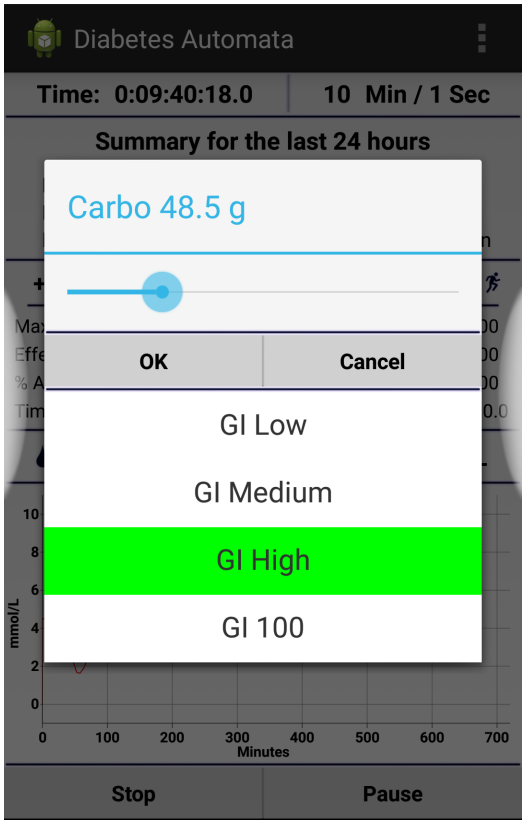


Figure 34: New Carbo Record dialog on the Main screen of the prototype v0.001a.

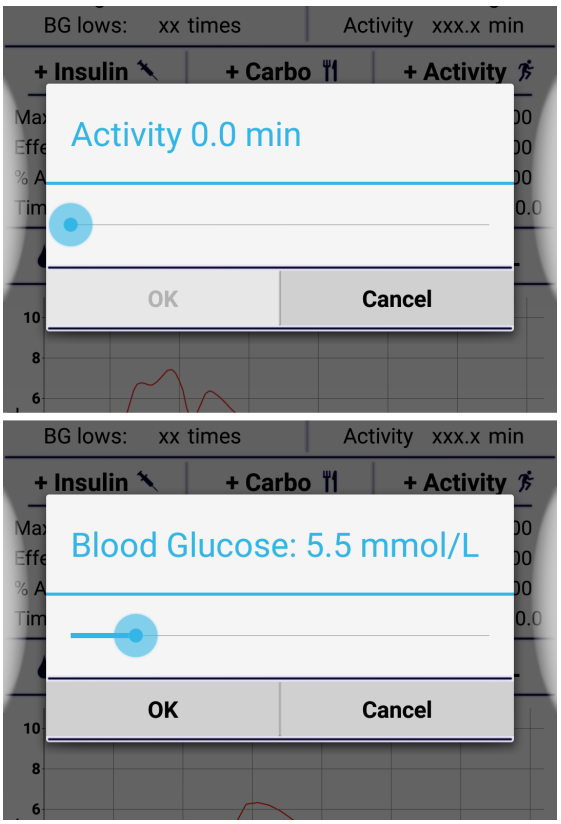


Figure 35: New Activity Record and Manual Blood Glucose Level dialogs on the Main screen of the prototype v0.001a.

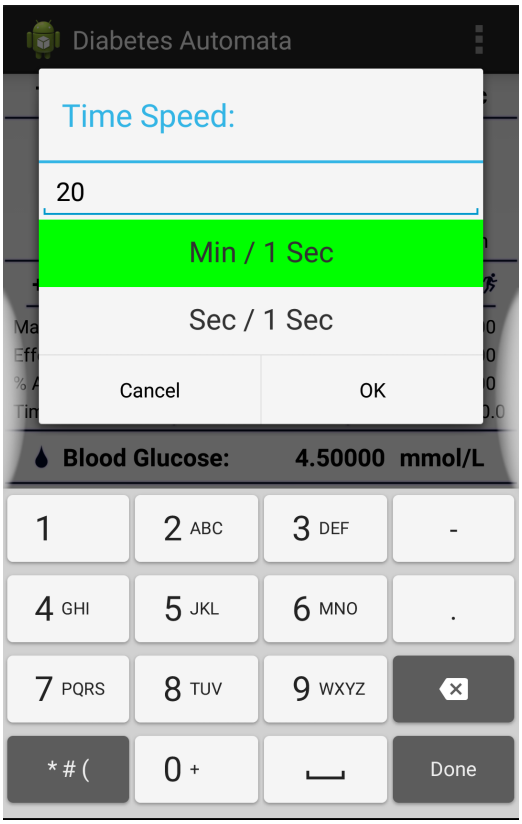


Figure 36: Time Speed dialog on the Main screen of the prototype v0.001a.

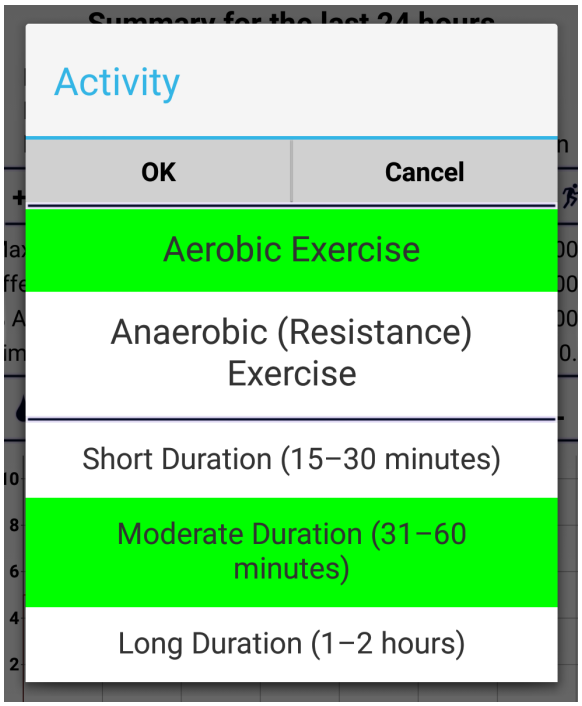


Figure 37: New Activity Record on the Main screen of the prototype v0.002a.

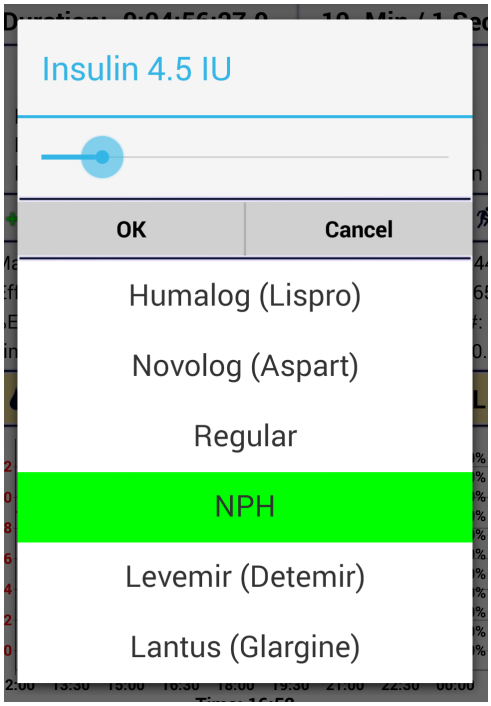


Figure 38: New Insulin Record dialog on the Main screen of the prototype v0.003a.

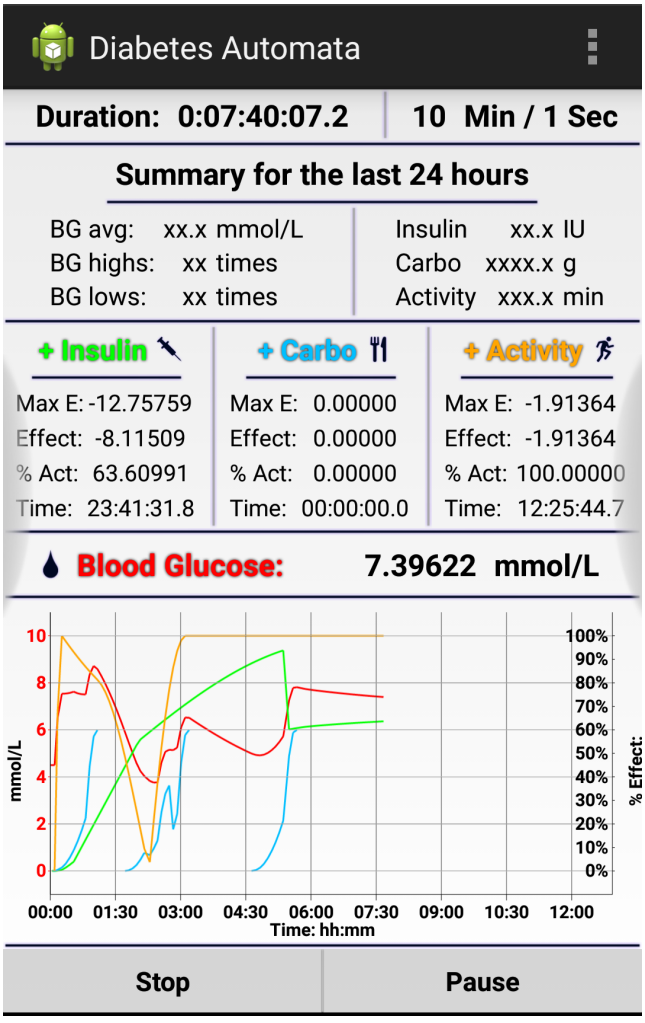


Figure 39: The Main screen of the prototype v0.002e with running simulation.

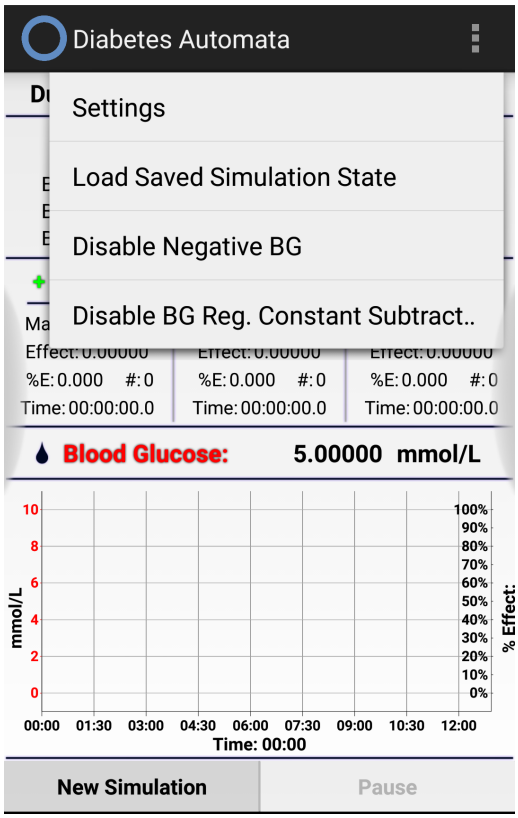


Figure 40: The Main screen of the prototype v0.004a with pressed options menu.

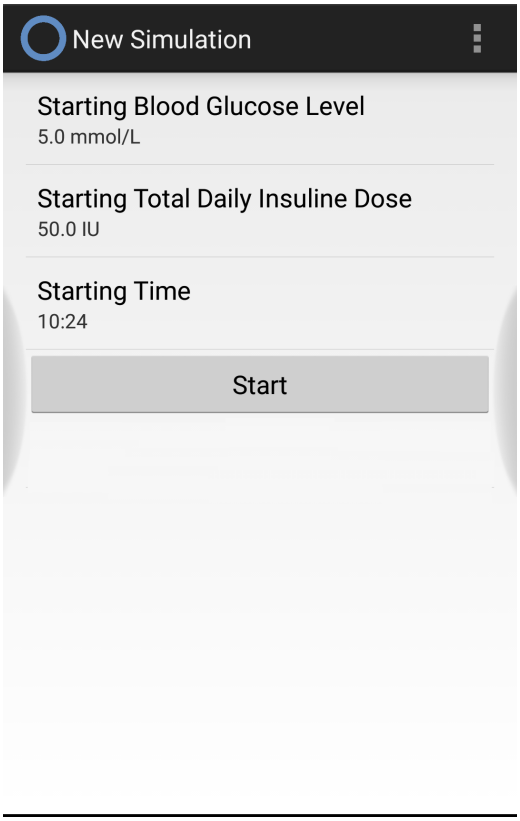


Figure 41: New Simulation screen of the prototype v0.004a.

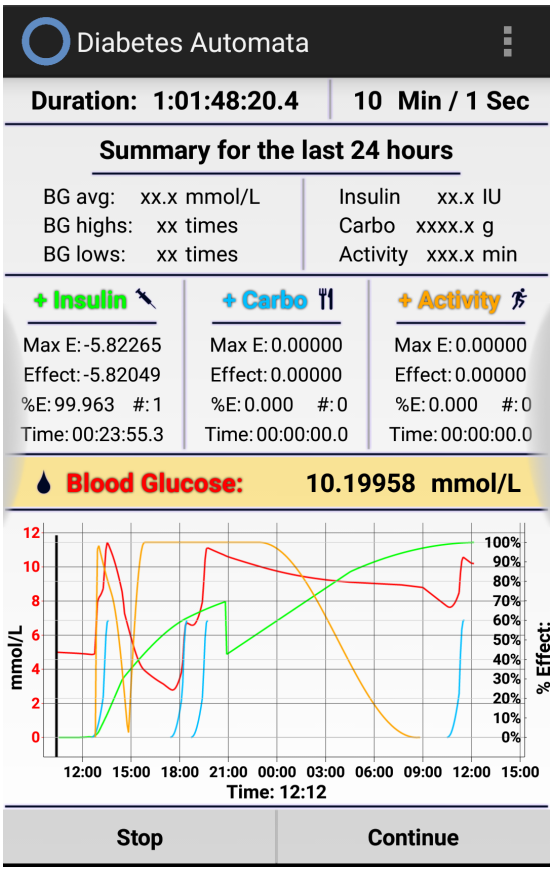


Figure 42: The Main screen of the prototype v0.004a with paused simulation, high blood glucose.

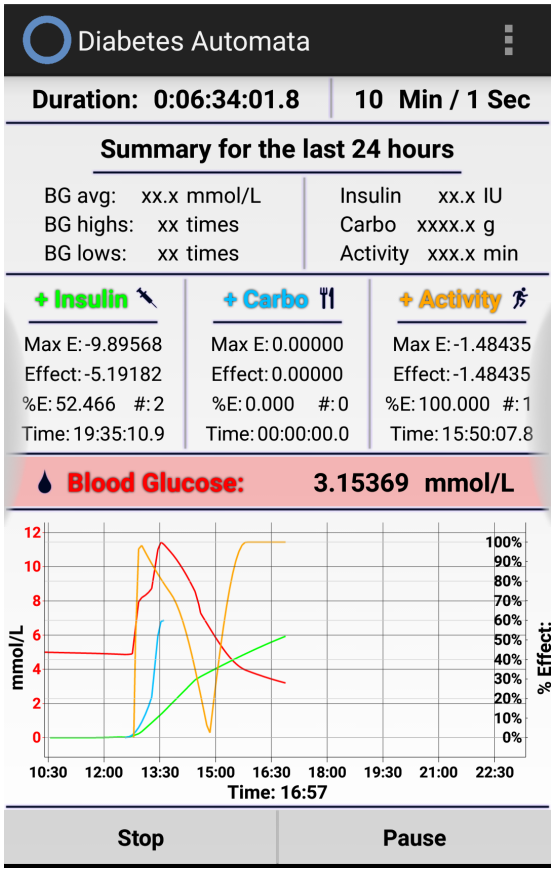


Figure 43: New Simulation screen of the prototype v0.004a with running simulation, low blood glucose.

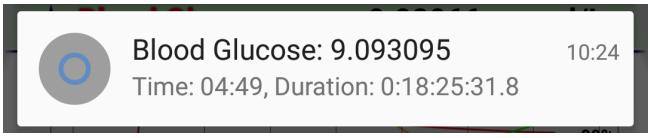


Figure 44: Notification of the prototype v0.004a with running simulation.

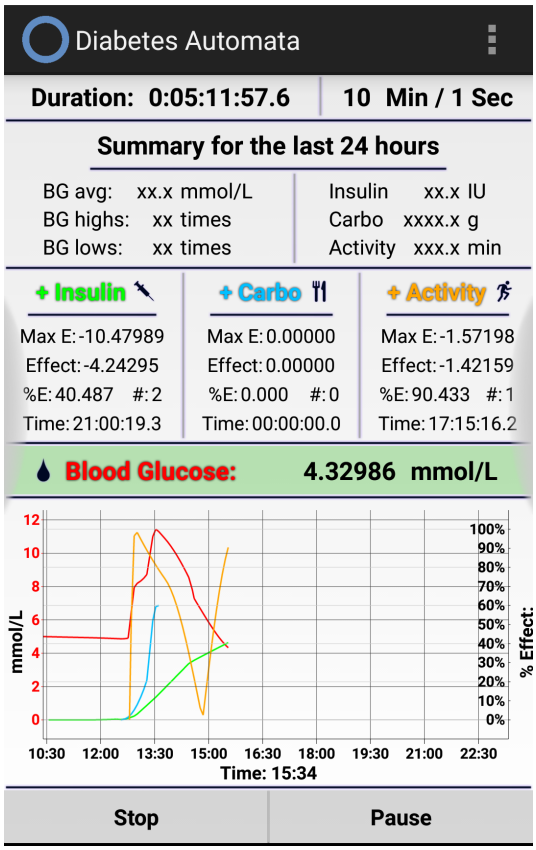


Figure 45: The Main screen of the prototype v0.004a with running simulation, normal blood glucose.

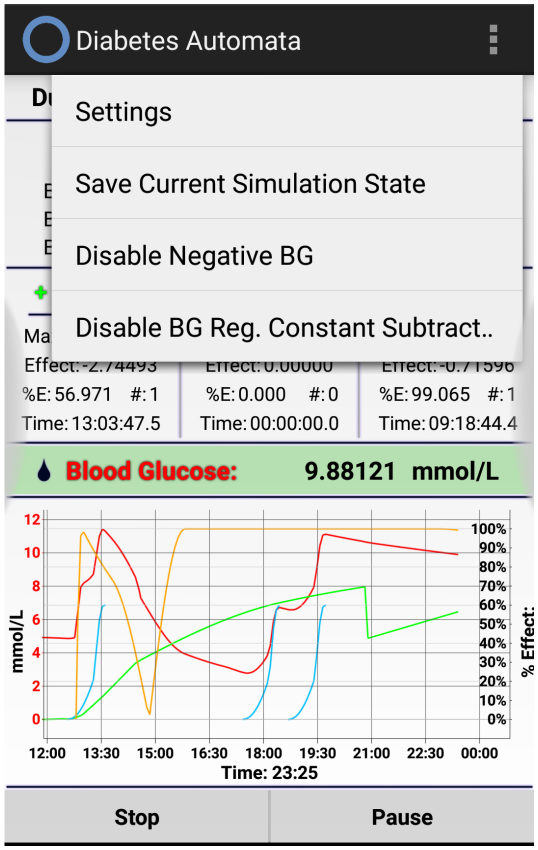


Figure 46: The Main screen of the prototype v0.004a with running simulation and options.

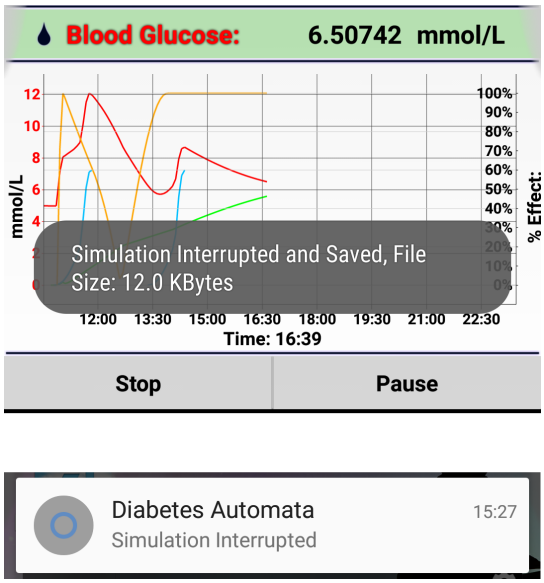


Figure 47: Prototype v0.004a, interrupted simulation, part of the Main screen (upper) and notification (lower).

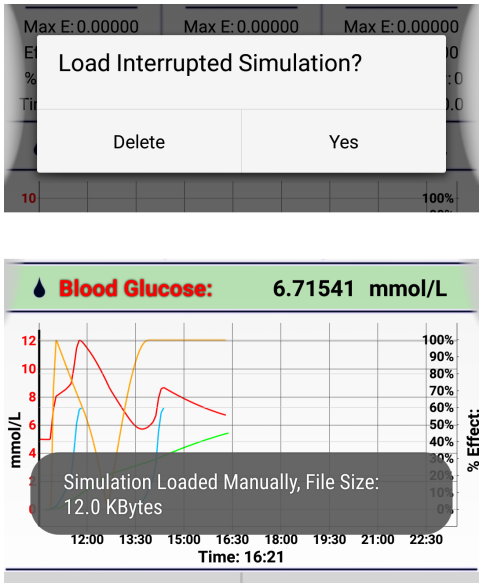


Figure 48: Prototype v0.004a, "Load Interrupted Simulation" dialog (upper) and loaded simulation (lower) on the part of the

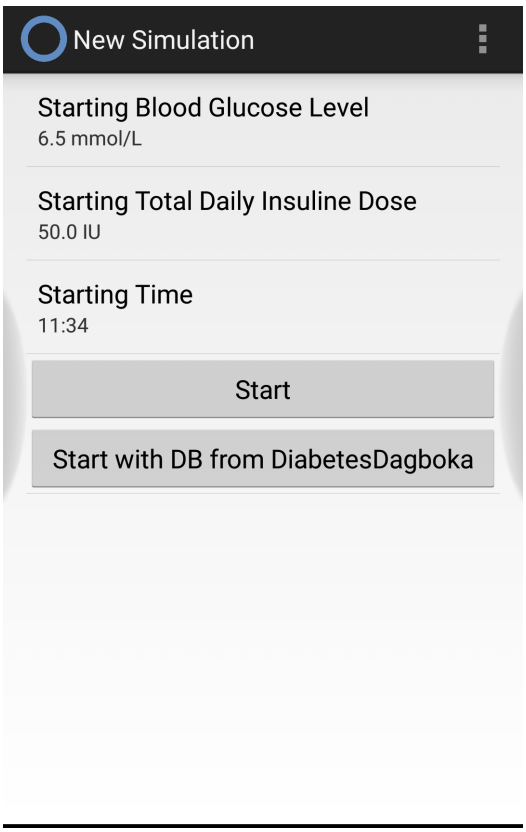


Figure 49: New Simulation screen of the prototype v0.006b.

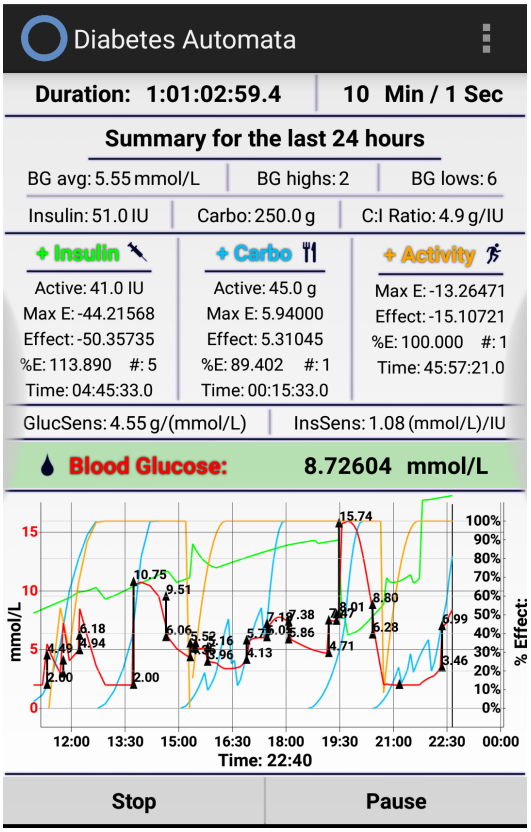


Figure 50: The Main screen of the prototype v0.006b with running DD database simulation.

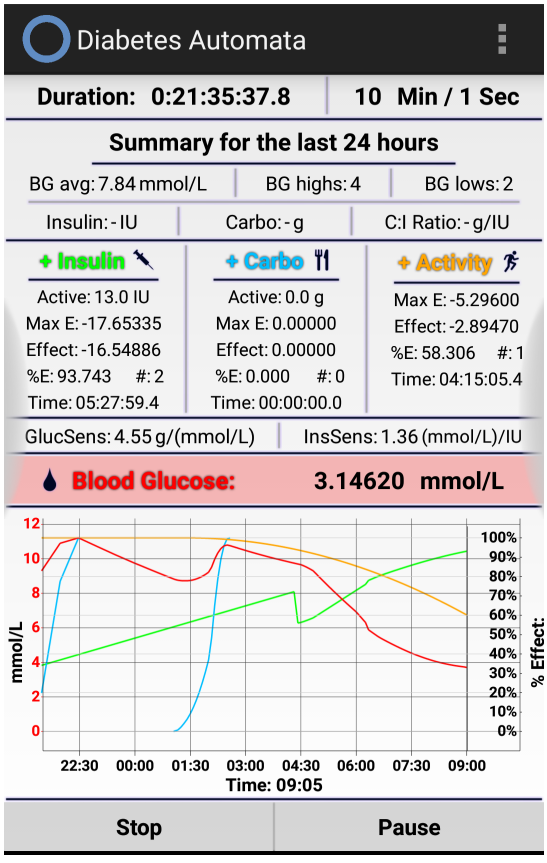


Figure 51: The Main screen of the prototype v0.006b with running simulation.

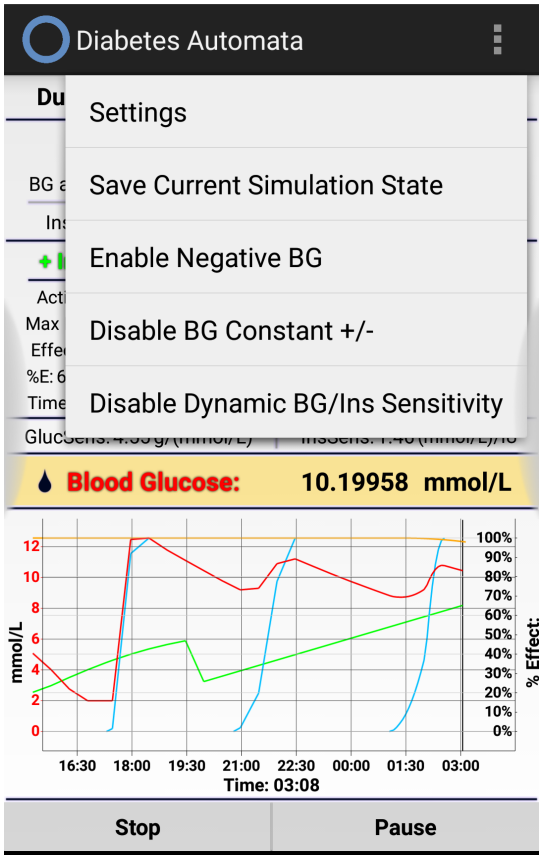


Figure 52: The Main screen of the prototype v0.006b with running simulation and options.

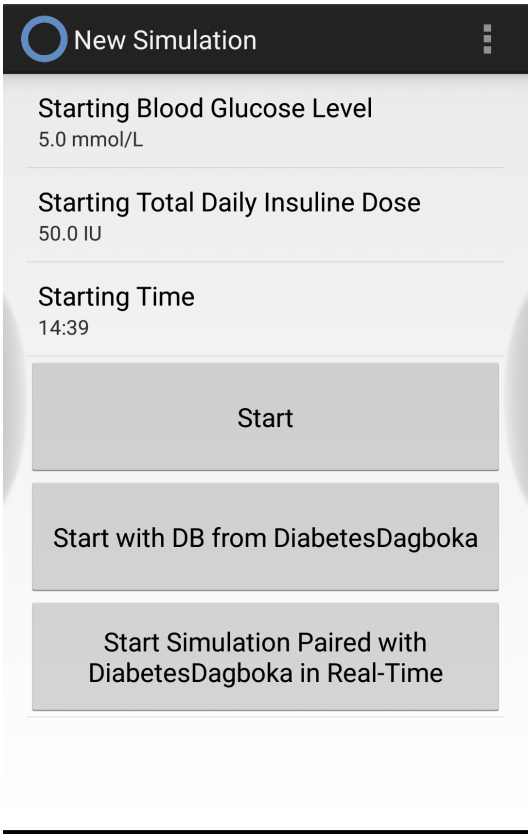


Figure 53: New Simulation screen of the prototype v0.007d.

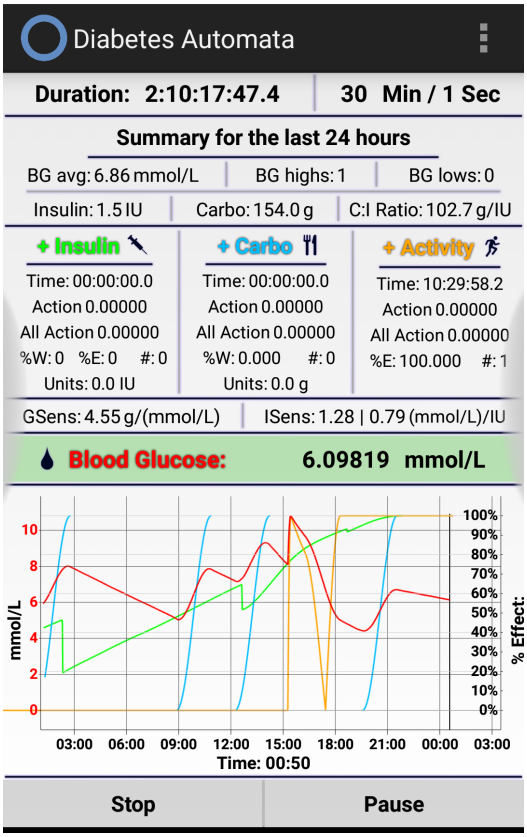


Figure 54: The Main screen of the prototype v0.007d with running simulation.

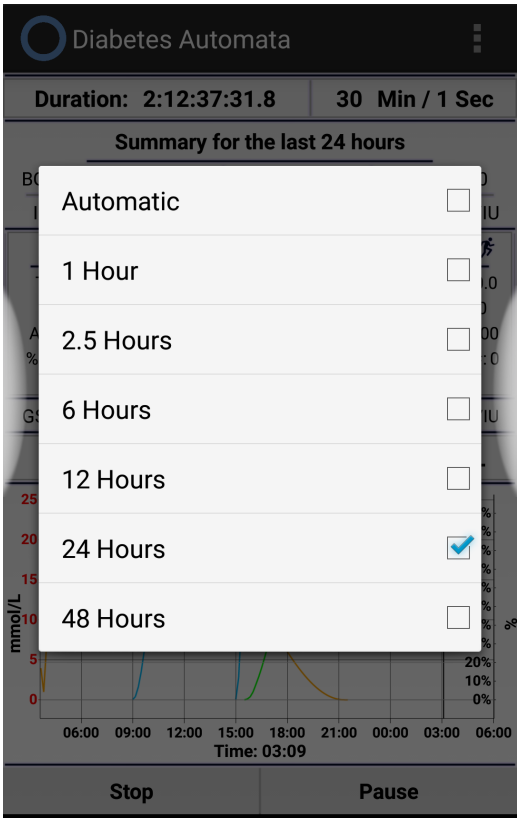


Figure 55: Graph Size menu on the Main screen of the prototype v0.007d.

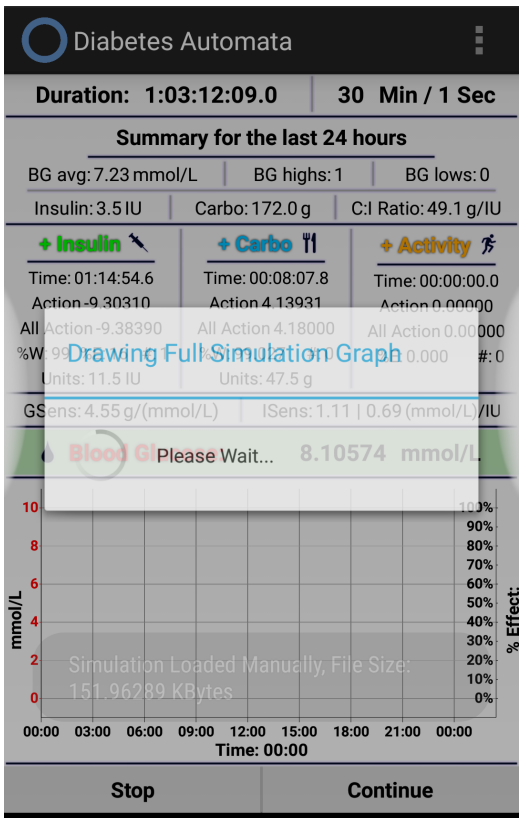


Figure 56: Prototype v0.007d, Drawing Full Graph Please Wait message after loading

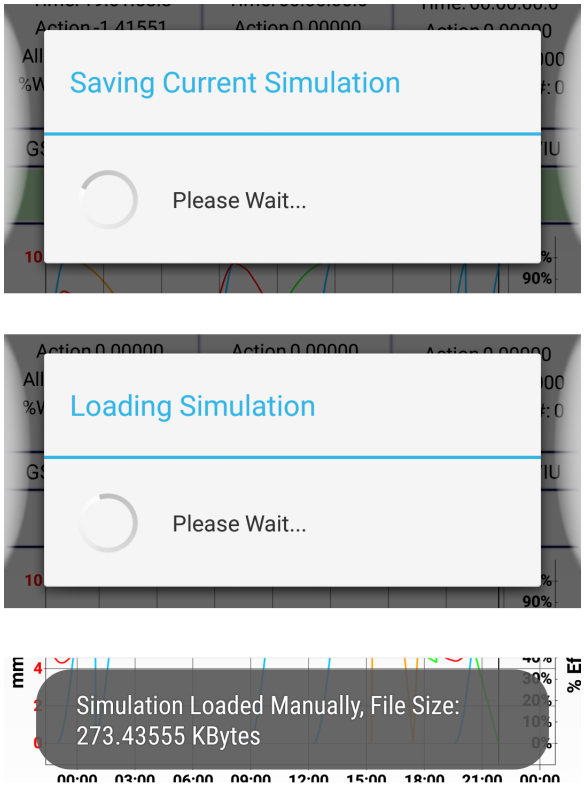


Figure 57: Saving Simulation (upper) and Loading Simulation (middle) windows, Simulation Loaded message (lower), on the part of the Main screen of the prototype v0.007d.

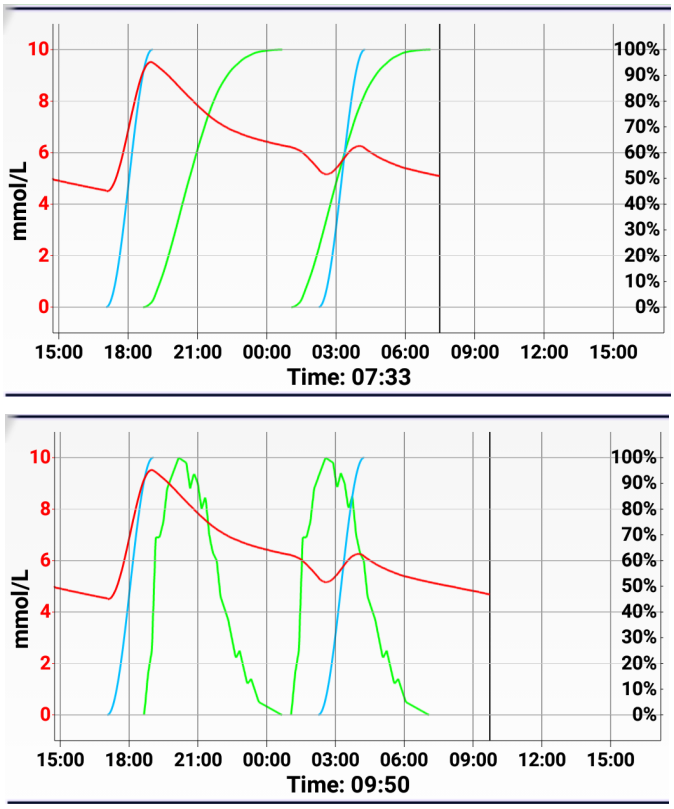


Figure 58: Prototype v0.007d, the difference between graph mode 1 (upper, Insulin (green) is presented as the work done in %) and graph mode 2 (lower, Insulin is presented as the action strength in %) (see sections 6.6 and 6.5 for more information).

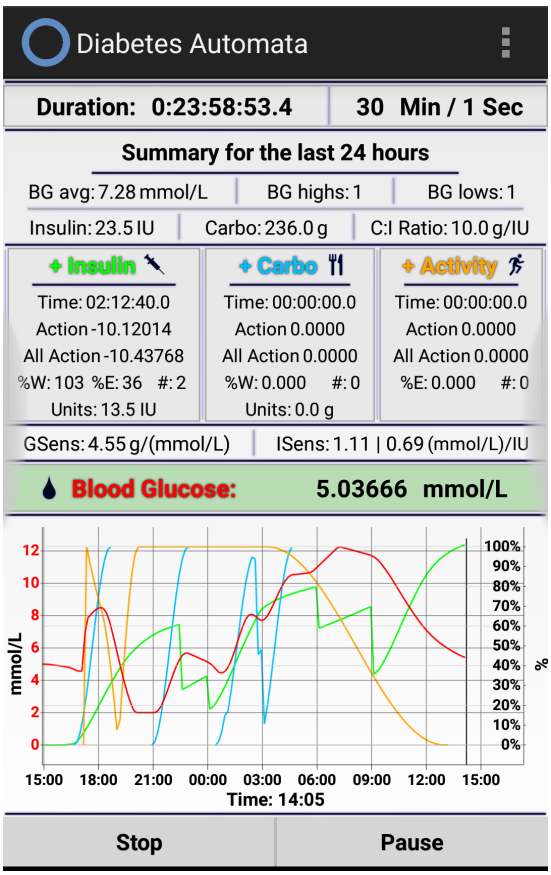


Figure 59: The Main screen of the prototype v0.008g with running simulation.

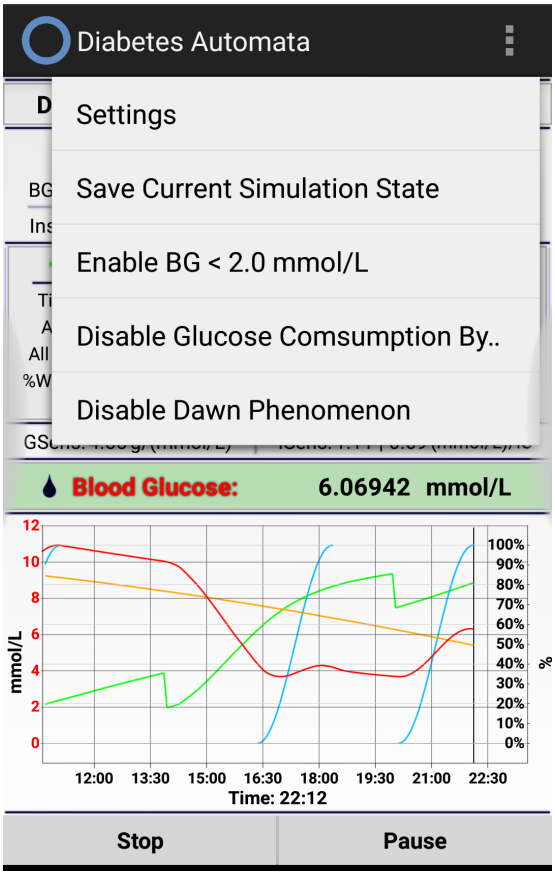


Figure 60: The Main screen of the prototype v0.008g with running simulation and options.

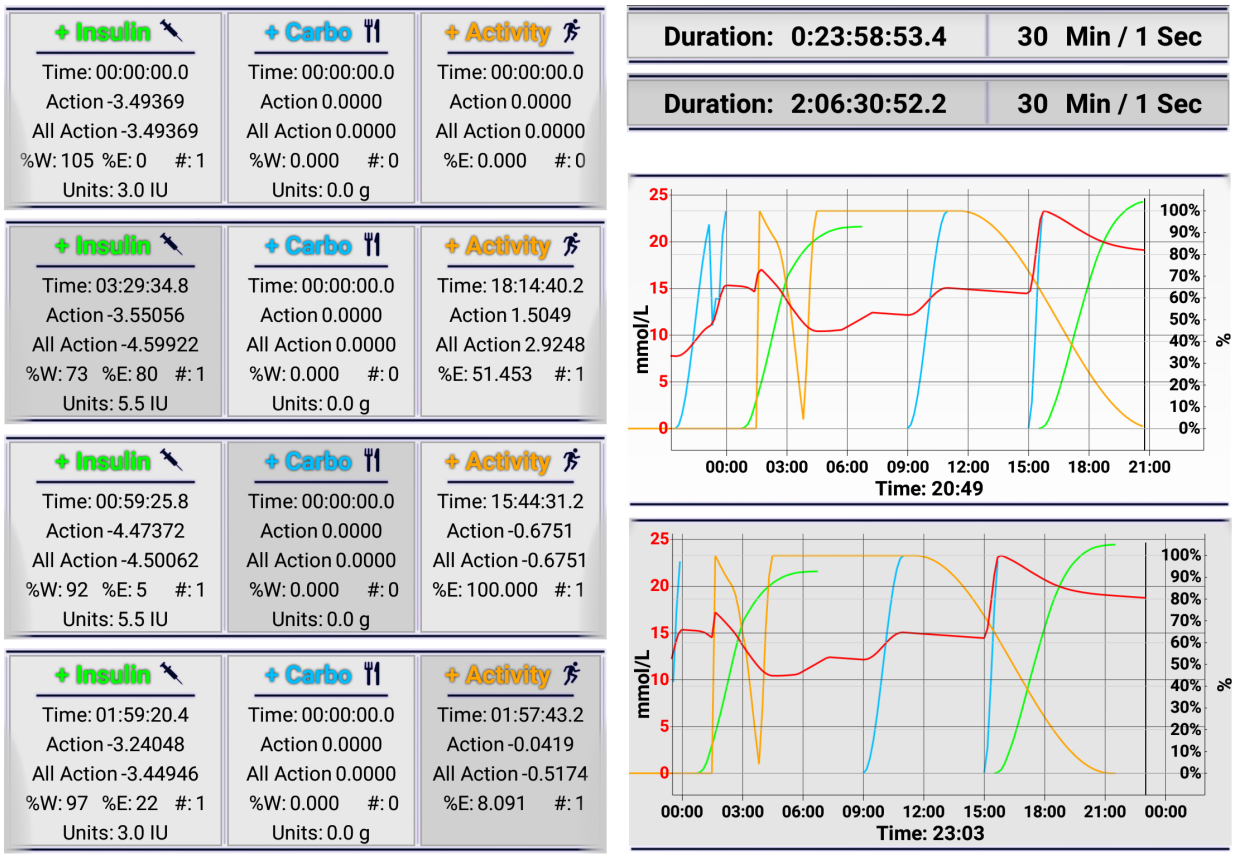


Figure 61: Cliclable elements of the Main screen of the prototype v0.008g in pressed and not pressed state.

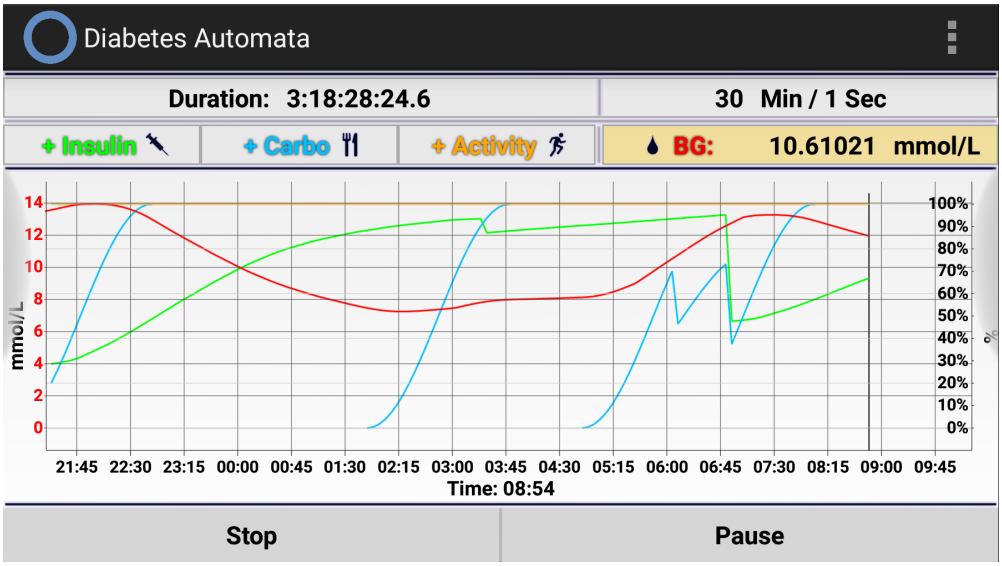


Figure 62: Prototype v0.008g, the Main screen in landscape mode with running simulation.

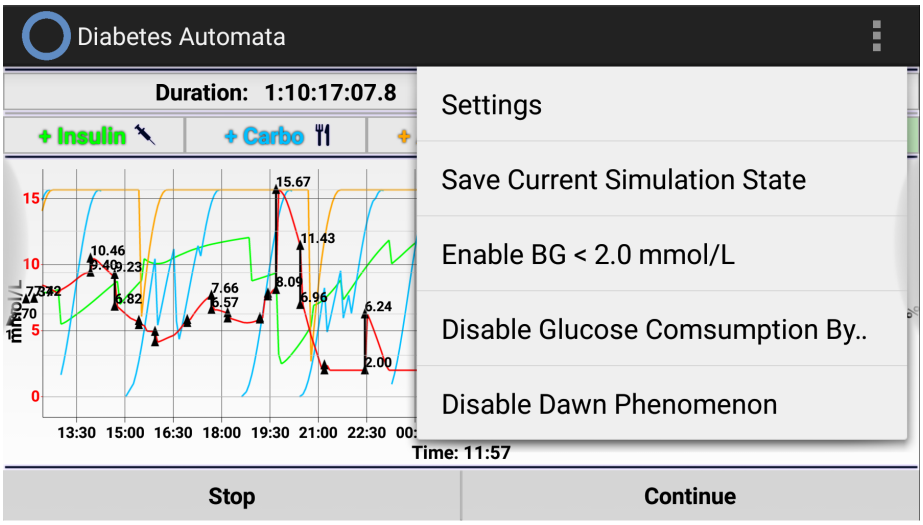


Figure 63: The Main screen of the prototype v0.008g with running simulation and options, landscape mode.

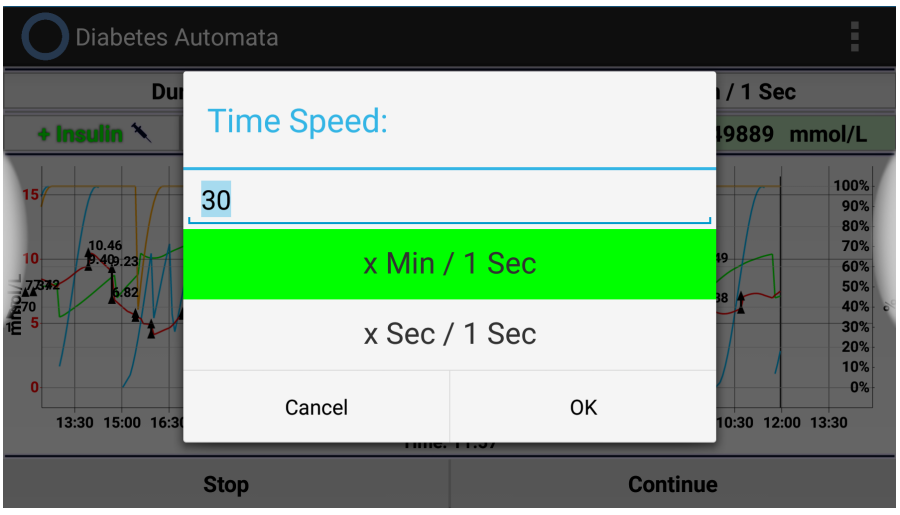


Figure 64: The Main screen of the prototype v0.008g with running Diabetesdagboka database simulation, setting new Time Speed, landscape mode.

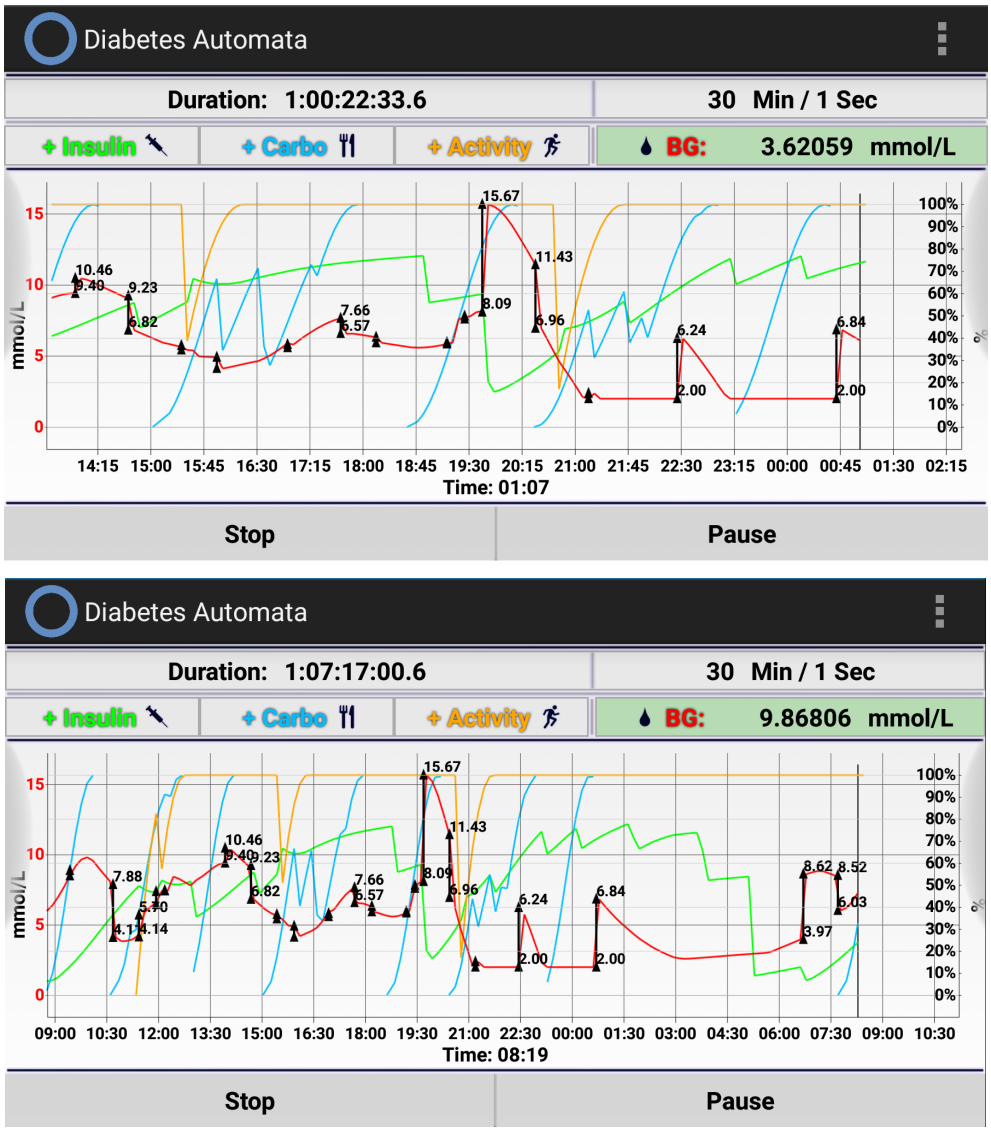


Figure 65: Prototype v0.008g, the Main screen in landscape mode with running Diabetesdagboka database simulation, difference between graph size 12 hours (upper) and 24 hours (lower).

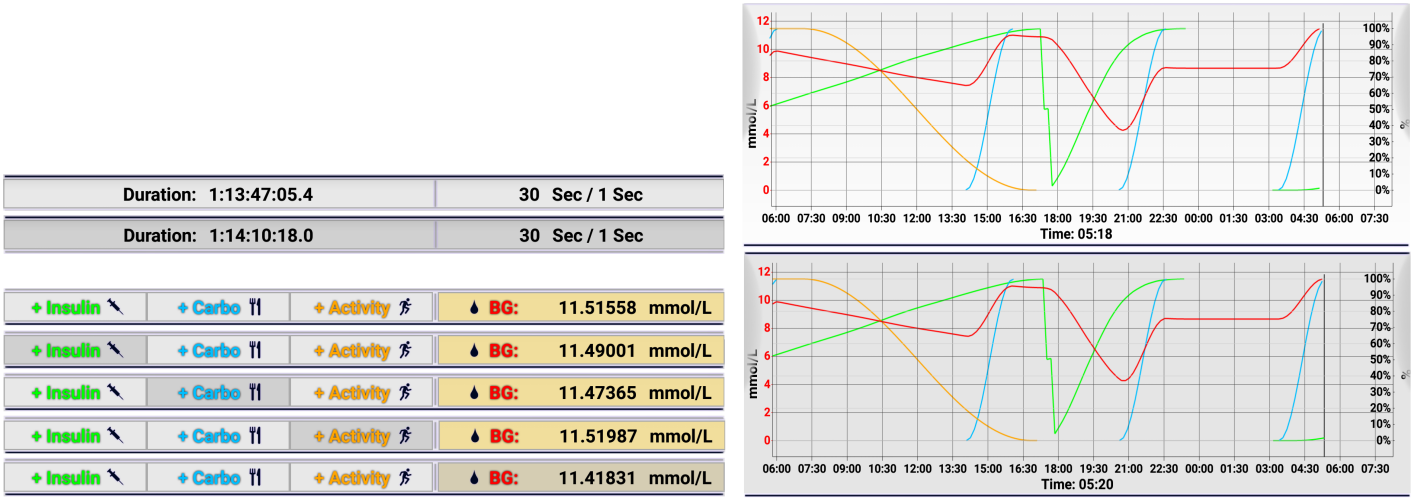


Figure 66: Cliclable elements of the Main screen of the prototype v0.008g in landscape mode, pressed and not pressed state.

For more information about how the prototype works and how to use it, as well as the description of the elements on all screens of the last version, see section 6.6. The history of prototype versions that describes the development progress and the changes made from a previous version, is presented in Appendix C: Prototype Version History.

5.3. Communication And Interaction Between External UI And Engine

The Diabetes Automata engine provides API for external applications. The API is used for the simulation control, communication and interaction between the UI and the engine. It is aimed to provide the external UI opportunities to access the simulation State, change various engine parameters, add new insulin/carbohydrate/activity simulation records, fetch simulation details to be displayed, start/pause/stop the running simulation, etc. The implementation of the API is presented in sections 6.5.3 and 6.5.5.3.

The Main module presented in Figure 20 contains one part of the API. The part provides access to various simulation parameters, simulation operations and control. Another part of the API resides in Settings and provides access to the biometric parameters (see Figure 23 and Figure 24).

5.4. Testing

Three kinds of prototype testing methods were designed. They are constant feedback from testers during the development, simulation trial by testers, and testing of the engine algorithms with testers' private data.

During the first kind of testing, the testers get new versions of the prototype, run them, and give feedback about how accurate the blood glucose simulation algorithms work, as well the correctness of the behavior of particular modules such as insulin, carbohydrates and activity. Also, the feedback can be about bugs and usability, which gives opportunity to make the next version of the prototype be easier, more comfortable to use, and more stable. After the corrections and improvements of the prototype are made based on the provided feedback from testers, they get a new version of the prototype, and then the same steps are repeated.

During the second kind of testing, the test persons run the prototype during several days, register all information about insulin, carbohydrates and activity, and see how close the simulated blood glucose level is to the real level. In the end, they send the results to the author.

During the third kind of testing, the testers provide a sample database made in Diabetesdagboka, containing personal data – the history of their diabetes self-management. Additionally, they also provide extra information about insulin, carbohydrates and activity, such as the types of insulin they use, the pictures of food they have registered as a carbohydrate record in the Diabetesdagboka database (in order to determine correct GI), and types of activities done during the testing period. The extra information is used to hardcode these important details that cannot be registered in Diabetesdagboka. After receiving the database and the additional details, the author runs the simulation of the records fetched from

the provided database, gets the simulation results and analyzes the deviations between the measured blood glucose level and the simulated level.

During the second and the third kind of testing, the developed Demonstrator is used as a testing tool.

5.5. Summary

In this chapter, we have discussed the design of the two parts of prototype – the Diabetes Automata engine and the Demonstrator. In particular, we have discussed the architecture of the engine, the general way how it should work, and why the described architecture was chosen. Also, the mockups of the Demonstrator were presented, as well as the screenshots from different prototype versions, which partially show the progress and the changes during the development. Finally, general words about the interaction between the engine and the UI were said.

6. Implementation

The engine is a Java-project that can be used by external diabetes-related applications. The demonstrator is an Android application project that uses the engine and demonstrates what it can do. First, we discuss tools used in the development of the engine and the demonstrator – Java programming language and Android development framework. After that we discuss the development of the engine and the UI described in the chapter 5 (5.1 – 5.3).

6.1. Java Programming Language And Platform

Diabetes Automata engine is developed using Java programming language and computing platform. Java was developed by Sun Microsystems and released in 1995 (Java.com, 2015) as Java 1.0. Java was started by the "Green Team" Sun engineers in 1991 as a project called “Oak”, with the goals to implement a virtual machine and a purely object-oriented programming language with C-like syntax, that would have greater uniformity and simplicity than C or C++ (JavaTalk, 2014; Woodger Computing Inc., 2001; Oracle.com, 2011b; Oracle.com, 2011a).

Java technology is used to develop applications for a big range of environments – “From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet” (Java.com, 2015).

The components of Java technology are Java programming language, Java compiler, Java virtual machine, garbage collector, Java development kit and Java runtime environment (Perry, 2010).

Java language is C-like derivative, its paradigm is based on the concept of object-oriented programming. On the top of the Java language structure are packages, which are Java’s namespace mechanism. Packages contain classes, which contain methods, variables, constants, etc. (Perry, 2010).

Java compiler is used to check the programming code and compile it into bytecodes that standard instructions for virtual machine (Perry, 2010).

Java virtual machine interprets bytecodes, like a CPU would interpret assembly language. JVM is a piece of software available for various chipsets, and is a heart of the Java’s “write one, run anywhere” principle (Perry, 2010).

Garbage collector is the Java implicit memory management approach that provides automatic memory allocation for objects and reclaiming memory from them when the application doesn’t need them anymore (Perry, 2010).

Java development kit is a complete class library of prebuilt utilities (Perry, 2010), the Java APIs (application programming interfaces). These libraries are available for programmers to use in the development of their applications – they let them add ready-made and customizable functionality to save the development time (Pawlan, 1999). “Java supports millions library classes which you can be imported while coding. These libraries provide billions of methods and these methods help programmer to develop a well optimized and readable code” (JavaTalk, 2014).

Java runtime environment is included in the Java development kit, is available for multiple platforms, and includes the JVM, code libraries, components that are necessary for running Java programs (Perry, 2010).

6.2. Android OS

The Diabetes Automata demonstrator is a mobile application developed for Android. Android is an open source software stack, a complete operating environment for a wide range of mobile devices that was created by Google and the Open Handset Alliance (Burnette, 2010; Source.Android.com, 2015; IBM developerWorks, 2011). It is based on Linux kernel version 2.6 with approx. 115 patches (TutorialsPoint.com, 2014) that provides core system services like process and memory management, networking, and other operating system services (Burnette, 2010). The system architecture is represented in Figure 67.

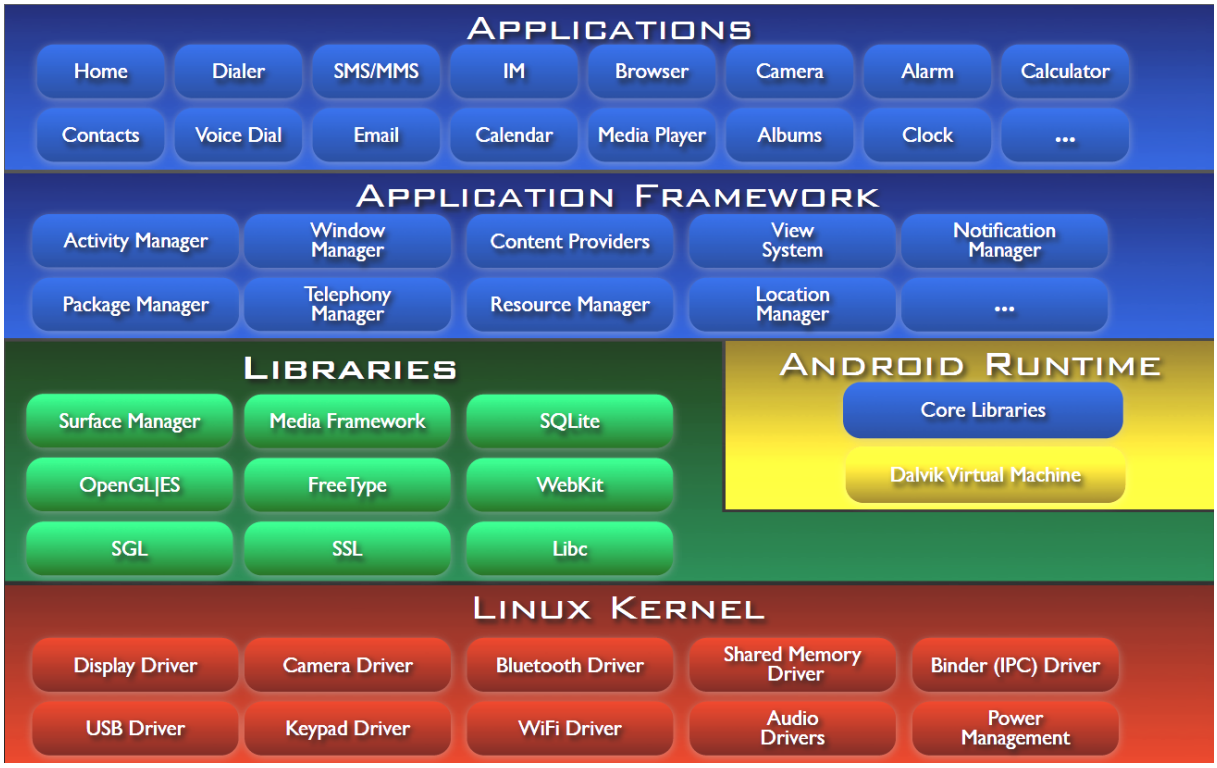


Figure 67: Android system architecture.
Source: (Brady, 2008, Slide 3)

"The software stack contains Java applications running on a virtual machine, and system components are written in Java, C, C++, and XML" (Butler, 2011).

Applications for Android are written in Java programming language. Android contains Dalvik virtual machine that is lying in the runtime layer and is heavily optimized (CPU/RAM/Battery-Optimized) for mobile devices (Android Development Community, 2007). Each application runs within an instance of the Dalvik virtual machine lying in the runtime layer, and each Dalvik instance runs within a Linux-kernel managed process (IBM developerWorks, 2011). See Figure 68.

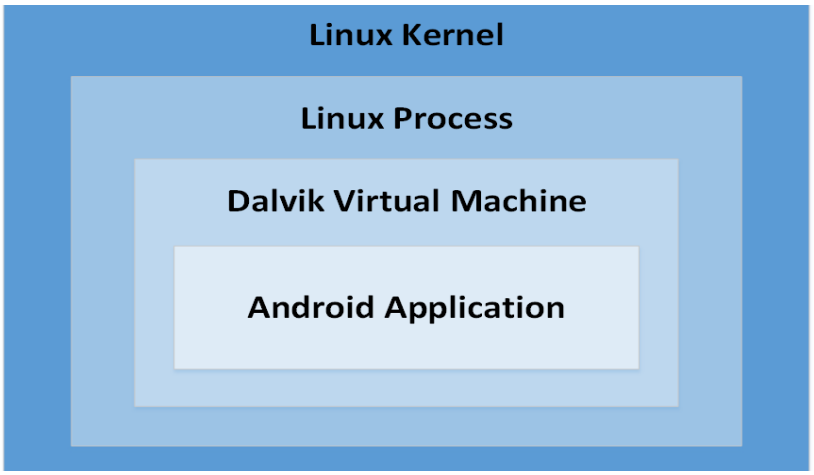


Figure 68: Android application within a Dalvik virtual machine instance.
Based on: (IBM developerWorks, 2011).

The Activity Manager that resides in Application Framework controls the life cycle of applications. When an application is started, it is brought to the foreground. Other applications or screens of the same application might be invokable from it. The sequence of invoked applications and screens is written to the Application Stack. Activity class represents each screen that has its life cycle. (Burnette, 2010)

The activity life cycle is represented in Figure 69.

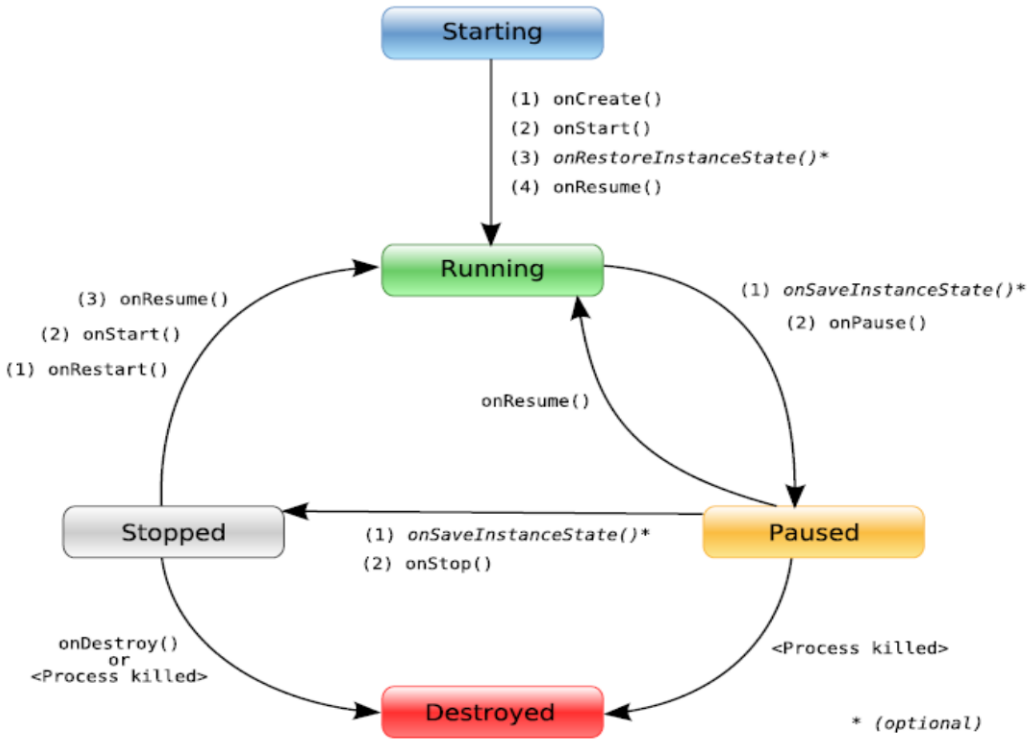


Figure 69: Activity life cycle.
Source: (Burnette, 2010, Figure 2.3)

When the state of activity changes as shown in the figure above, the following methods can be automatically called by the system depending on the state (these methods must be overwritten by programmer) (Burnette, 2010):

- onCreate: called when the activity first starts up;
- onStart: called when the activity is about to be displayed to the user;
- onResume: called when the activity can start interacting with the user;
- onPause: called when the activity is about to go to the background, for example if another activity has been launched in front of it;
- onStop: called when the activity is no longer visible;
- onRestart: called when the activity is being redisplayed from a stopped state;
- onDestroy: called before the activity is destroyed;
- onSaveInstanceState: called to save per-instance state. Its default implementation automatically saves the state for all user controls, so the programmer doesn't have to overwrite it.
- onRestoreInstanceState: called when the activity is being reinitialized by from the previously saved by "onSaveInstanceState" state. Its default implementation automatically restores the state of the last UI.

The unveiling of Android took place on 5th November 2007 (Hee-Yeon et al., 2010), the last version of Android is 5.1 (SocialCompare.com, 2015). Sometimes, developers have to optimize or update their software in order to make it compatible with a new-coming version of Android. During the development of this project, such problem has occurred. The problem is mentioned in sections 7.1 and 3.6.

6.3. Android Software Development Platform

Diabetes Automata is developed in Eclipse IDE for Java Developers Version: Luna Release (4.4.0) with installed Android Developer Tools (ADT) plugin.

Eclipse is "an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle" (Eclipse Foundation, 2015).

ADT is "a plugin for Eclipse that provides a suite of tools that are integrated with the Eclipse IDE" (Developer.Android.com, 2015a). It offers a developer "access to many features that help you develop Android applications. ADT provides GUI access to many of the command line SDK tools as well as a UI design tool for rapid prototyping, designing, and building of your application's user interface" (Developer.Android.com, 2015a), where SDK is software development kit.

6.4. Class Diagram For The Entire Project

The class diagram for the entire project is presented in Figure 70. More detailed implementation details are presented further in sections 6.5 and 6.6.

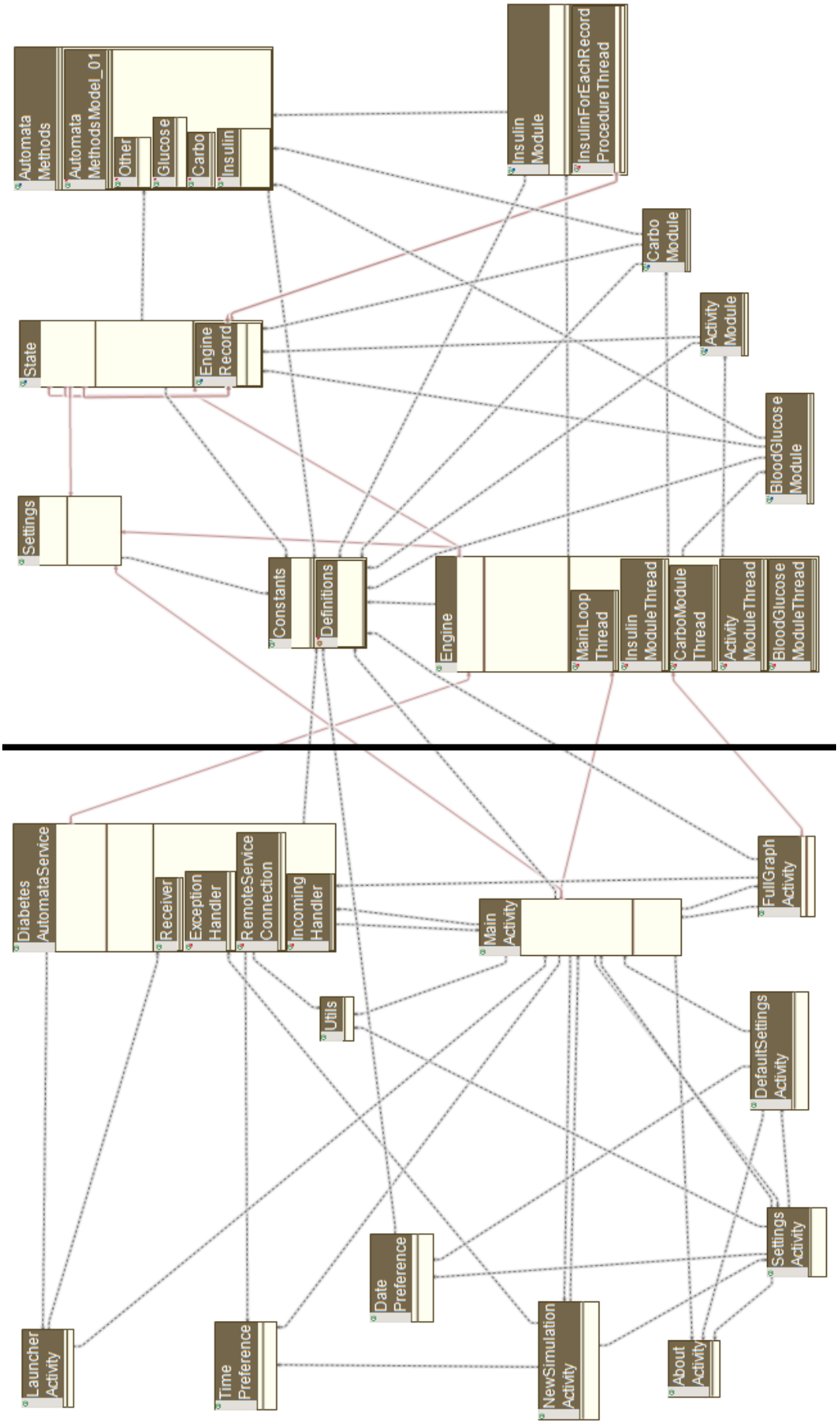


Figure 70: Class diagram for the entire project. Android UI components are to the left from the black line, Engine components are to the right from the black line.

6.5. Diabetes Automata Engine

Diabetes Automata Engine is developed Java programming language. In this section we discuss how it was developed and implemented. In particular, we discuss the implementation of each part of the Engine. The diagram for the Engine is presented in Figure 71.

The group of Insulin, Carbohydrates, Activity and Blood Glucose modules is presented in Figure 73. During the simulation, the simulation "infinite" loop (6.5.5.1), the Insulin module (6.5.6), the Carbohydrate module (6.5.7), the Activity module (6.5.8), and the Blood Glucose module (6.5.9), run in parallel in their own separated threads.

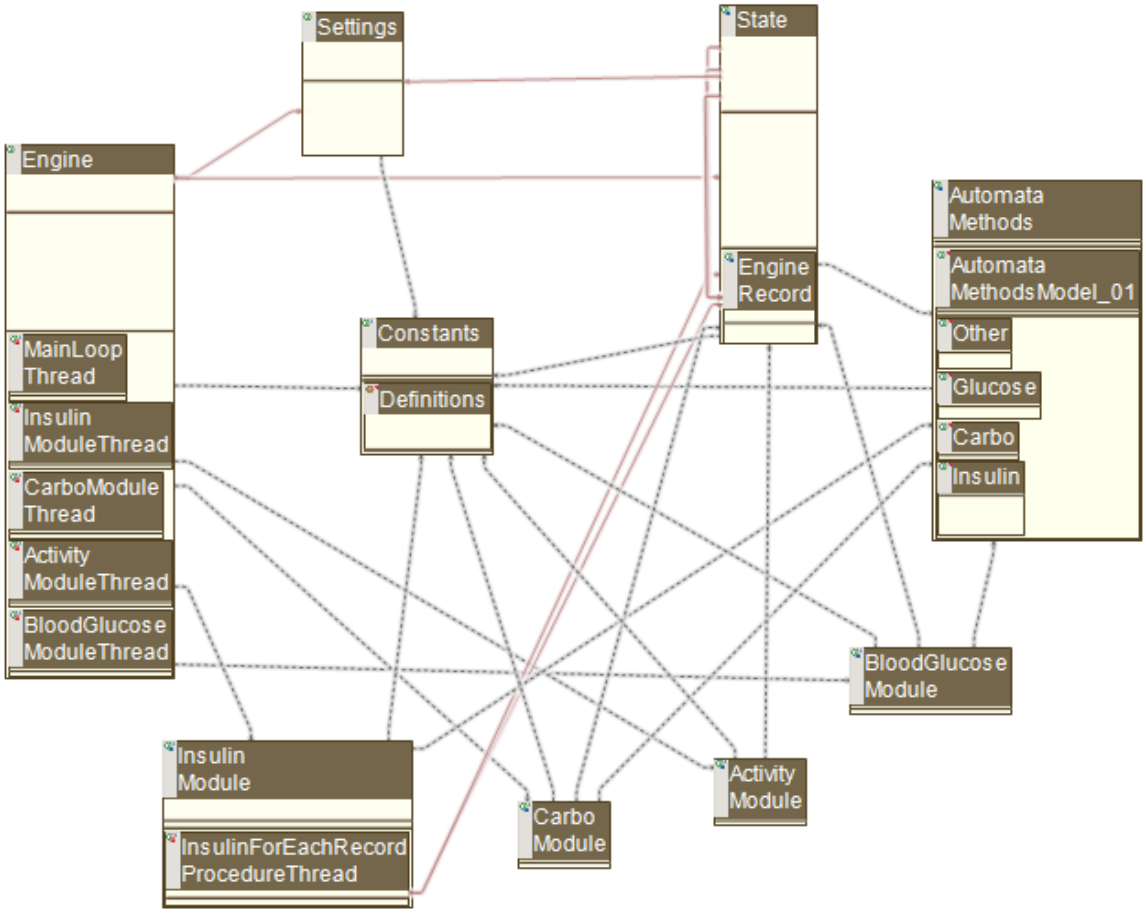


Figure 71: Class diagram of the Engine.

6.5.1. Collection Of Formulas And Equations

The collection of formulas and equations (some of them are presented in 2.3 and are implemented as presented or were modified) used in the Engine are placed in a static class. It is constantly used by Insulin, Carbohydrate, Activity and Blood Glucose modules for making necessary calculations. The class is represented in Figure 72. Some of the class methods were used in previous prototype implementations and are not used in the current version (however, the author decided to keep them, for the case if they will be needed in the future).

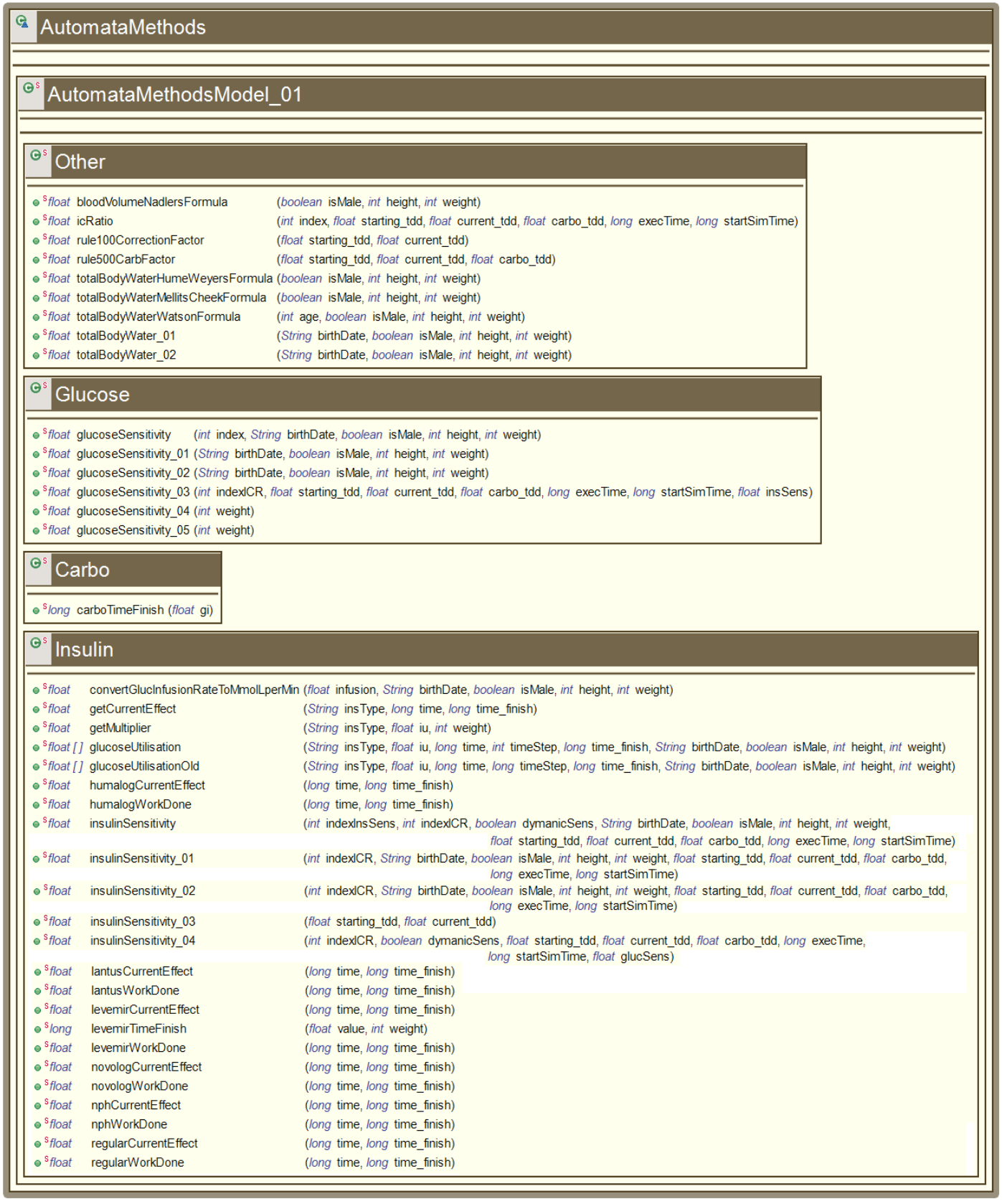


Figure 72: AutomataMethods class.

6.5.2. Constants

Constants is a static public class that contains general constants used by the Engine. The constants can be used by external applications as well, and they are used by the Android UI. The class contains the following constants and definitions:

- Number of milliseconds in 1 minute, 1 hour, 1 day and 30 days;
- Molar mass of glucose, 180.15588 g/mole;
- The values of GI 100, GI high, GI medium, GI low;
- Low blood glucose (4 mmol/L), very low blood glucose (3 mmol/L), the smallest possible blood glucose (2 mmol/L), high blood glucose (10 mmol/L), very high blood glucose (15 mmol/L);
- ENUM data structure with definitions: male, female, GI low, GI medium, GI high, GI 100, Humalog, Novolog, Regular, NPH, Levemir, Lantus, aerobic activity, anaerobic activity, activity duration short, activity duration medium, activity duration long.

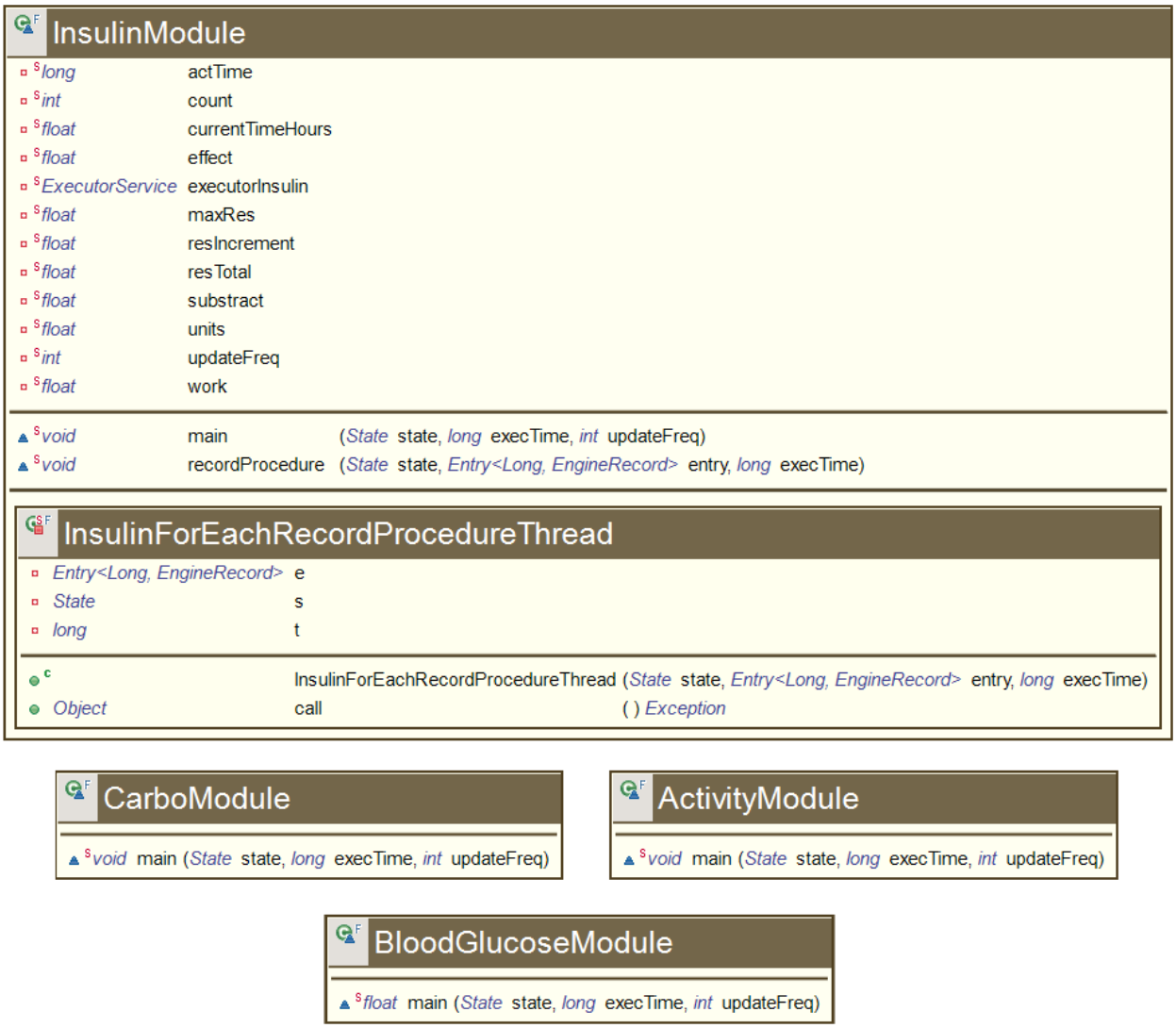


Figure 73: Insulin Module, Carbo Module, Activity Module and Blood Glucose Module classes.

6.5.3. Settings

Settings is a public class and is accessible outside the Engine. It holds the global simulation parameters, such as the frequency for every engine "infinite" loop iteration, the time-speed multiplier (which equals to 1 when the simulation is real-time, or 60 when the speed is 1 minute per second) (1 by default); the flags for enabling or disabling dawn phenomenon simulation (enabled by default), constant glucose consumption by brain (enabled by default) and blood glucose levels lower than 2 mmol/L (disabled by default); and biometric information about the user (which can be seen in Settings and Default Settings, see sections 5.2 and 6.6.2).

The class is represented in Figure 74. It provides a set of public "get" and "set" methods in order to provide public access to get and change the variables.

The public methods (marked with green circles in the right part of Figure 74) is a part of API. Another part of the API is provided by the "Engine" class (see 6.5.5.3).

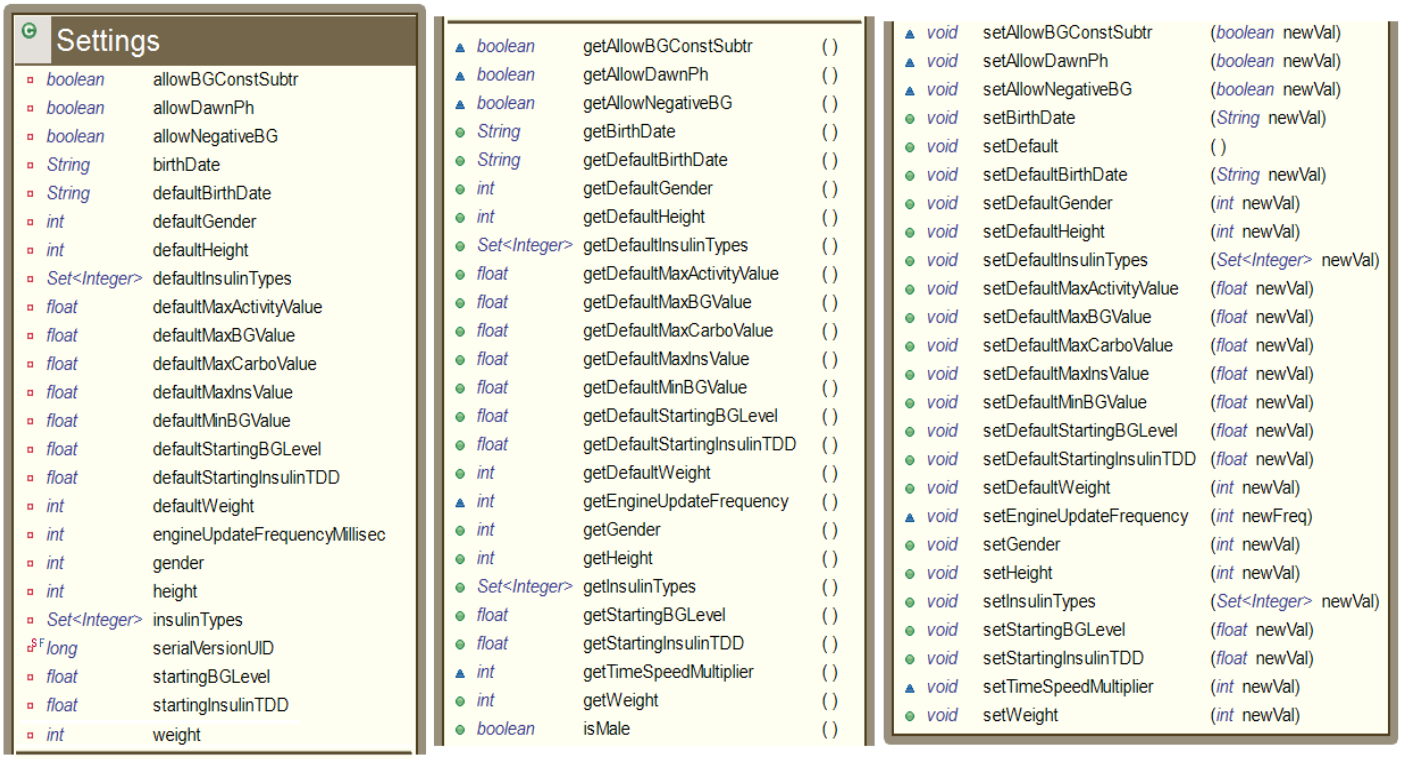


Figure 74: Settings class: variables (left colomn) and methods (middle and right colomn).

6.5.4. State

State is a private class that holds all data for the particular simulation, together with a "Settings" object. It keeps data only for the last 30 simulation-days, the older data is deleted from the State. The class diagram is presented in Figure 75.

Figure 75: State class: variables (left colomn), methods (middle and right colomn) and inner sub-class EngineRecord (lower).

State is not accessible from outside the engine, but the engine class contains a part of the API that provides sufficient access for an external game/simulator/demonstrator application to "get" and "set" some of the engine and simulation parameters and data in the State. See section 6.5.5.3 for more information.

Insulin, Carbohydrates, Activity and Blood Glucose modules operate with their own set of variables via "get" and "set" methods. One of such variables are collections of simulation records called "Engine records", which hold insulin, carbohydrates, activity and blood glucose records made during the simulation, together with some additional simulation data. Other module variables are current active time, current effect, maximal effect, work done, etc. The variables that don't belong to any of the four modules mention earlier, are various simulation and execution variables required for the simulation process.

Every new simulation starts with all state variables (except the Settings object) reset to 0 (or a new empty object, if the variable is not of a simple datatype such as int, float, long, etc.).

6.5.5. Main Module

Main Module (or as it is called in the code – Engine) is a public class and is accessible outside the engine. This is the main engine module. External application must create an Engine object in order to use it and run a simulation. The structure of the module is presented in Figure 76.

The main engine module has 3 major functions: the "infinite loop"; the simulation speed management; and the API for accessing the engine State – a set of public methods required to provide sufficient access for the game/simulator/demonstrator application to read, write and update various simulation parameters and data in the engine State.

6.5.5.1. The "Infinite Loop"

The "infinite loop" is the main simulation loop that is launched then user starts a new simulation or continues paused simulation. It runs in its own thread. It calls insulin, carbohydrate, activity and blood glucose modules and makes updates and preparations for the next iteration, with constant frequency value stored in "Settings".

The modules execution frequency is defined in the State and can be changed by the simulator. Every loop iteration the modules are called in the following sequence: insulin module, carbohydrate module, activity module and finally blood glucose module. After that, the engine thread is set to sleep during the amount of time defined in the frequency settings. After that, the thread is awake again and makes the next loop iteration.

If the "pause" command is sent from the simulator while the engine status is "running", the "infinite loop" is stopped. If the "pause" command is sent while the engine status is "paused", it starts the loop again, with the engine state from the paused simulation. If the "stop" command is sent from the simulator, the "infinite loop" breaks the simulation and resets the engine state.

Figure 76: Engine class: variables and sub-classes (left colomn) and methods (middle and left colomn).

6.5.5.2. The Simulation Time/Speed Management

After the "infinite loop" calls the modules, it updates various engine and state variables in order to prepare for the next iteration, one of such variables is the current simulation time that is updated using the following code:

```
setExecutionTime((System.currentTime() - startTime) * getTimeSpeedMultiplier());
```

where `System.currentTime()` returns the current time, `startTime` is the time when the simulation has started, and `getTimeSpeedMultiplier()` returns the multiplier that provides opportunity to speed up the simulation time in relation to the real time (for example if the multiplier equals 2, then the engine runs twice faster and it takes 1 second of the real time to simulate 2 seconds of the simulated time; or it could be set to 60, so that 1 minute is simulated per second) and can be changed by the external UI. In order to make the time management work correctly, some algorithm had to be implemented. During the development, the following issue raised up: when the UI changes the time multiplier, we want the time to speed up/down right from the particular point in simulated time.
Example:
The multiplier equals 2, so we go with the speed of 2 seconds per second, the current simulation time stopwatch shows 30 simulated seconds (so in 1 second it would become 32 seconds), which means that the simulation was started by the user 15 seconds ago. Right at this moment, we set the multiplier to 3, so in one second the simulated time should become 33 seconds. But it didn't. Instead, it would become 48 seconds. Why? Because the whole simulated time would get multiplied, and we would get 15*3=45 seconds plus 3 "new" simulated seconds.

In order to fix this issue, the following code had to be applied inside the "`setTimeSpeedMultiplier`" method (the method that is a part of the API and that provides opportunity for the external UI to change the current simulation time speed):

```
change = getTimeSpeedMultiplier() / newMultiplier;
realTime = getExecutionTime() / getTimeSpeedMultiplier();
if(change > 1.0)
{
	startTime = startTime - (realTime * change - realTime);
}
else if(change < 1.0)
{
	startTime = startTime + (realTime - realTime * change);
}
stateSetTimeSpeedMultiplier(newMultiplier);
```

This chunk of code changes the time when the simulation was started in order to compensate the overall time shift caused by changing the time multiplier. In case when the new multiplier is smaller than the previous one, the code substitutes the potential difference, otherwise it adds it. The result is that the new time speed is applied correctly from the particular moment in the simulation time when the time multiplier was changed.

The paper draft represented in Figure 77 shows what happens when the new time multiplier is bigger than the last time multiplier:

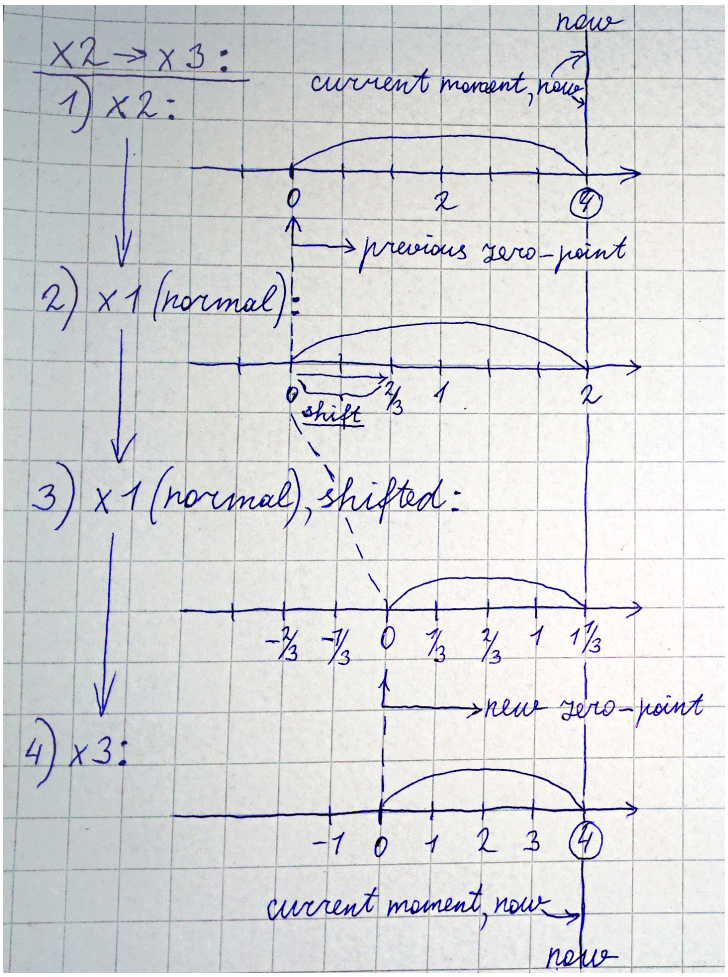


Figure 77: Time/speed management example, last < new:
Changing the time multiplier from 2 to 3, the current time value remails the same.

If we continue the example from above and set the same values, we get the following situation:
Assume that the simulation start time was 0 seconds, the current time speed multiplier is 2 seconds per second, the simulated time is 30 seconds (real time = 30/2 = 15 real time seconds), at this moment the multiplier is set to 3, the multiplier change equals 2/3 < 1. The start time changes and gets equal to (0 + (15 – 15 * 2/3)) = 5. We want the simulated time to be calculated and displayed correctly – 30 seconds, and we want the time to become 33 seconds in one second. Is the time management working correctly now? In order to check it, we need to go back to the following line of code in the “infinite loop”:

```
setExecutionTime((System.currentTime() - startTime) *
getTimeSpeedMultiplier());
```

If we replace the constituents with the values from our example, the new simulation time will become ((15 – 5) * 3) = 30 simulated seconds. And in one real time second, it will be ((16-5) * 3) = 33 simulated seconds.

The paper draft represented in Figure 78 shows what happens when the new time multiplier is smaller than the last time multiplier:

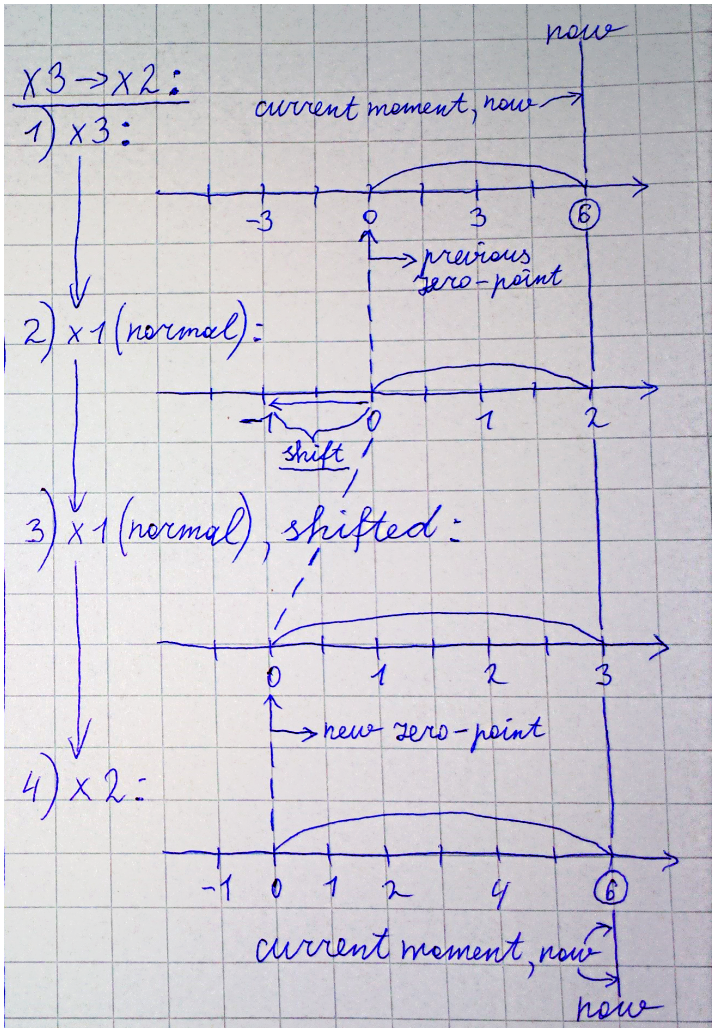


Figure 78: Time/speed management example, last > new:
Changing the time multiplier from 3 to 2, the current time value remails the same.

Another issue could rise up when the simulation is set to pause. When the simulation was continued, the time the engine has spent in the “pause” state could be added to the simulated time. This is simply solved by the following algorithm:
When the engine is paused, the pause-timestamp is written to a variable. When the simulation is continued, the simulation start time gets changed:

```
startTime = System.currentTime() - timeStamp + startTime;
```

Example:
Assume that the simulation start time was 0 seconds, the time speed multiplier is 2. The simulation has been running for some time and was set to pause when the simulated time was 30 seconds, so the real time (`System.currentTime()`) was equal to 15 seconds. If we

continue the simulation in 25 real-time seconds, we want the simulation to be continued from 30, not from 80 (30 + 25 * 2) seconds.
The pause-timestamp equals to 15. When we continue the simulation, the start-time changes to ((15 + 25) – 15 + 0) = 25. Will the simulation continue in correct way now?
In order to check it, we need to go back to the timing line of code in the "infinite loop" again:

```
setExecutionTime((System.currentTime() - startTime) *
getTimeSpeedMultiplier());
```

The simulated time becomes (((15 + 25) – 25) * 2) = 30 simulated seconds, exactly when we paused the simulation.

These are the main logics of how the simulation time/speed management algorithms are implemented in the engine.

6.5.5.3. The API For Accessing The State

As said above, the Engine's API is the set of public methods required to provide sufficient access for the game/simulator/demonstrator application to read, write and update various simulation parameters and data in the engine State.

The methods are presented in Figure 76 (middle and left colomn), the methods marked with green circles (which means "public"). Table 4 presents the description of them:

startSimulation	starts simulation if it is not already running
stop	stops running simulation
pause	requests simulation pause
isRunning	returns true if simulation is running, false otherwise
isOnPause	returns true if running simulation is paused, false otherwise
isFinished	returns false if simulation is running or if the process of stopping the simulation and resetting the engine state is not complete yet, false otherwise
getState	returns the engine state to the simulator as a raw object, for saving the simulation state
setState	sets the state object given in the arguments as the current engine state, for loading previously saved simulation
getSettings	returns the engine settings
updateStateVariables	for manual updating of statistic variables for the last 24 hours: insulin and carbohydrates consumed, blood glucose average, highs and lows; the engine updates them every loop iteration automatically
getExecutionTime	returns passed simulation-time
getExecStartTime	returns the timestamp when the simulation was started
getStartingTime	returns time in hh:mm when the simulation was started
setStartingTime	sets time in hh:mm when the simulation is about to be started
getUpdateFrequency	returns "infinite loop" iteration frequency

setUpdateFrequency	changes “infinite loop” iteration frequency
getTimeSpeedMultiplier	returns the value of the simulation speed multiplier
setTimeSpeedMultiplier	requests changing current simulation speed multiplier to the one given in the arguments
getInsRecords	returns the collection of insulin records made during the current simulation
getCarbRecords	returns the collection of carbohydrate records made during the current simulation
getActRecords	returns the collection of activity records made during the current simulation
setInsRecord	adds a new insulin record, with or without provided timestamp
setCarbRecord	adds a new carbohydrate record, with or without provided timestamp
setActRecord	adds a new carbohydrate record, with or without provided timestamp
getBGManualChanges	returns collection of records for manually changed blood glucose levels
getBGChangeOverTime	returns collection of blood glucose levels over time
setBGChangeOverTime	makes a new blood glucose level into the collection, with or without provided timestamp
getInsBGChangeOverTime	returns collection of records with blood glucose level influences by Insulin module
getCarbBGChangeOverTime	returns collection of records with blood glucose level influences by Carbohydrate module
getActBGChangeOverTime	returns collection of records with blood glucose level influences by Activity module
getCurrentBGLevel	returns current blood glucose level
setCurrentBGLevel	used for manual change of current blood glucose level, with or without provided timestamp
setCurrentBGLevelValue	changes current blood glucose level without making a manual blood glucose change record
getInsActiveUnits	returns the sum of insulin units in currently active records
getCarbActiveUnits	returns the sum of carbohydrate grams in currently active records
getInsCurrentBGChange	returns the difference between the last and the current blood glucose level influence by Insulin module
getCarbCurrentBGChange	returns the difference between the last and the current blood glucose level influence by Carbohydrate module
getActCurrentBGChange	returns the difference between the last and the current blood glucose level influence by Activity module
getInsCurrentTotalWork	returns current blood glucose level influence by Insulin module
getCarbCurrentTotalEffect	returns current blood glucose level influence by Carbohydrate module
getActCurrentTotalEffect	returns current blood glucose level influence by Activity module
getInsMaxModuleWork	returns the maximal blood glucose level influence by

	Insulin module that will be reached when the activity time of the longest record is out
getCarbMaxModuleEffect	returns the maximal blood glucose level influence by Carbohydrate module that will be reached when the activity time of the longest record is out
getActMaxModuleEffect	returns the maximal blood glucose level influence by Activity module that will be reached when the activity time of the longest record is out
getInsNumberActive	returns number of currently active insulin records
getCarbNumberActive	returns number of currently active carbohydrate records
getActNumberActive	returns number of currently active activity records
getNumberActive	returns number of currently active records in the provided collection of records
getInsActiveTime	returns the time value of the currently active insulin record that will be affecting blood glucose level longer than the other insulin records (if there are active insulin records that are currently affecting blood glucose level); the value is a “timer” and is decreasing over time
getCarbActiveTime	returns the time value of the currently active carbohydrate record that will be affecting blood glucose level longer than other carbohydrate records (if there are active carbohydrate records that are currently affecting blood glucose level); the value is a “timer” and is decreasing over time
getActActiveTime	returns the time value of the currently active activity record that will be affecting blood glucose level longer than the other activity records (if there are active activity records that are currently affecting blood glucose level); the value is a “timer” and is decreasing over time
getInsCurrentWorkPercent	returns % of work done by the insulin module
getInsCurrentTotalEffectPercent	returns % of current insulin module influence strength
getCarbCurrentWorkPercent	returns % of work done by the carbohydrate module
getInsCurrentTotalEffectPercent	returns % of current activity module influence strength
getTotalDailyInsulinDose	returns the number of insulin units that user has set in during the last 24 simulation-hours
getTotalDailyCarboDose	returns the number of carbohydrate grams that user has set in during the last 24 simulation-hours
getBGAverage	returns the average blood glucose level for the last 24 simulation-hours
getBGHighs	returns the number of blood glucose highs occurred during the last 24 simulation-hours
getBGLows	returns the number of blood glucose lows occurred during the last 24 simulation-hours
getCurrentGlucSens	returns users glucose sensitivity in g/(mmol/L)
getCurrentInsSens	returns users insulin sensitivity in (mmol/L)/IU for rapid- and short-acting insulin – Humalog, Novolog or

	Regular, assuming that user uses one kind of rapid- or short-acting insulin per time
getCurrentLongInsSens	returns users insulin sensitivity in (mmol/L)/IU for intermediate- and long-acting insulin – NPH, Levemir or Lantus, assuming that user uses one kind of intermediate- and long-acting insulin per time

Table 4: A part of the API, located in Engine class.

As we can see, there are more "gets" than "sets", because some data should not be changed or updated outside the engine.

The methods presented in Table 4 is a part of the API. Another part of the API is a set of public methods required to provide sufficient access for the game/simulator/demonstrator application to read, write and update the engine Settings; and is implemented by Settings class. See section 6.5.3 for more information.

6.5.6. Insulin Module

This section presents the last implementation of Insulin module (the older implementation can be found in Appendix D: Previous Implementation Of Insulin Module). The algorithm was implemented in accordance with what we have discussed in sections 2.1.1 and 2.1.2. If there are active insulin simulation records, the module iterates through them, and for each currently active insulin record, the module runs record procedure. Since the Insulin module calculations are the heaviest ones in the Engine, the procedure runs in separate threads in parallel for each record. The procedure consists of the following steps:

First, define the duration of action of an insulin simulation record. For different types of insulin it is different.

For Humalog, various sources (Figure 5, Figure 8) (The Diabetes Mall, 2015; Straight Healthcare, 2014; WebMD, 2015; UP Health System Marquette, 2013) say the 5-6 hours as the duration of action, or a little longer with very small action remaining during the hours 7-8 (Figure 7, Figure 9) (Drugs.com, 2002; Diabetes-Support.org.uk, 2015). The duration of action is defined in the engine as 6 hours. For Novolog, the duration of action is defined as 6 hours, same as Humalog (see Figure 5). For Regular, the main duration is set to 8 hours, however, the action is not finished in 8 hours, so additional 1.5 hours of almost-zero action were added. See Figure 8. For NPH, the duration is set to 18 hours, including almost-zero action during the last 2 hours. See Figure 11.

For Lantus, different sources (Sanofi US, 2015a; McAuley, 2015; The Diabetes Mall, 2015) (Straight Healthcare, 2014; WebMD, 2015; UP Health System Marquette, 2013; Diabetes Education Online and UCSF, 2015b) provide the duration of action time as 18-26 or 22-24 or 24+ hours. So the duration of linear action is defined as 24 hours, plus 2 extra hours for the disappearance. See Figure 11 and Figure 13.

For Levemir, the duration, the duration varies depending on dose. The algorithm for calculation of the duration is based on Figure 12 and is implemented as the set of functions shown in Figure 79.

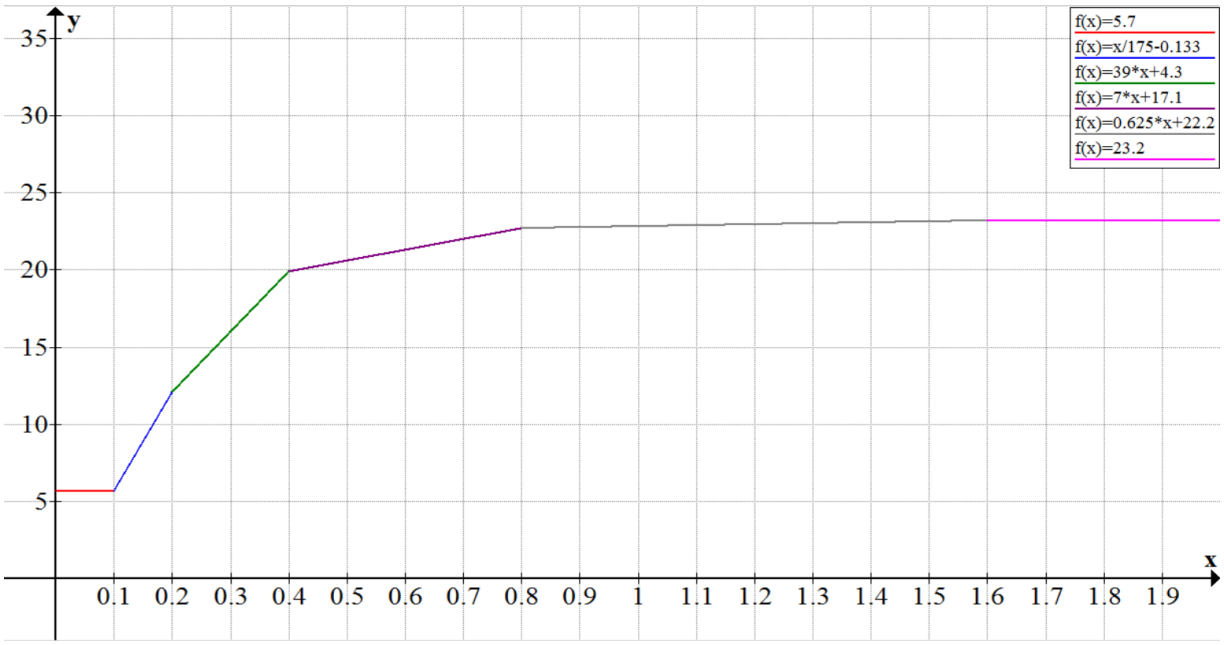


Figure 79: The function for finding the duration of Levemir action, depending on dose. Based on the data from Figure 12.

The next step is to define the peaking value (1) in mg/min/kg for the defined number of IU/kg (2), based on the information presented in Figure 5 – Figure 13. For Humalog, Novolog, Regular, NPH, Levemir, Lantus, the values 6.3 mg/min/kg for 0.2 IU/kg, 6.5 mg/min/kg for 0.2 IU/kg, 5.(6) mg/min/kg for 0.2 IU/kg, 3.4 mg/min/kg for 0.3 IU/kg, 3.0 mg/min/kg for 0.8 IU/kg, 0.82 mg/min/kg for 0.3 IU/kg, are set respectively.

Then, this value is used to calculate a coefficient:

```
Coeff (mg/kg/min) =
number of IU in the record * (1) / ((2) * weight)
```

The next step is to find the current influence and the full influence of the record, by calculating the area under the action curve for particular type of insulin. This is implemented as a loop that iterates through the whole duration line (from 0, when the record was made, till the defined duration) and makes effect sum in mg/kg, or just mg if we multiply it by weight after the area. The influence is measured in mg/kg/min, so each iteration step is defined as 1 minute. Each iteration looks as follows:

```
Res +=
Coeff (mg/kg/min) * Influence for the moment in time (%/100)
```

The functions used for the calculation of the influence over time, build the action curves, which are estimated versions of Figure 5 for Humalog and Novolog, Figure 8 for Regular, Figure 11 for NPH and Lantus, Figure 12 for Levemir. These estimated curves are shown below in Figure 80 – Figure 86.

The result of this loop is the total influence done during the given time in mg/kg. Current influence is calculated in the same way but with the only difference – the loop runs not until the end of the duration, but until “now”.

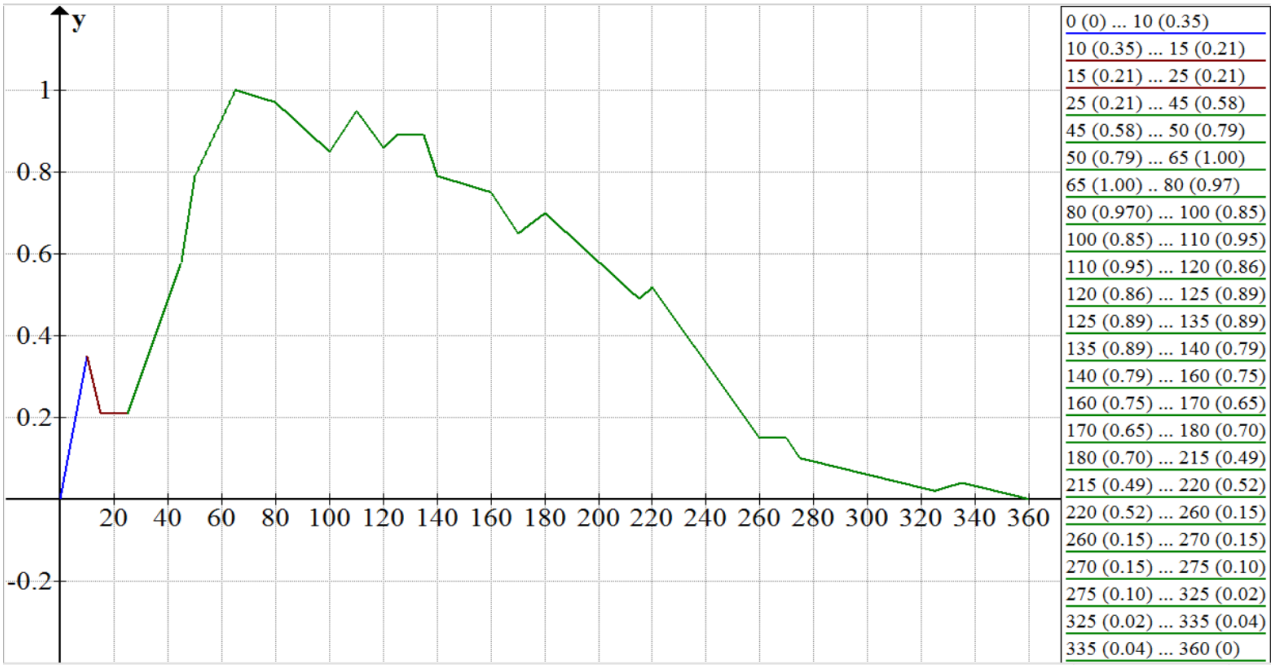


Figure 80: Current implementation of action curve for Humalog, estimation of Figure 8. X-axis represents minutes, Y-axis represents action from 0 to 1.

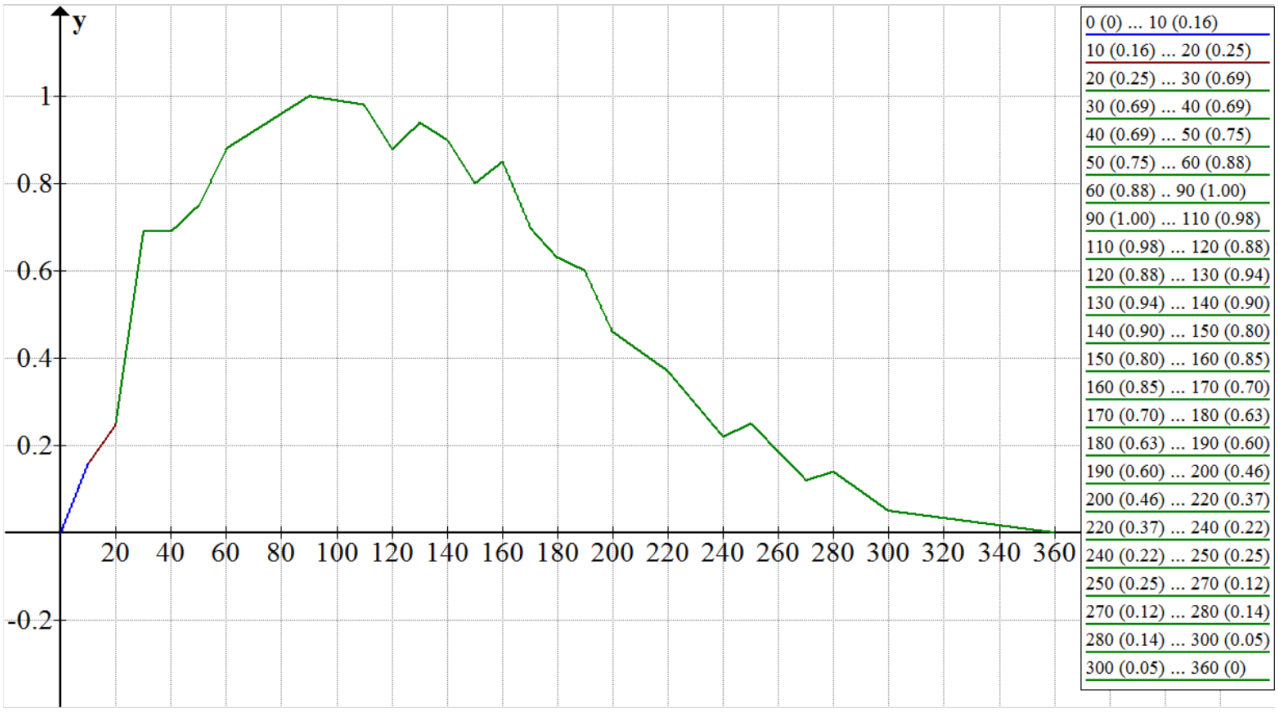


Figure 81: Previous implementation of action curve for Humalog, estimation of Figure 5. X-axis represents minutes, Y-axis represents action from 0 to 1.

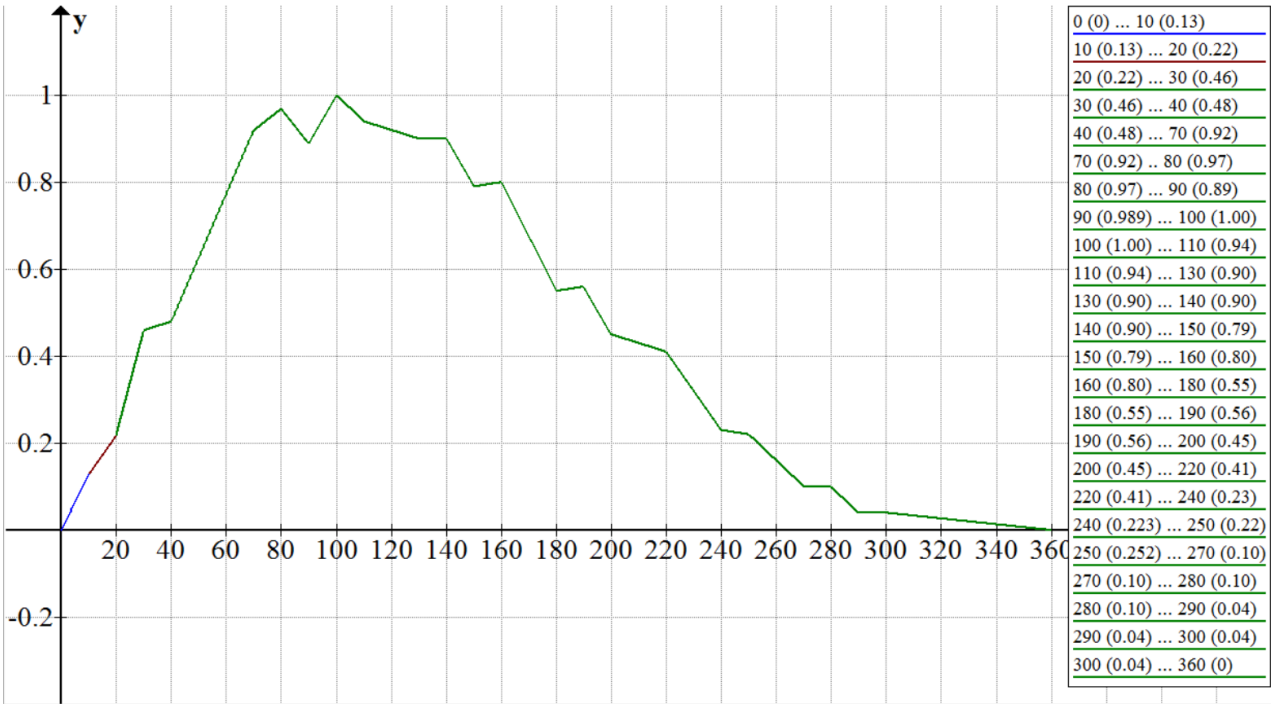


Figure 82: Action curve for Novolog, estimation of Figure 5. X-axis represents minutes, Y-axis represents action from 0 to 1.

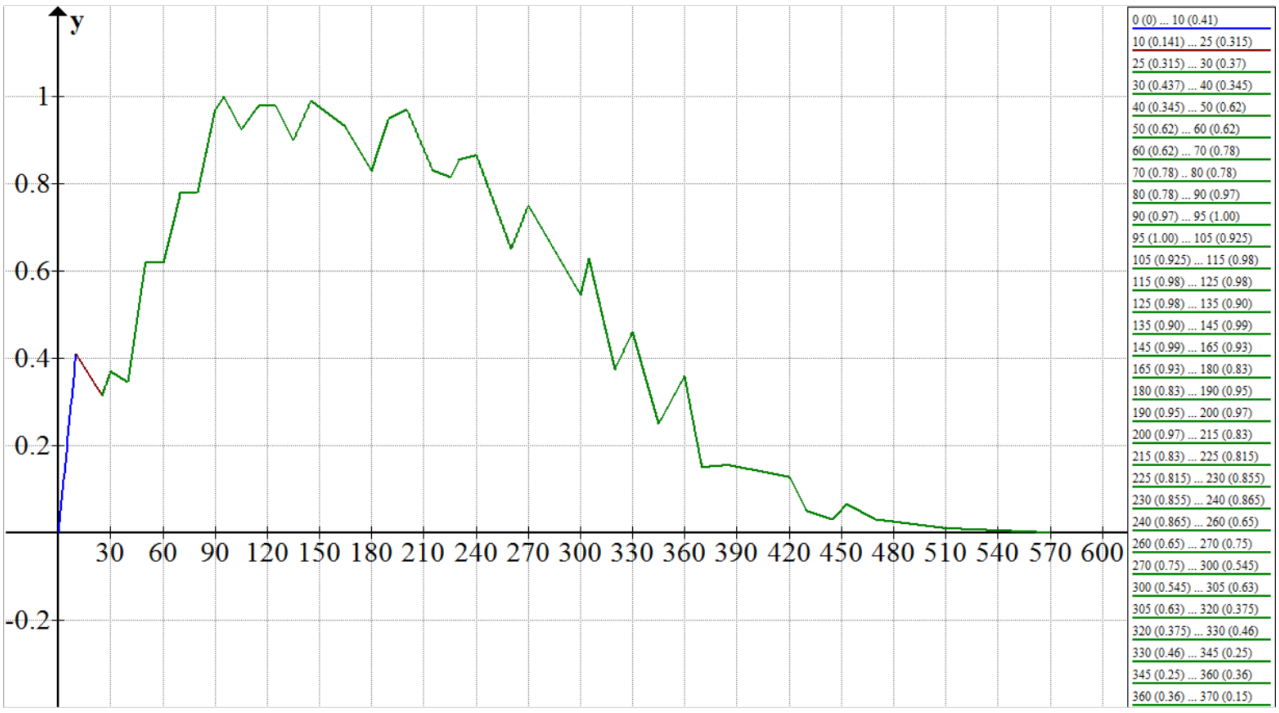


Figure 83: Action curve for Regular, estimation of Figure 8. X-axis represents minutes, Y-axis represents action from 0 to 1.

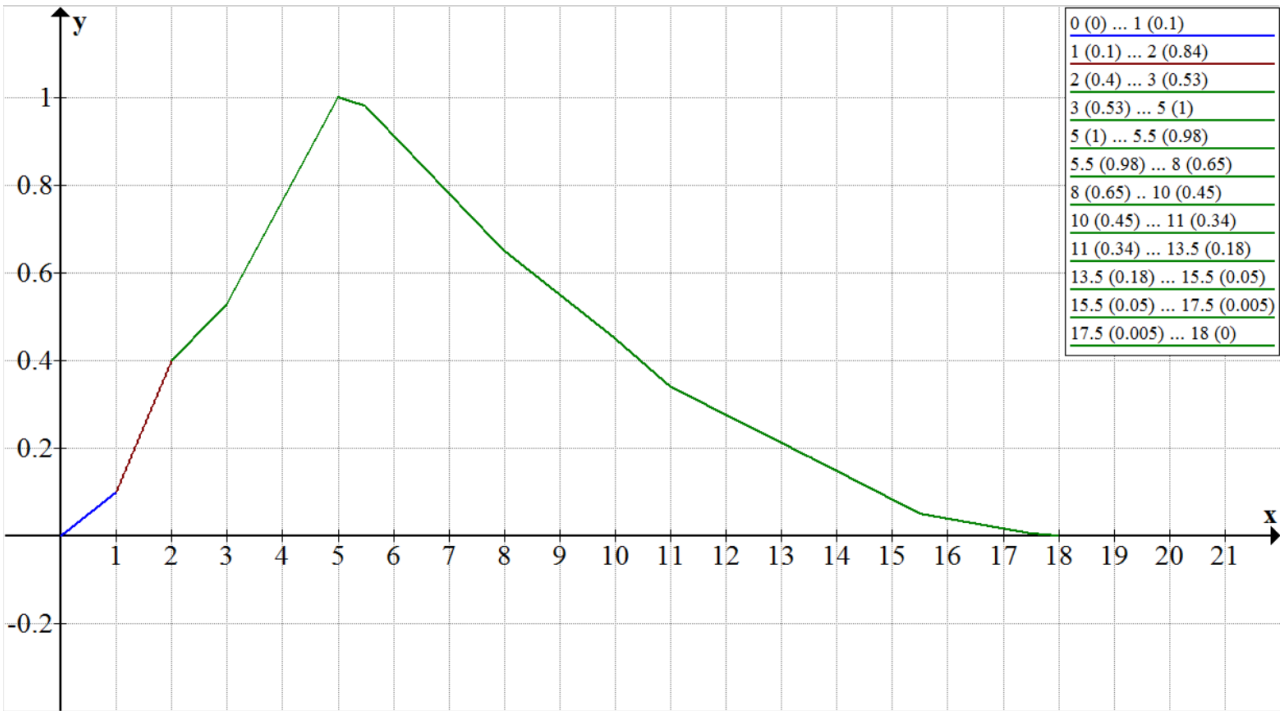


Figure 84: Action curve for NPH, estimation of Figure 11. X-axis represents minutes, Y-axis represents action from 0 to 1.

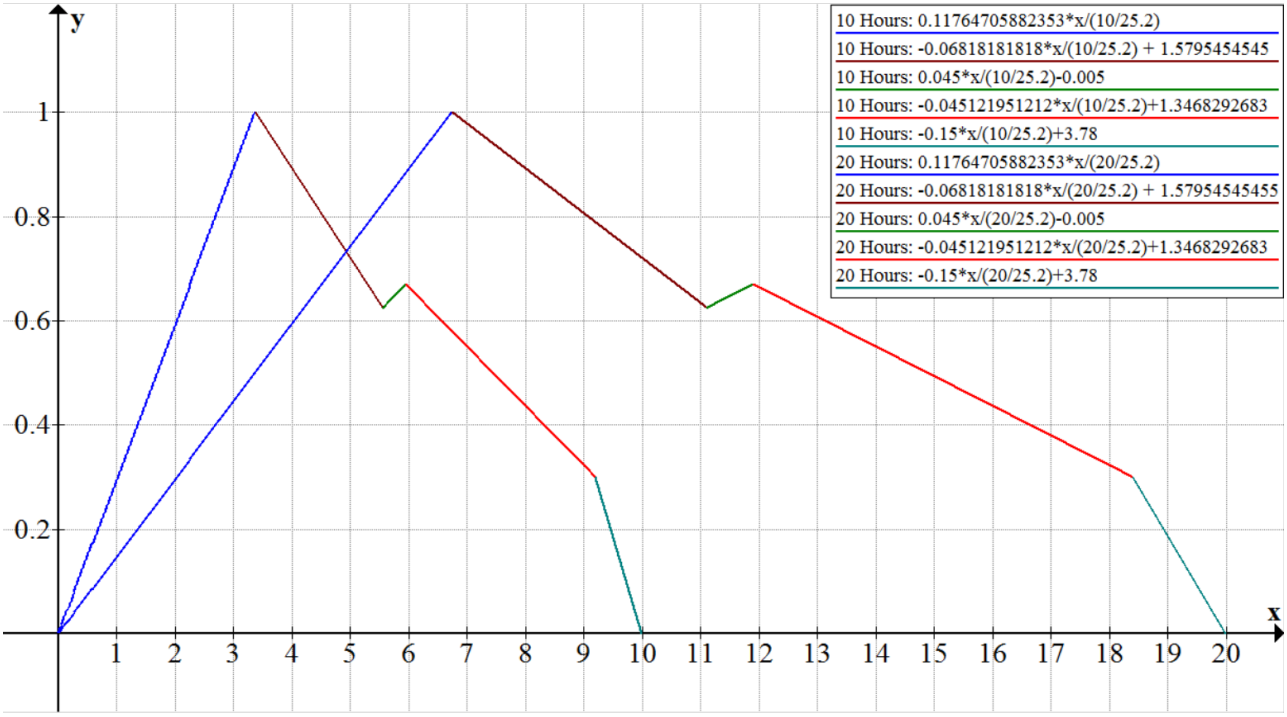


Figure 85: Action curve for Levemir, estimation of Figure 12. X-axis represents minutes, Y-axis represents action from 0 to 1. Examples for t=10h and t=20h.

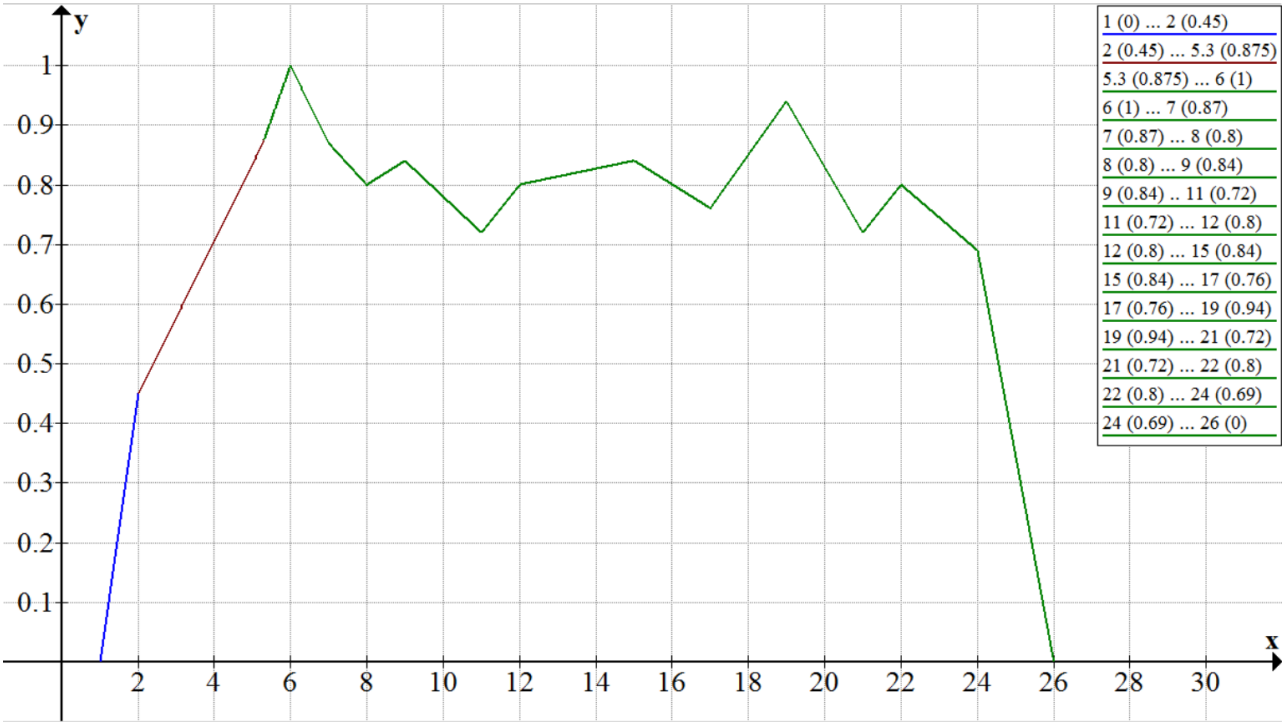


Figure 86: Action curve for Lantus, estimation of Figure 11. X-axis represents minutes, Y-axis represents action from 0 to 1.

The last step is to convert the results to mmol/L. We know that 1 mole of glucose is 180.15588 g of glucose (ConvertUnits.com, 2015), or 1 mmol of glucose is 180.15588 mg of glucose. So first the conversion was implemented as follows:

```
# (mmol/L) =
# (mg/kg) * weight (kg) / (180.15588 * blood volume (L))
```

However, it was not enough. The results were very much overestimated and unrealistic. What we need to know in order to make the formula work, is that insulin is distributed in the extracellular space in the body (Pharmacorama.com, 2015), as well as glucose (Wick et al., 1950; Ishihara and Giesecke, 2007, p. 9; Hirota et al., 1999; Boundless, 2014). Extracellular space consists of interstitial space and blood plasma (Stedman, 2012, "extracellular fluid"; Farlex, 2015a). The interstitial volume equals to 16% of body weight (Stedman, 2012, "interstitial fluid"; Farlex, 2015b). The blood plasma volume is about 55% of blood volume (Venes and Taber, 2013, p. 303; American Society Of Hematology, 2015; The American National Red Cross, 2015). This information is used to calculate the value that “weight” in the formula above must be multiplied with. This value is calculated as follows:

```
Multiplier =
0.16 + (0.55 * blood volume / weight)
```

The final implementation of the conversion in implemented as follows:

```
# (mmol/L) =
# (mg/kg) * weight (kg) * Multiplier / (180.15588 * blood
volume (L))
```

Now, we have the current influence and the total influence that will be made by insulin record. Before the module finishes its execution, it updates the additional module values, such as the total module current influence, the total module 100% influence, the current influence strength in percent (Figure 81 – Figure 86), the relation of the current total influence work done to the 100% work (total influence) in percent, the number of currently active records, the remaining time of action and the history blood glucose influence value for future purposes, and writes all the values to the State. If the action time of the current record is over, it gets marked as deleted.

If the "Dawn Phenomenon" option is turned on, the action done during 2:00-8:00 is reduced by 50%.

6.5.7. Carbohydrate Module

In this section we discuss how Carbohydrate module is implemented. The algorithm was implemented in accordance with what we have discussed in sections 2.1.1 and 2.1.3. If there are active carbohydrate simulation records, the module iterates through them, and for each of them does the following steps.

First, the module calculates the duration of action. The duration depends on GI of the record and is calculated by the function represented in Figure 87.

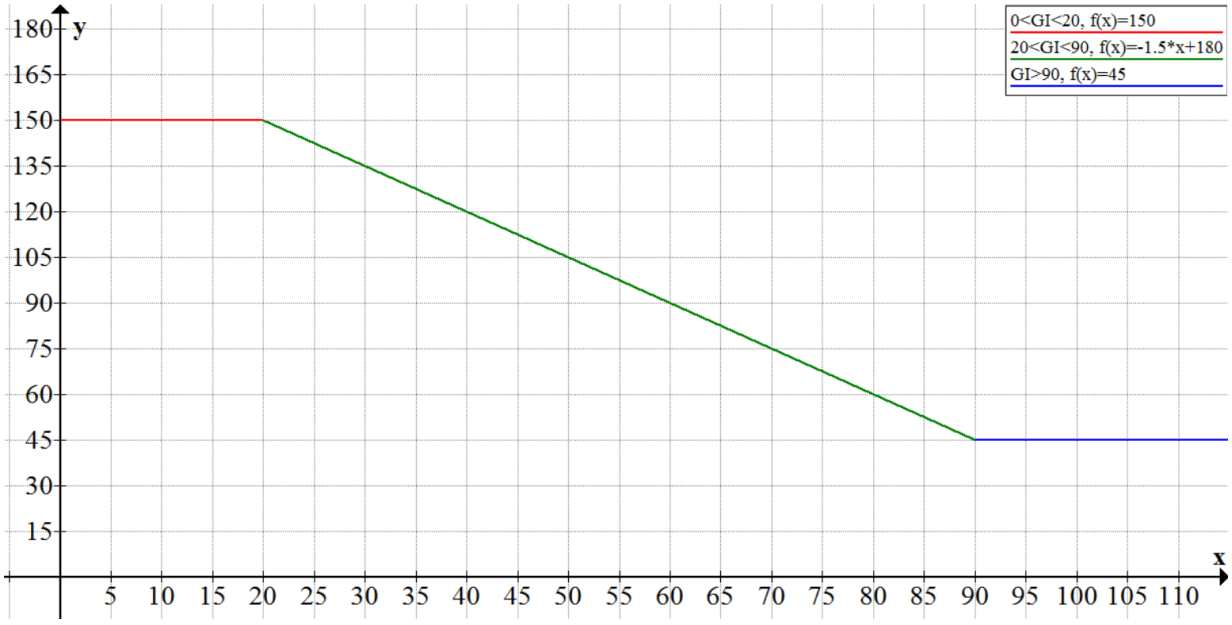


Figure 87: Calculation of the duration of action of a carbohydrate simulation record. X-axis represents GI, Y-axis represents minutes.

Next step is to calculate the influence work done in %/100 for the current moment in time. The result depends on the duration, the example curves for 45, 60, 90 and 120 minutes are represented in Figure 88.

After that, the module finds the glucose sensitivity. The calculations are based on the data in Table 1 and are implemented as the set of function shown in Figure 89.

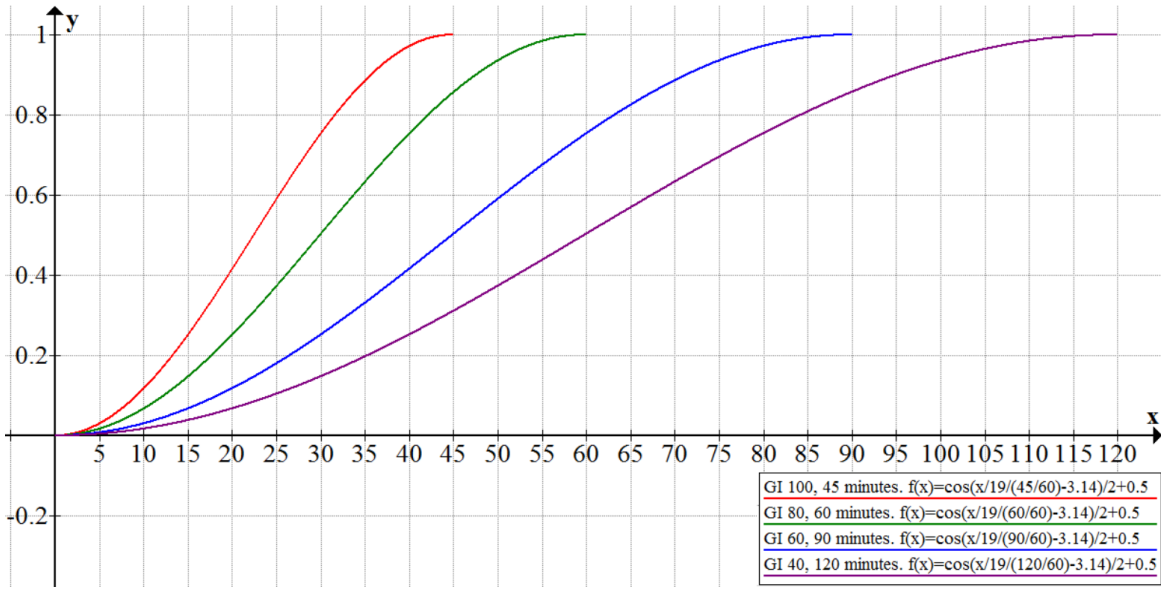


Figure 88: Blood glucose influence work done over time; duration = 45, 60, 90, 120 minutes. X-axis represents minutes, Y-axis represents progress from 0 to 1.

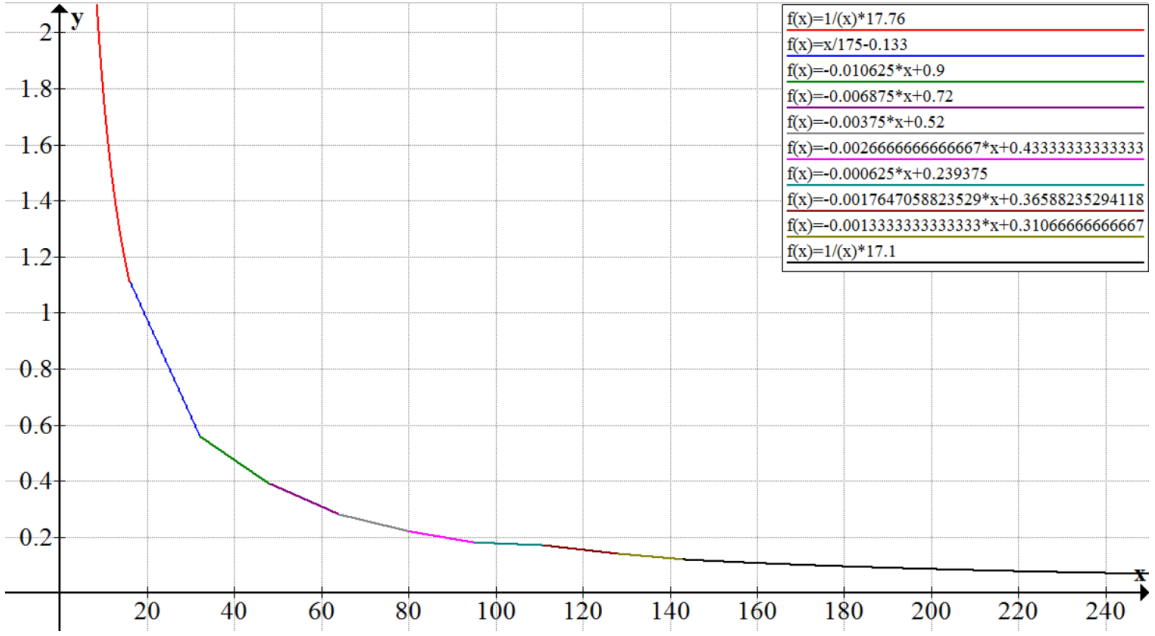


Figure 89: Glucose sensitivity curve based on the data in Table 1. X-axis represents weight in kg, Y-axis represents mmol/L.

Testing (presented in chapter 7) has shown that the data presented in Table 1 underestimates glucose sensitivity, so the number of grams of glucose that would increase blood glucose by 1 mmol/L is increased by 20%.

The last step is to calculate the rise in blood glucose in the current moment in time. The calculation is as follows:

```
Rise in blood glucose (mmol/L) =
Number of carbohydrates in the record (g) *
GI / 100 *
Work done for the moment in time (%/100) /
Glucose sensitivity (g per 1 mmol/L)
```

Before the module finishes its execution, it calculates additional module values, such as such as the total module current influence (work), the total module 100% influence, the relation of the current work done to the 100% work in percent, the number of currently active records, the remaining time of action and the history blood glucose influence value for future purposes, and writes all the values to the State. If the action time of the current record is over, it gets marked as deleted.

6.5.8. Activity Module

This section presents the implementation of the Activity module.

6.5.8.1. The general explanation of the Activity Module algorithm

As discussed in 2.1.3, aerobic and anaerobic types of exercise affect blood glucose in a different way, so 2 different algorithms were implemented for the activity module.

In general, for each active activity record, the module checks whether the current activity record is aerobic or anaerobic, and makes necessary calculations (see 6.4.7.2 and 6.4.7.3), and defines the influence duration for the current record and calculates the percentage of the insulin sensitivity influence (and direct blood glucose influence in case of anaerobic activity record). After the iteration through all activity records is done, the final results of the blood glucose influence and insulin sensitivity influence (the sum of the results for all active records) are written to the State and the module finishes its execution. If the active time of the current record is over and the record doesn't make any influence, it gets removed from the record list in the State.

The module doesn't affect blood glucose levels directly, it affects them by increasing the insulin sensitivity, which means that insulin decreases blood glucose levels more than if there was no activity. In other words, activity module will not affect blood glucose if there is no currently active insulin records.

Different kinds of activity have different influence duration and behavior of action. The calculation of the influence percentage for the current moment in time is explained below.

6.5.8.2. The algorithm for aerobic activity records

According to what we have discussed in 2.1.3, the increase of insulin sensitivity in the engine was set first to average percentage – 30%, but testing has shown that it is too much, so the percentage was reduced to 15%; and the duration of increased insulin sensitivity can be between 24 and 72 hours coming after a single exercise session, depending on the duration and the intensity of the exercise (see section 2.1.3).

The engine defines the action of the activity effect as follows: if the current activity record is marked with "short duration (15-30 min.)" or "moderate duration (31-60 min.)" or "long duration (1-2 hours)", then the duration of increased insulin sensitivity is 24, 48 and 72 hours respectively. The implemented activity action curves are represented in Figure 90:

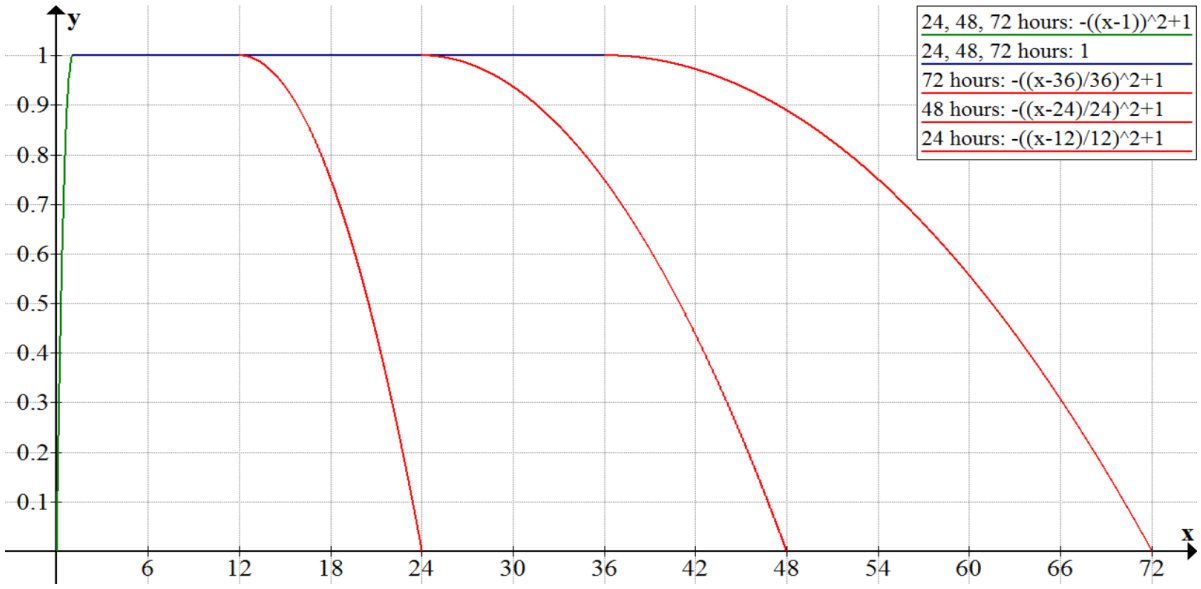


Figure 90: Curve that shows how the influence of aerobic activity on insulin sensitivity multiplier (which is 0.3, or 30%) over time (24h, 48h, 72h) is implemented.

In the curves presented in Figure 90 we see how the action starts taking effect, reaches 100% (the point where the insulin sensitivity increase reaches 30%), remains on this level, and then goes down to 0 when the action time is over.

Testing has shown that 30%-increase is actually too big, so the increase in insulin sensitivity was corrected to 15% during testing (see section 7.2).

Finally, after the current percentage of the insulin sensitivity influence is calculated, the module changes the current insulin sensitivity multiplier in the State, calculates and writes the statistical module information to the State.

Only the last aerobic activity record goes through the calculations, because the aerobic activity's influence lasts long and should not be summed with the influence of another aerobic activity record. The maximal insulin sensitivity influence should not be doubled if there are 2 currently active aerobic records in the list.

6.5.8.3. The algorithm for anaerobic activity records

According to section 2.1.3, the estimated algorithm for the engine can look as follows. Doing anaerobic activity rises blood glucose from the moment of exercise and until 1 to 2-3 hours after; in most subjects the rise can be over 2 and up to 4 mmol/l, and in some subjects it can be over 4 mmol/l, so the decision was to take 3 mmol/l as a peak value; the blood glucose level starts going down 2-3 hours after due to the increased insulin sensitivity (similar to aerobic activity), insulin sensitivity is increased up to 15% and this effect finishes in 14-20 hours.

This is what the algorithm for anaerobic activity is based on. The first stage lasts 3 hours and increases the blood glucose level to maximum 3 mmol/l. The blood glucose influence curve is represented in Figure 91:

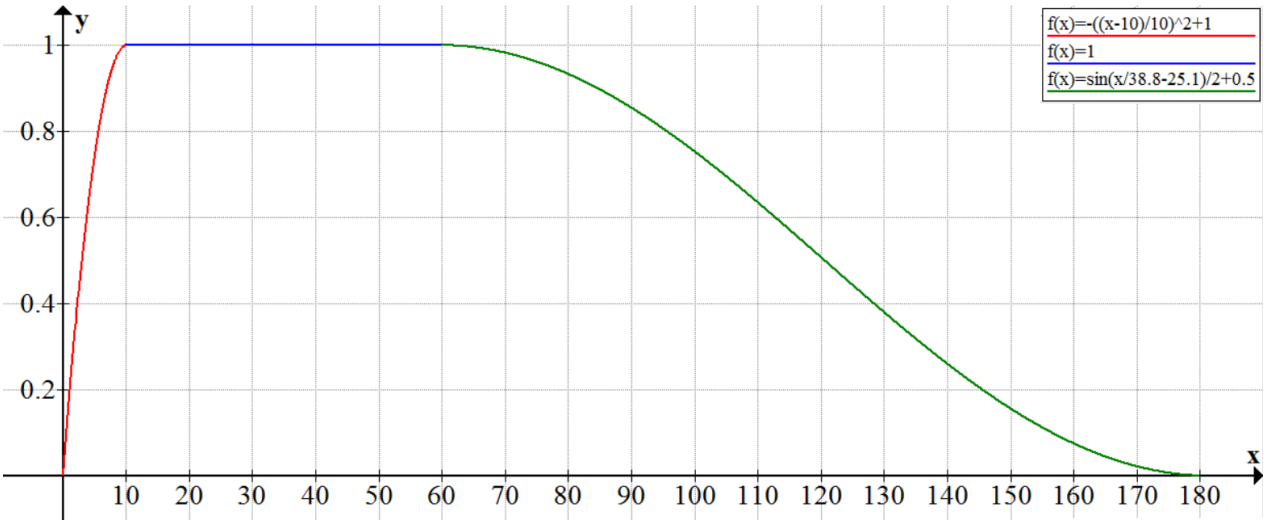


Figure 91: Curve that shows how the influence of anaerobic activity on blood glucose level over time is implemented.

The curve in Figure 91 shows how the blood glucose increase starts taking effect, reaches 100%, remains 100%, goes down and reaches 0 when the action time is 3 hours.

The second stage influences insulin sensitivity, has 20 hour active influence time and starts at the same time with the first stage. The insulin sensitivity influence curve is represented in Figure 92:

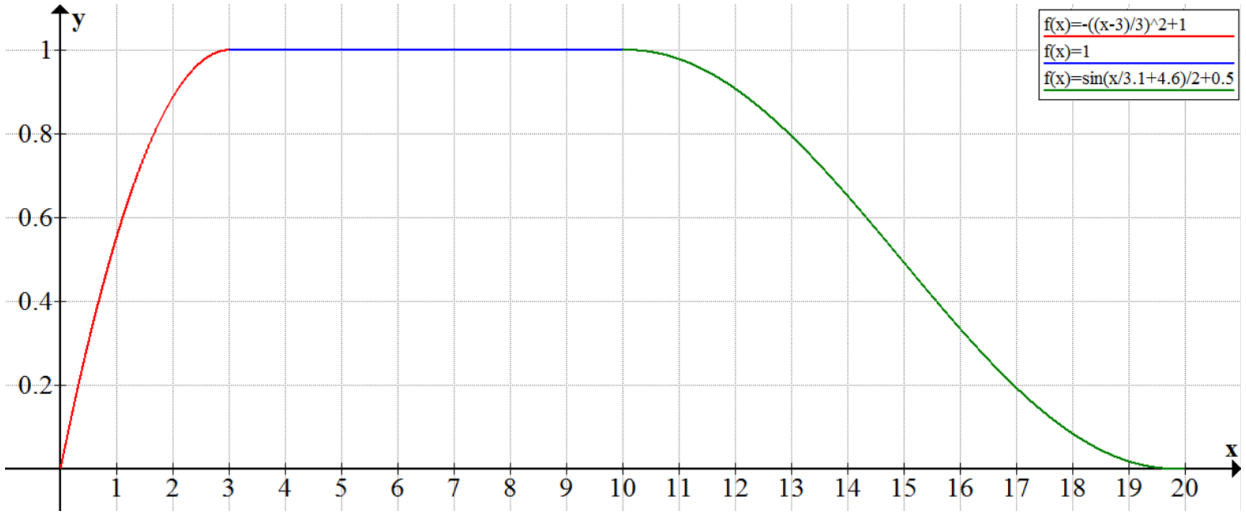


Figure 92: Curve that shows how the influence of anaerobic activity on insulin sensitivity multiplier (which is 0.15, or 15%) over time is implemented.

The curve in Figure 92 shows how the 15%-increase in insulin sensitivity starts taking effect, reaches 100%, remains on this level, and then goes down and reaches 0 when the time is 20 hours since the exercise was done.

6.5.9. Blood Glucose Module

Blood glucose module is responsible for updating current blood glucose value over time.

First, it finds the sum of the current blood glucose level influences by Insulin, Carbohydrate and Activity modules, which are saved in the state, and updates the current blood glucose value by adding this sum to the blood glucose value.

Then, if the option for the consumption of glucose by brain is enabled, the consumption value is calculated and subtracted from the blood glucose value. The calculations are based on the unit conversion algorithm from Insulin module (see 6.4.5.1) and on what we have previously discussed in 2.1.3, and are implemented in the formula:

```
Brain consumption (mg/kg per time passed) =
[-4.0(g) / weight (kg) * 0.55 (%/100) / (180.15588 * blood volume
(L) / 1000)] * [time passed / 1 minute]
```

After that, if the option for dawn phenomenon is turned on and if the current simulation time is between 3:00 and 7:00, the raise of glucose for the passed time is calculated and added to the blood glucose value. The calculations of this value are based on what we have previously discussed in 2.1.3, in particular on (Freckmann et al., 2008), and are implemented as follows. If the simulation time is between 3:00 and 5:30, the value is calculated by the formula:

```
Value (mmol/L per time passed) = 0.5 * (time passed / 1 minute)
```

If the simulation time is between 5:30 and 7:00, the value is calculated by the formula:

```
Value (mmol/L per time passed) = 1.4 * (time passed / 1 minute)
```

Finally, the module makes a new blood glucose record with timestamp and writes the updated blood glucose value to the state.

The module doesn't allow blood glucose level to go below 2 mmol/L, if the corresponding option is enabled (see sections 6.5.3, 6.6.3 and 6.6.4). This engine option is enabled by default.

6.5.10. Database

Initially, the Database module supposed to hold all simulation data that is no longer needed for the calculation of the current blood glucose level, whereas State supposed to hold data that is currently in use. In the current version of the prototype, the Database module is not implemented, and State takes a part of its designed functionality – it keeps the simulation data for the last 30 simulation-days, as mentioned in section 6.5.4.

6.6. Diabetes Automata UI (Demonstrator)

Diabetes Automata UI, or the demonstration simulator, is developed Java programming language for Android, using Eclipse with installed ADT plugin. The mobile phone used during the development is Sony Xperia Z2. The operating system on the mobile phone was Android 4.4.2 most of the time, and Android 5.0 during the last 2 months of the development.

In this section we discuss the implementation of the Simulator part of last version of the prototype (v0.010). It uses the Diabetes Automata Engine described in sections 5.1 and 6.5 in order to run, control and display the ongoing simulation, and consists of 5 Android Activities (see section 6.2) presented in this section: Main Activity, New Simulation Activity, Settings and Default Settings Activities, and About Activity. Further, they are called “screens”.

The class diagram for the Android UI is presented in Figure 93:

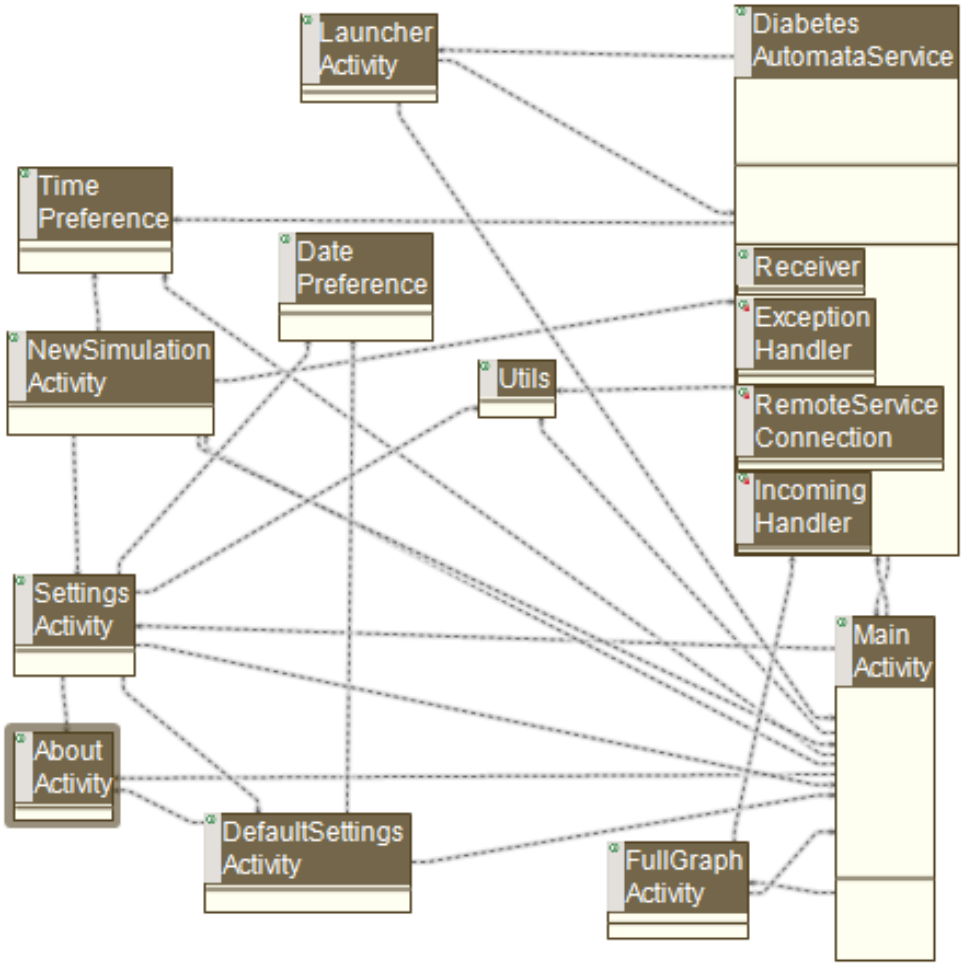


Figure 93: Class diagram of the UI.

All elements of Figure 93 are presented below, is sections 6.6.1 – 6.6.7.

6.6.1. Three Simulation Modes

The simulator provides three simulation modes. First mode is regular simulation, where user chooses time speed, calibrates blood glucose if needed, makes new insulin records with choice of insulin types and the number if IU, carbohydrate records with choice of types of GI and the number of grams, and activity records with choice of activity types (and activity duration for aerobic activity).

Second mode is history simulation of records imported Diabetesdagboka records. This is an experimental mode that makes integration with Diabetesdagboka – the simulator imports all blood glucose, insulin, carbohydrate and activity records for an amount of time (for example, 30 days), and runs simulation while automatically setting the records in the right time. Each Diabetesdagboka insulin record is set as a Humalog simulation record, with the number of IU taken from the Diabetesdagboka record. Each Diabetesdagboka carbohydrate record is set as a GI Medium simulation record, with the number if grams taken from the Diabetesdagboka record. Each activity record is set as an Aerobic activity record, with the number of minutes taken from the Diabetesdagboka record. In this mode, user can't make new insulin/carbohydrate/activity records. Diabetesdagboka v1.5.3 (ee7a27fb) with inter-process communication service (IPC) is required for this mode.

Third mode is also experimental and based on the integration with Diabetesdagboka – it provides real-time simulation paired with Diabetesdagboka. When user registers a new blood glucose, insulin, carbohydrate or activity record in Diabetesdagboka, it is automatically imported by the simulator and set into the running simulation, in the same way as described the description of the second mode. Changing time speed is not allowed in this mode. Same as in the mode 2, Diabetesdagboka v1.5.3 (ee7a27fb) with IPC is required for this mode. The frequency of sending request to the IPC in order to check if there are new Diabetesdagboka records, is 1 time in 30 seconds.

6.6.2. Settings & Default Settings

Settings screen is the first screen that user sees after installing and launching the demonstrator. The screen is presented in Figure 94. In order to personalize the simulation and get the results for particular person, the biometric values should be set correctly, according to the information about the user.

"Types of insulin" setting was made in order to make it more comfortable for the user to make new insulin records – only the ones that are marked in Settings, will be listed in the new insulin simulation record. See section 6.6.4.

The description of the values under the heading "New Simulation Values" can be found in section 6.6.3. The only available language is English. Norwegian version of the demonstrator was planned, but was not implemented due to the time limitation and the presence of more important things to implement for the project.

In order to set all settings to default with one click, the button "Use Default Settings" must be clicked. The Default settings can be changed as well – the user must click "Change Default Values" in the options menu of the screen, in order to open Default Settings screen. See Figure 95. Default Settings contain default biometric information, types of insulin, language and starting blood glucose value. The values in the "Ins/Carb/Act/BG Min/Max Values" are aimed to change the maximal values for the new insulin and carbohydrate record dialogs, and minimal and maximal value of the blood glucose correction dialog of the Main screen (for more information about the mentioned dialogs, see section 6.6.4). Default Settings screen is presented in Figure 96.

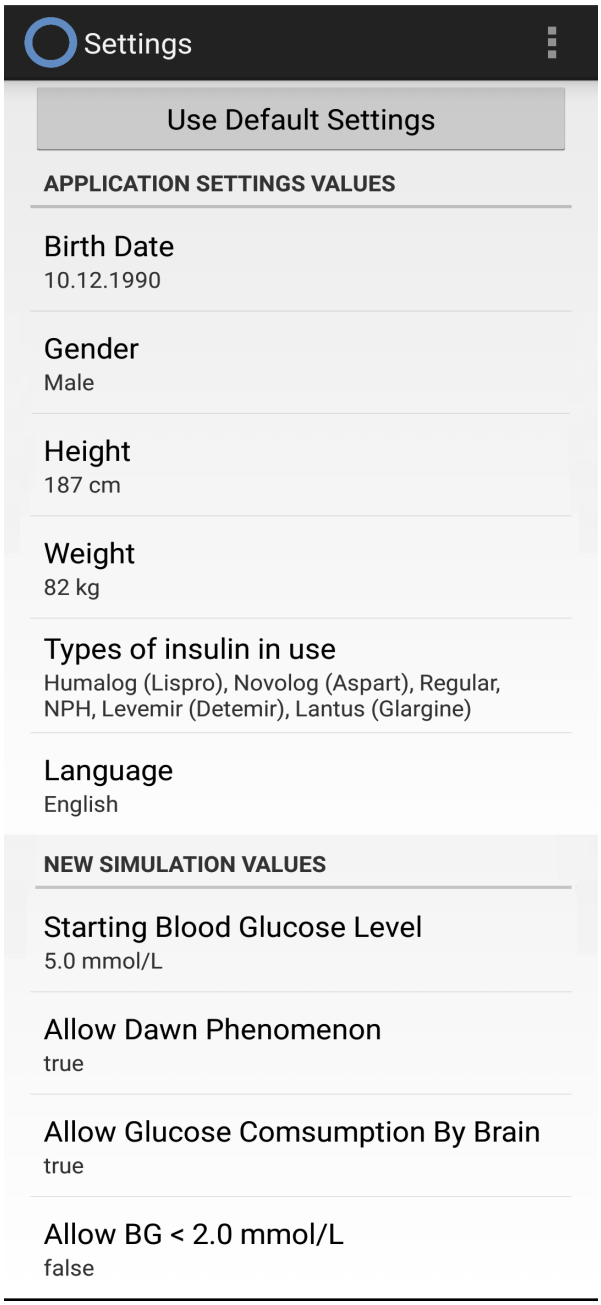


Figure 94: Settings screen of the prototype v0.010.

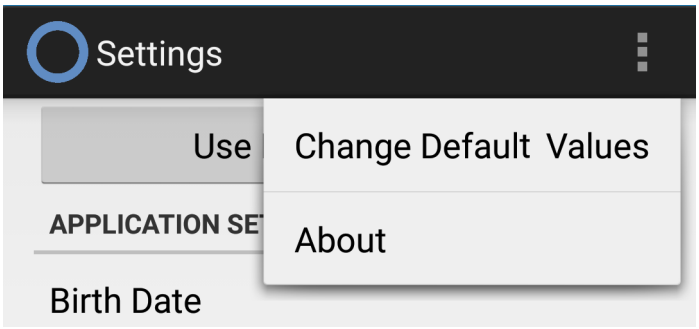


Figure 95: Options menu of the Settings screen of the prototype v0.010.

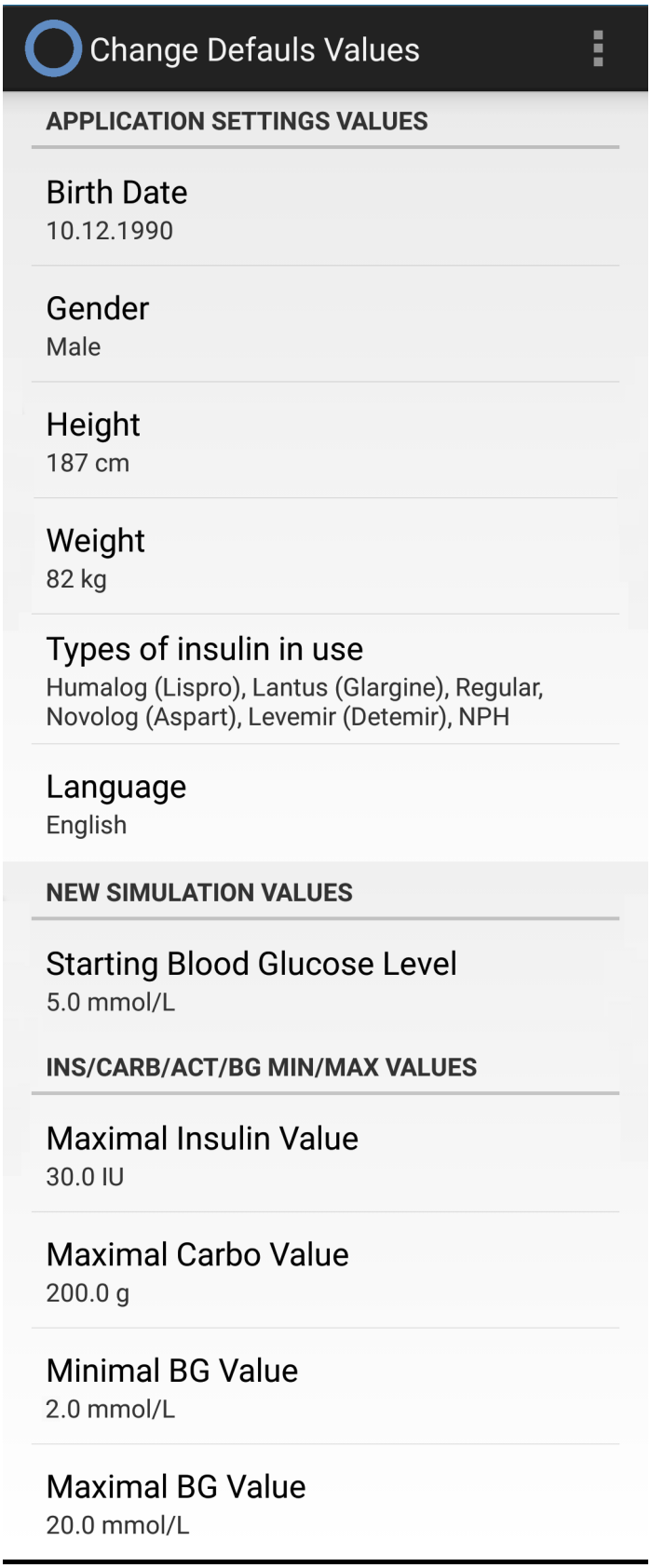


Figure 96: Default Settings screen of the prototype v0.010.

6.6.3. New Simulation

New simulation screen is presented below, in Figure 97. It provides opportunity to launch three types of simulation, which are described in section 6.6.1.

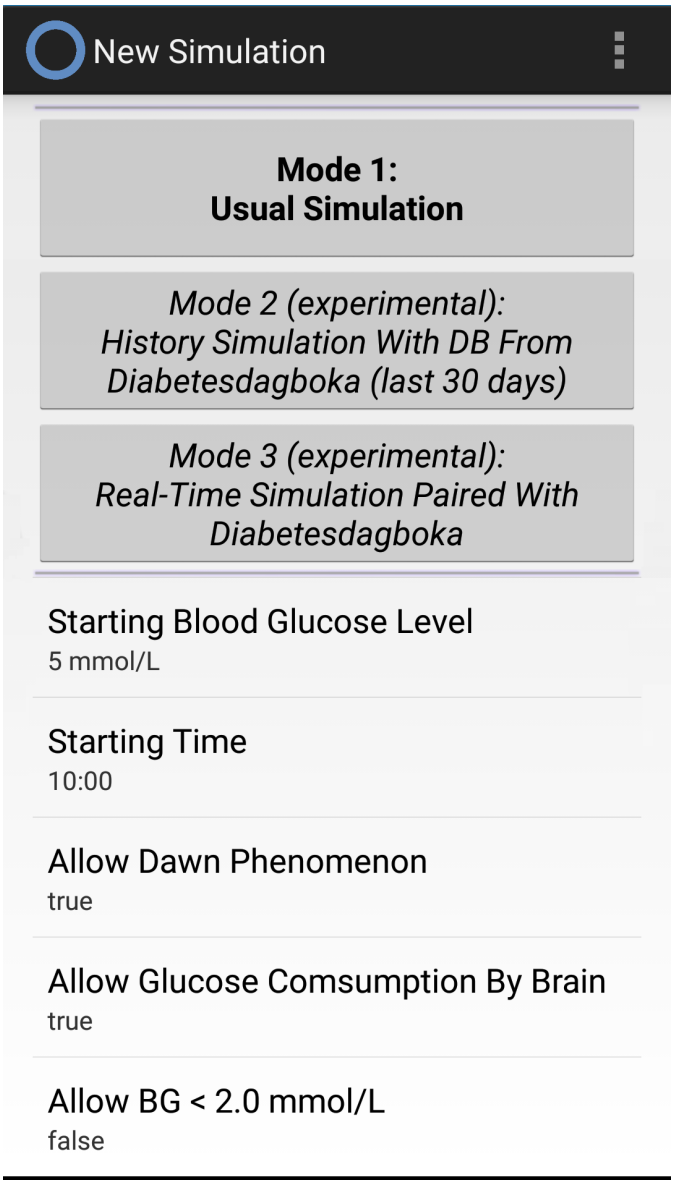


Figure 97: New Simulation screen of the prototype v0.010.

Below the simulation buttons, it has 5 simulation parameters that must be set before the simulation is started according to user's needs.

- Starting Blood Glucose Level – which blood glucose level to start with. Before the value is changed by user, it is the same as defined in Settings (see Figure 94).
- Starting Time – the starting simulation time of the day, between 00:00 and 23:59.
- Allow Dawn Phenomenon, Allow Glucose Consumption By Brain and Allow BG < 2.0 mmol/L set the corresponding simulation parameters.

6.6.4. Main Screen

Main Screen of the demonstrator consists of several clickable elements, among which are the Duration / Time Speed button, information about the last 24 simulation-hours, Insulin/Carbo/Activity buttons, Blood Glucose button, Graph, Start/Stop button, and Pause/Continue button.

The small graphical icons representing syringe, fork and knife, running man, and blood drop, were taken from Diabetesdagboka.

To provide the chart graphics, the demonstrator uses AChartEngine (4ViewSoft, 2013) – a charting software library for Android applications. The library version used in the prototype is v1.2.0. Also worth noting is that Diabetesdagboka uses AChartEngine as well.

When simulation is not running, the Main Screen of the demonstrator looks as presented in Figure 98.

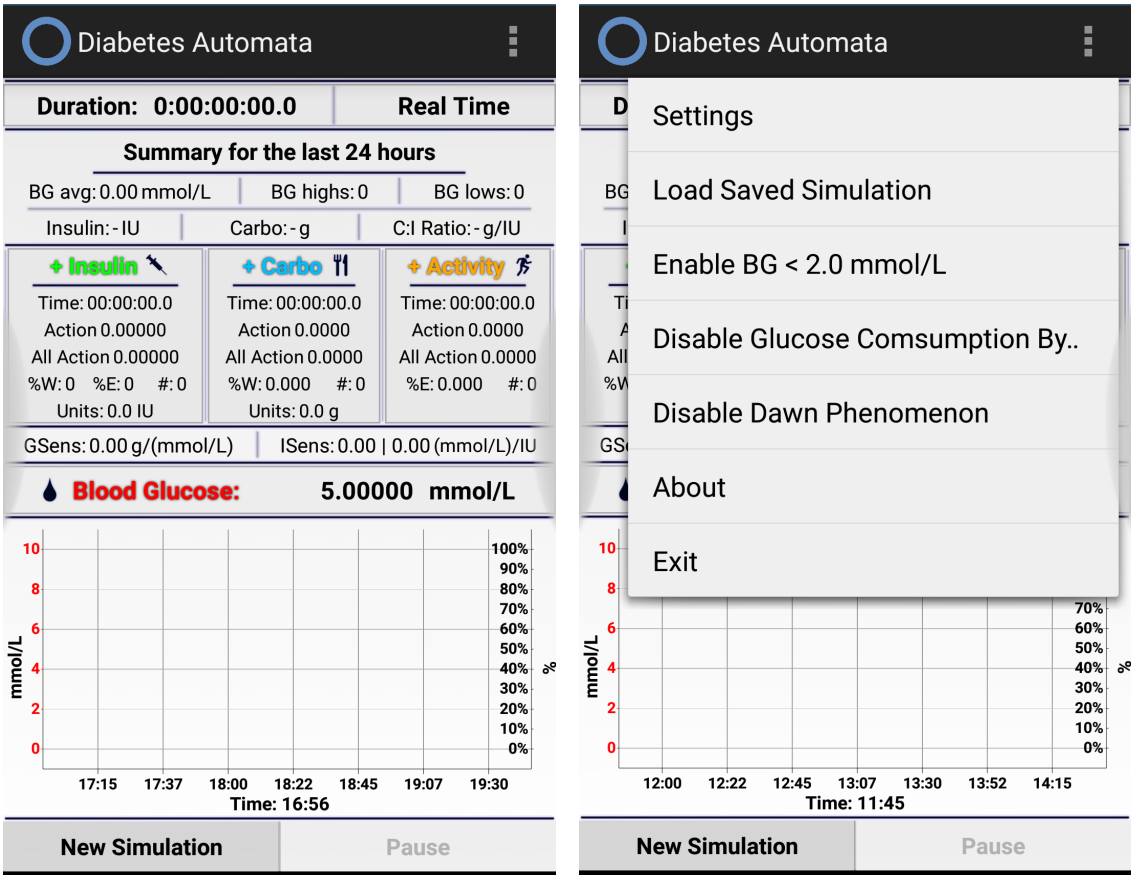


Figure 98: The Main screen of the prototype v0.010 (left), with pressed options menu (right).

New simulation screen presented in section 6.6.3 can be opened by pressing the "New Simulation" button. Options menu presented in Figure 98 (right), provides opportunities to enable/disable "BG < 2.0 mmol/L", "Glucose Consumptions By Brain" and "Dawn Phenomenon" simulation parameters, launch Settings screen presented in section 6.6.2, and open list of previously saved simulations by pressing on "Load Saved Simulation".

The dialog "Load Saved Simulation" is presented in Figure 99. It lists all saved simulation that are found of the disk.

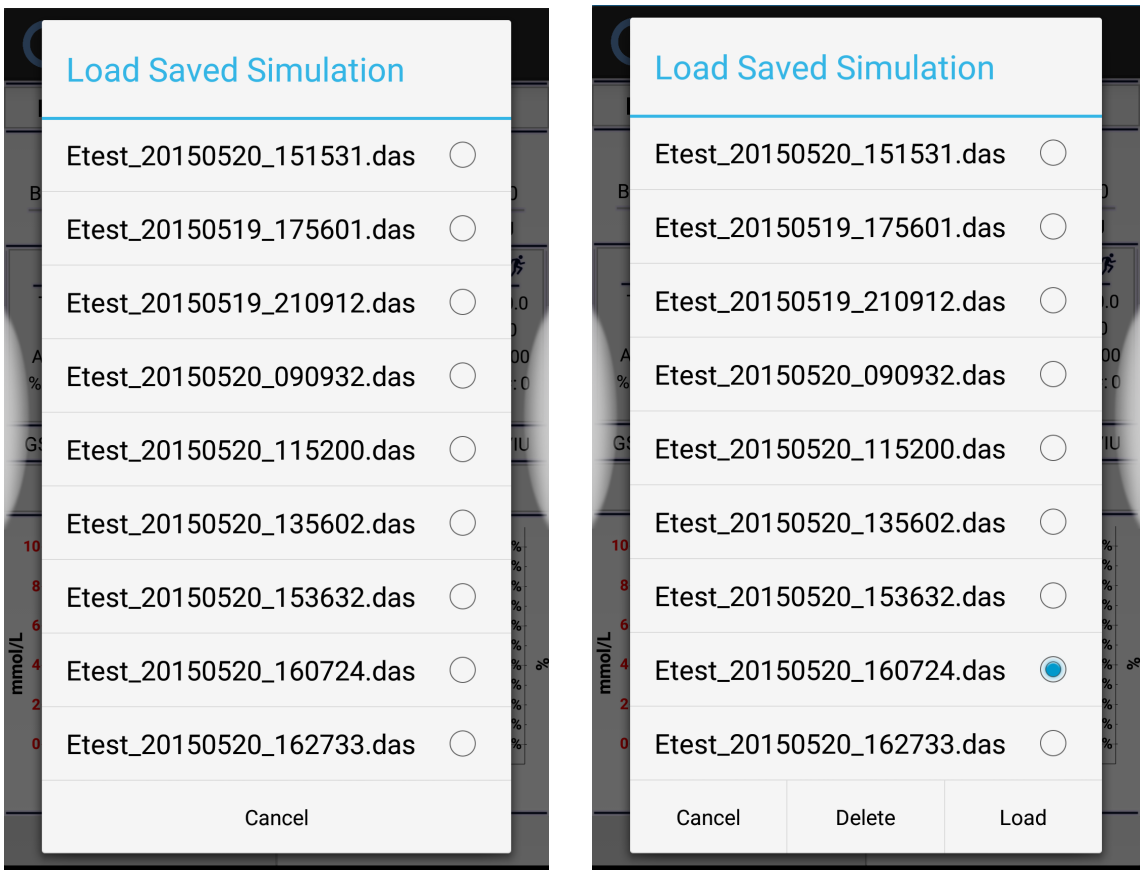


Figure 99: Load Saved Simulation dialog of the prototype v0.010 (left), simulation selected (right).

After selecting one of the simulations, the dialog is refreshed and contains 3 buttons that give opportunity to cancel the dialog, delete the selected simulation, or load in the simulation mode 1.

When “Load” button is pressed, the loading window presented in Figure 100 (left) appears on the screen and remains until the loading process is finished. After the simulation is loaded, it will be automatically set to Pause.

When “Delete” button is pressed, the confirmation dialog presented in Figure 100 (right) appears on the screen.

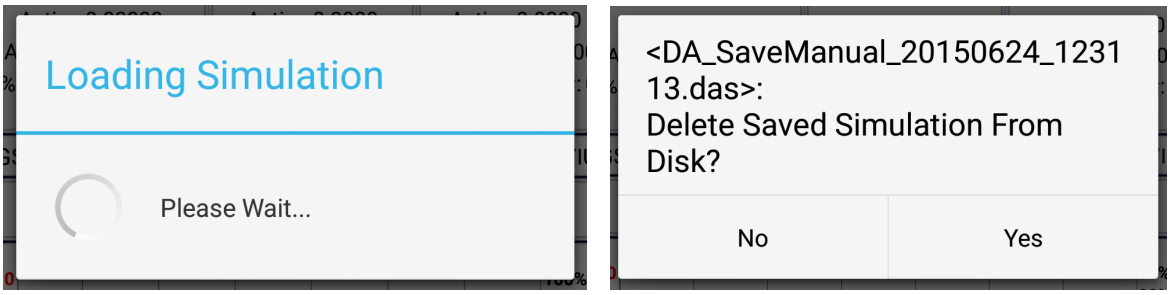


Figure 100: Loading Simulation window (left) and delete saved simulation confirmation dialog (right) of the prototype v0.010.

When a simulation is started or continued, it looks as presented in Figure 101, which also shows the simulation running with the speed of 10 minutes per second. The speed can be changed whenever user wants.

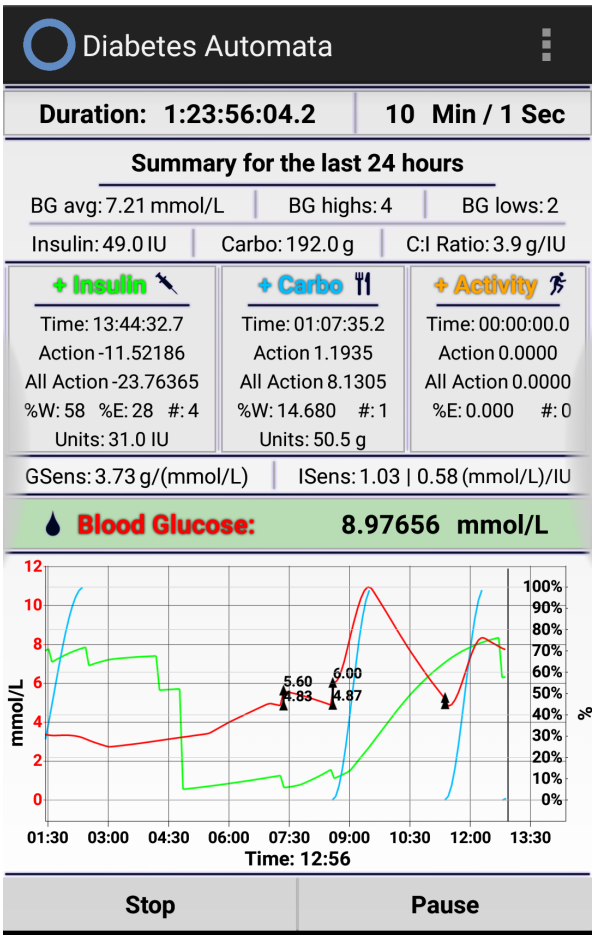


Figure 101: The Main screen of the prototype v0.010 with running simulation, time speed = 10 Min / 1 Sec.

Duration is the total duration of the simulation counted in simulation time. The value to the right of Duration is the simulation speed. GSens is glucose sensitivity – the number of carbohydrate grams that lower blood glucose level by 1 mmol/L. ISens is insulin sensitivity – the number of mmol/L dropped by 1 unit of insulin. The first ISens value corresponds to rapid- and short-acting insulin, assuming that 1 same type of rapid- or short-acting insulin is used. The second ISens value corresponds to intermediate- and long-acting insulin, assuming that 1 same type or intermediate- or long-acting insulin is used. The Time below the graph represents the current simulation-time of the day.

Each of the square buttons in the middle contains current information about the represented module. Time is action time left. Action is the number of mmol/L increased/reduced by all currently active records for the current moment in simulation time. Total Action is the number of mmol/L that is and will be increased/reduced by all currently active records, starting from the beginning time of the earliest currently active record, and finishing with the ending time of the longest currently active record. %W is current work

done by all currently active records; in other words, Action / Total Action * 100; it starts from 0% and finishes with 100%. %E is current action strength for the group of all currently active records; it starts with 0%, has a 100%-peak between the beginning and the end of the effect, and finishes with 0%. The difference between %E and %W can be visually seen in Figure 103. # is the number of currently active records. Units shows the sum of units of all currently active records.

The Main screen can be in landscape mode as well. Landscape presentation of the Main screen is shown in Figure 102:

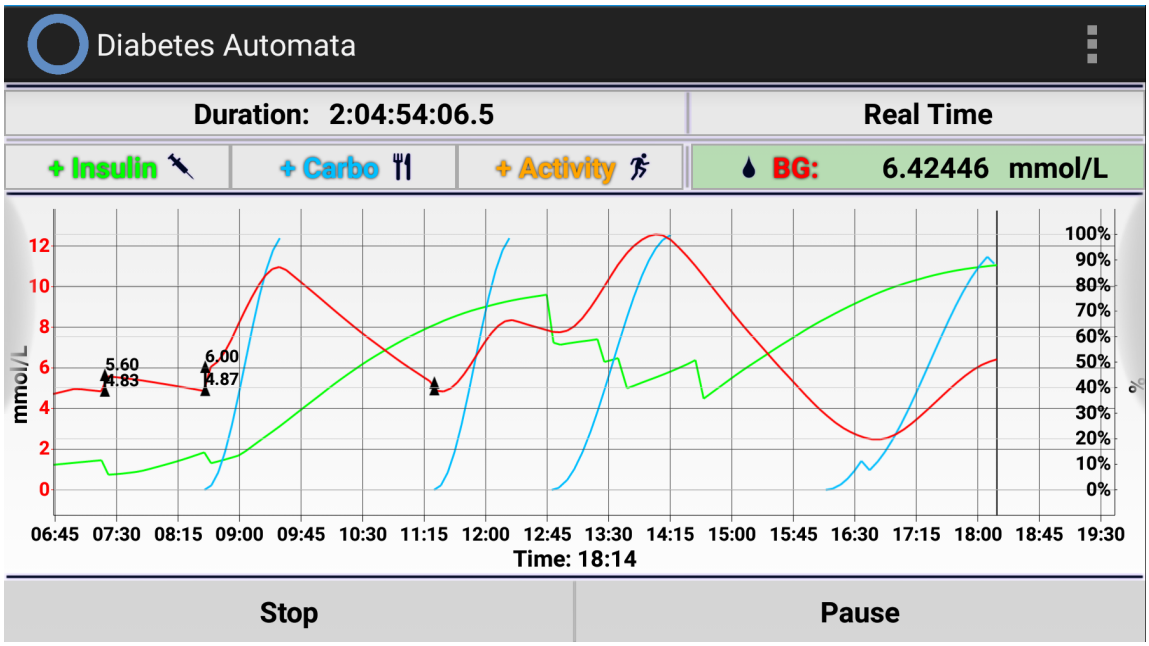


Figure 102: The Main screen of the prototype v0.010 with running simulation, landscape mode.

The graph presents information about Insulin, Carbo, Activity and Blood Glucose with corresponding colors. There are 3 graph modes that are shown in Figure 103. Graph mode change is performed by short-clicking on the graph.

- Mode 1: Blood glucose in mmol/L, insulin and carbohydrates in % work done, activity in % effect strength.
- Mode 2: Blood glucose in mmol/L, carbohydrates in % work done, insulin and activity in % effect strength.
- Mode 3: Blood glucose in mmol/L.

Black triangles joined by a black line represent blood glucose correction that was either made by user in case of simulation mode 1, or by a Diabetesdagboka blood glucose record in case of simulation mode 2 and 3.

Both triangles are marked with values – one value is the previous one, simulated, and the other one is the new, corrected one.

Figure 103: Prototype v0.010 with running simulation, portrait (left) and lanscape (right) orientation, graph mode 1 (left), 2 (middle), 3 (right).

As we have seen in Figure 101 – Figure 103, the Blood Glucose field has green background when the blood glucose levels are in normal range, between 4 and 10 mmol/L.

In cases when it is below 4 mmol/L, the field's background blinks red, and in case of 3 mmol/L or lower the blinking becomes more aggressive. In cases when it is higher than 10, the field's background blinks yellow, and in case of 15 mmol/L or higher the blinking becomes more aggressive. See Figure 104:

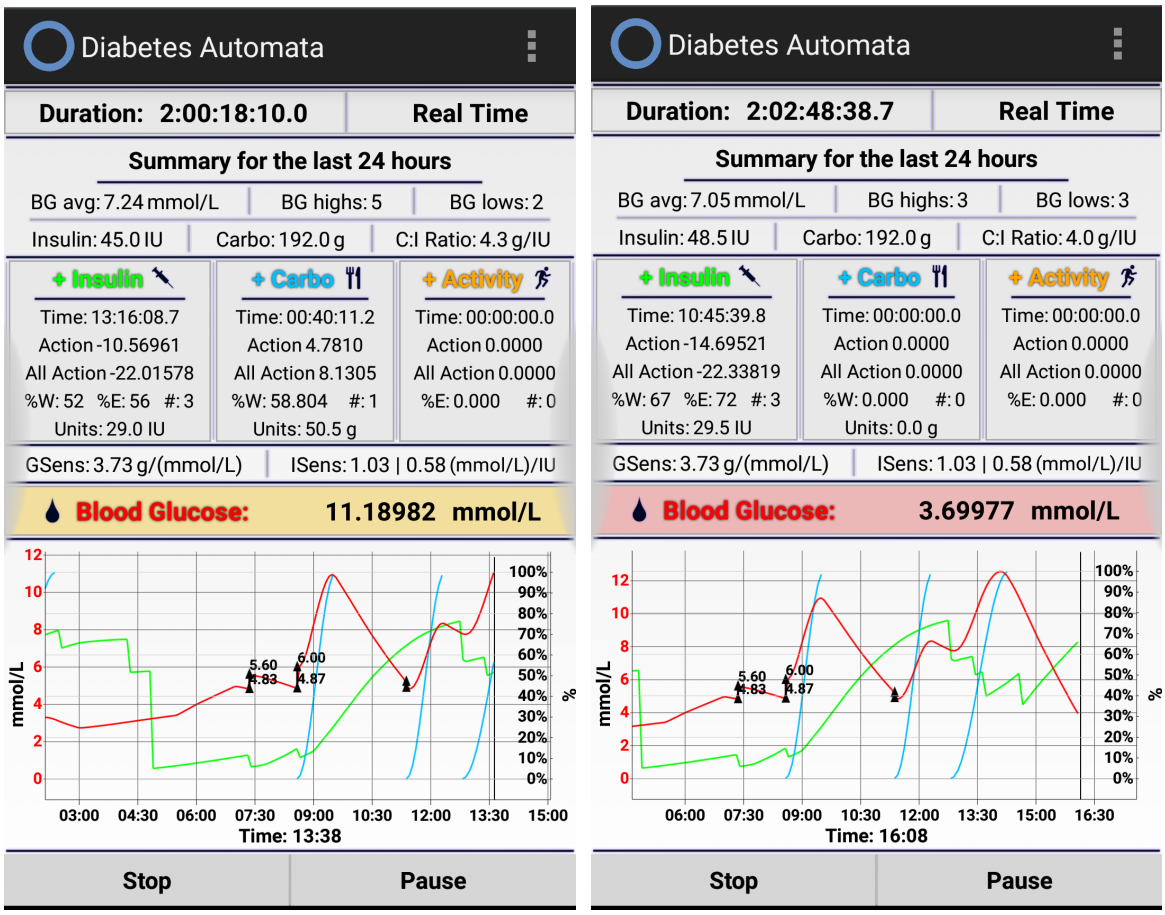


Figure 104: The Main screen of the prototype v0.010 with running simulation, blood glucose level too high (left), blood glucose level too low (right).

In case of simulation mode 1 and 2, the time speed can be changed by pressing on the corresponding button. The user must enter speed value, and choose either seconds per second or minutes per second. In the second case, if the value is bigger than 60, then it will be automatically set to 60 minutes per second – the maximal available speed in the prototype (however, the Engine can accept bigger speeds). See Figure 105:

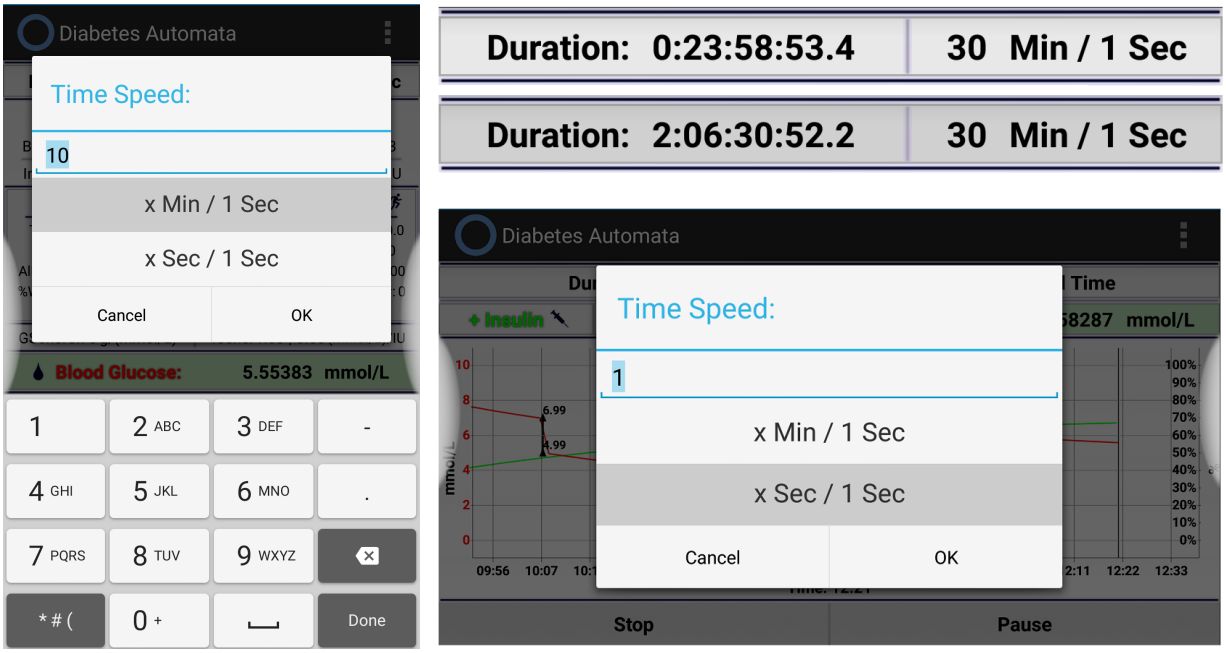


Figure 105: Prototype v0.010, Time Speed / Duration button is not pressed and pressed (upper), new time speed dialog (left), new time speed dialog in the lanscape mode (right).

During the simulation in the mode 1, the process can be managed by setting new simulation records via Insulin, Carbo, Activity and Blood Glucose buttons.

When Blood Glucose button is pressed, the new blood glucose level dialog appears. It provides opportunity to correct the current blood glucose level. See Figure 106:

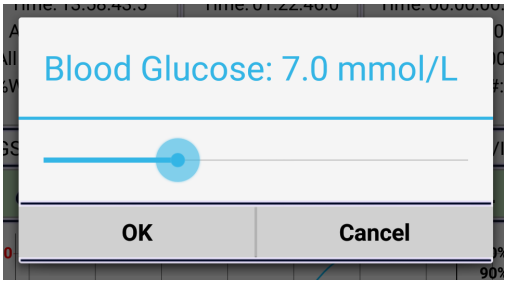


Figure 106: Prototype v0.010, new blood glucose calibration dialog.

When the Insulin button is pressed, the new insulin record dialog appears on the screen. The types of insulin listed in the dialog consists of the types that were defined in the Settings (see section 6.6.2). In order to make the “OK” button active, the type of insulin and the number of IU must be set. See Figure 107.

The same applies to the new carbohydrate and activity record dialog. For the first one, the type of GI and the number of grams must be set. For the second one, the type of activity must be chosen. After that, the “OK” button will be activated. See Figure 108 and Figure 109.

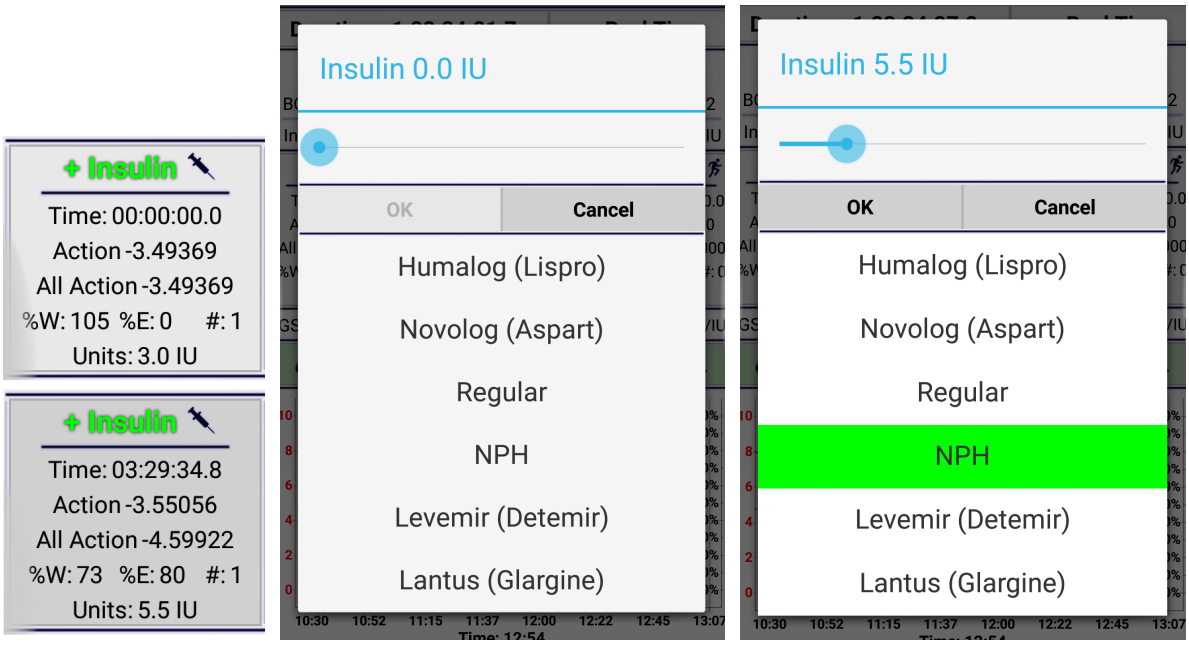


Figure 107: Prototype v0.010, insulin button not pressed and pressed (left), new insulin record dialog apprears (middle), the dialog is ready to set new record (right).

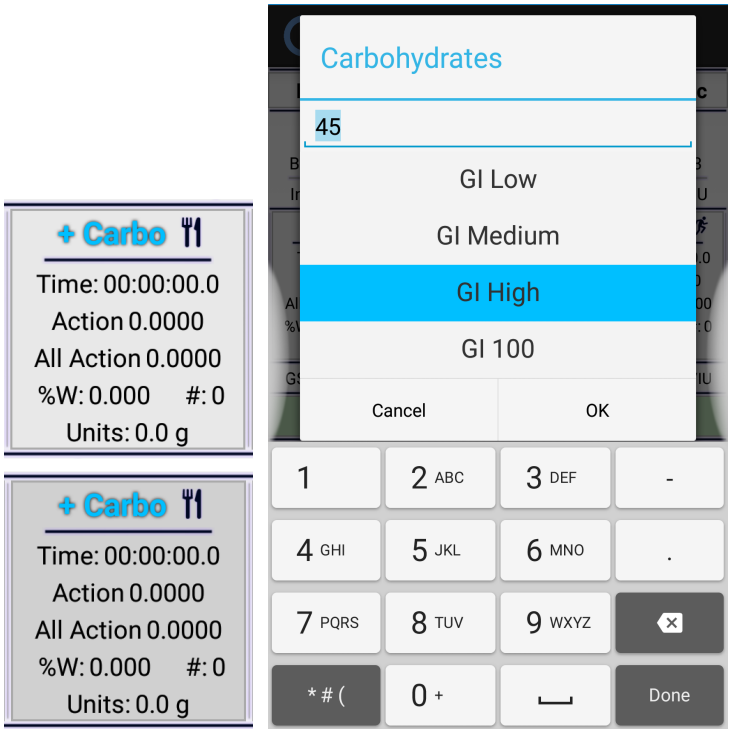


Figure 108: Prototype v0.010, Carbo button not pressed and pressed (left); new carbo record dialog – ready to set new record (right).

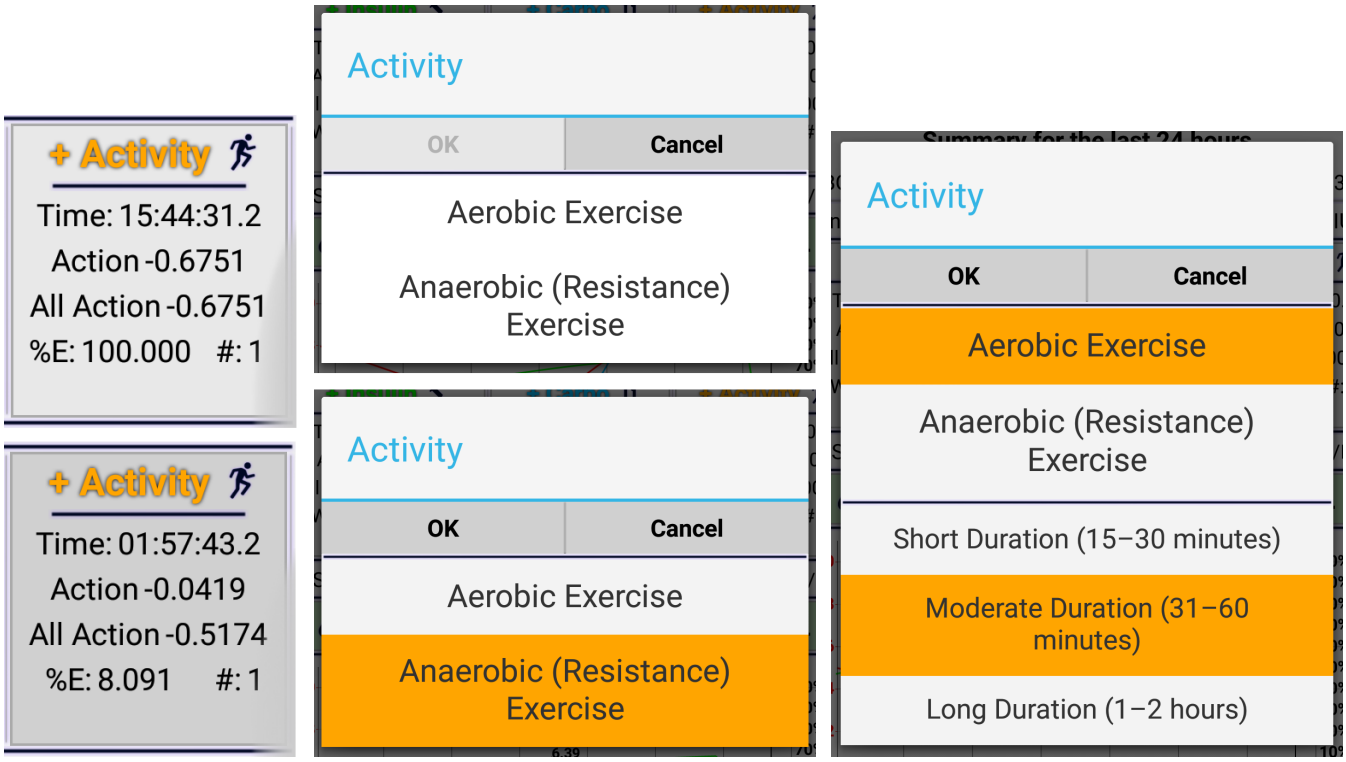


Figure 109: Prototype v0.010, Activity button not pressed and pressed (left), new carbo record dialog apprears (middle upper), the dialog is ready to set new record when aerobic (middle lower) or anaerobic (right) activity is chosen.

The minimal and maximal values for the new blood glucose dialog, and the maximal values for the new insulin and new carbohydrate dialogs, can be changed from the Default Settings screen. See section 6.6.2.

If the simulation is not paused, the graph scale can be changed by performing long click on the graph. See Figure 110 and Figure 111. The list of the scales that can be chosen:

- Automatic
- Last 1 hour
- Last 2.5 hours
- Last 6 hours
- Last 12 hours
- Last 24 hours
- Last 48 hours
- Last 5 days
- Last 10 days
- Last 30 days

By default, the graph scale is automatic – changes depending on simulation speed, until user chooses another scale.

When simulation is set to pause, changing graph mode and graph scale via dialog is disabled, since it becomes possible to scroll and zoom the graph during the simulation pause.

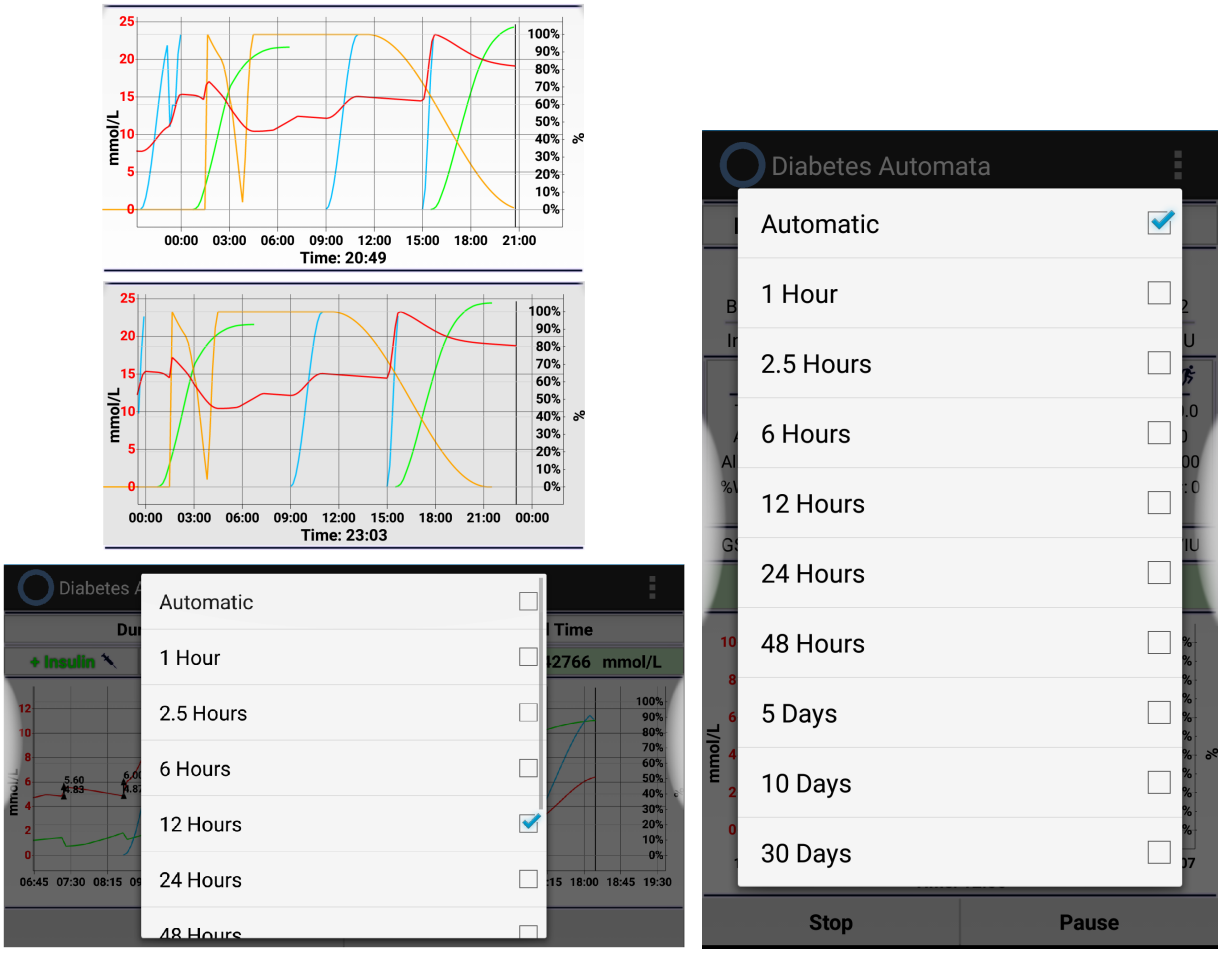


Figure 110: Prototype v0.010, Graph is not pressed and pressed (upper left), change graph scale dialog lanscape (lower left) and portrait mode (right).

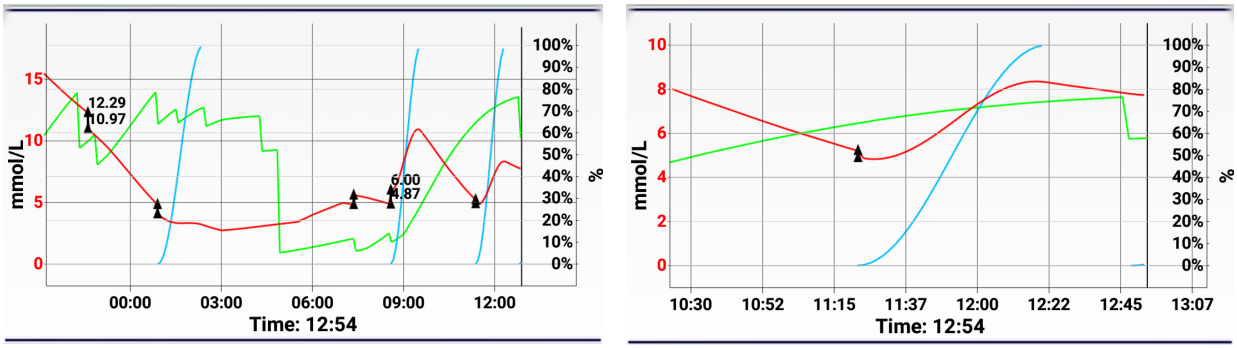


Figure 111: Prototype v0.010 with running simulation, Graph scale 12 (left) and 2.5 hours (right).

The Y-axis that represents number of mmol/L for blood glucose, and % work / effect strength for insulin, carbohydrates and activity, is scaled automatically, from 0 till the Y-coordinate of the maximal visible point.

The options menu of the Main screen when simulation is running, is presented in Figure 112. In addition to the opportunities to "BG < 2.0 mmol/L", "Glucose Consumptions By Brain" and "Dawn Phenomenon" simulation parameters (see section 6.6.3), it provides opportunities to open Settings (see section 6.6.2), save currently running simulation, switch to fullscreen mode, and exit.

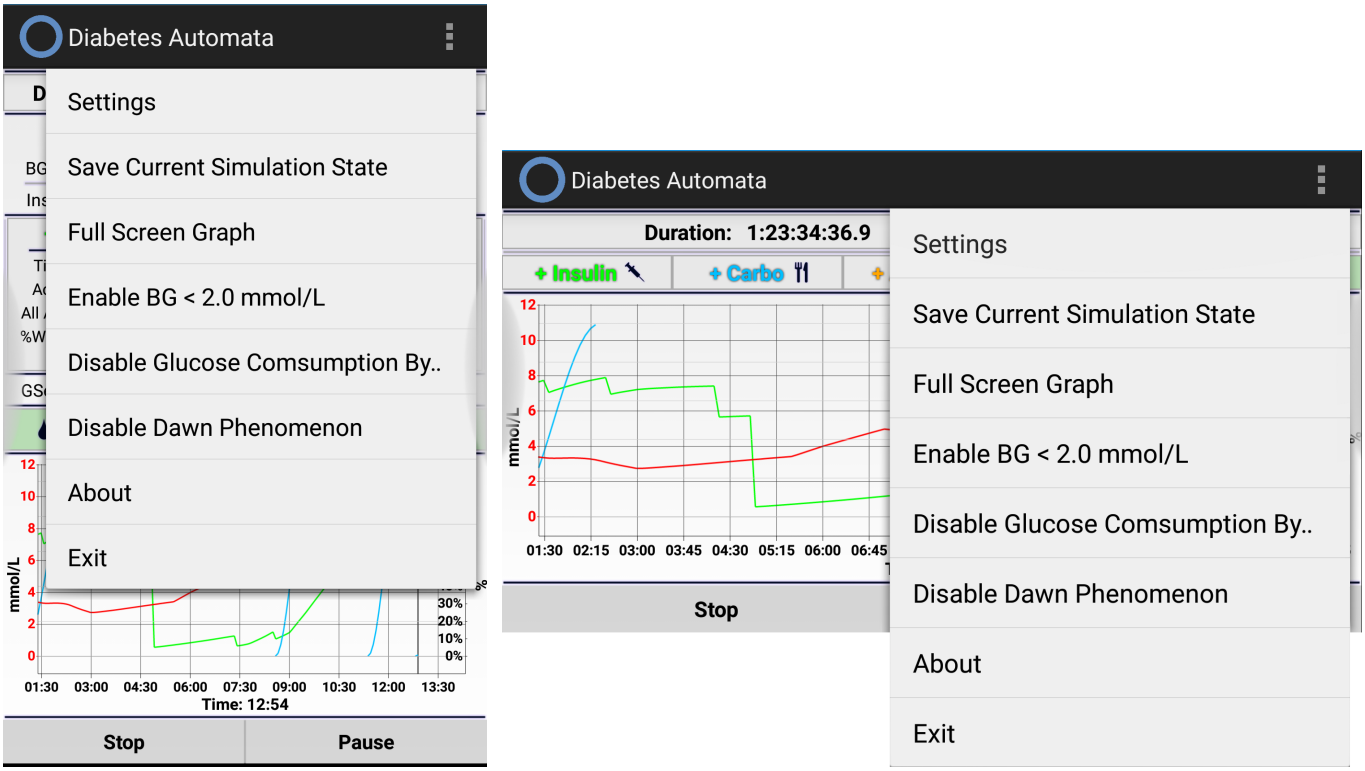


Figure 112: The Main screen of the prototype v0.010 with running simulation and pressed options menu, portrait (left) and landscape mode (right).

When "Exit" is pressed, all processes running by the Engine and the Simulator are interrupted, and the application together with its background service (see section 6.6.5) is stopped. If simulation was running for that moment, it is automatically saved as Interrupted Simulation (see section 6.6.6).

When "Save Current Simulation State" is pressed, the window presented in Figure 113 appears and remains on the screen until the saving process is finished. Simulations in all modes can be saved, however, they all will be saved as simulations in the mode 1. Save files have name "`SA_SaveManual_<date>_<time>.das`" and are located on the phone in "`<internal storage>/DiabetesAutomata/`".

When simulation is not running, the saved simulations can be loaded, looked through, and continued as simulations in mode 1, as said earlier in this section.

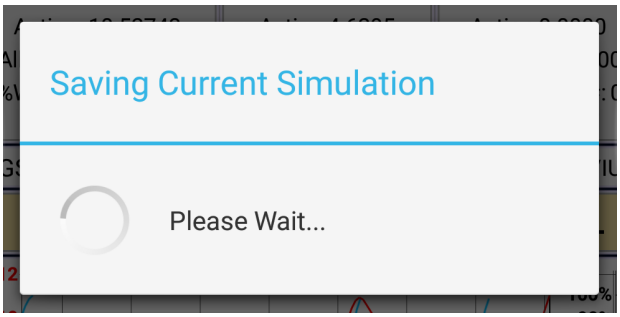


Figure 113: Saving Simulation window of the prototype v0.010.

When “Full Screen Graph” is pressed, the Fullscreen Graph screen is activated. The screen doesn’t have any clickable elements, and is aimed to provide comfortable graph demonstration and monitoring. Probably the most comfortable orientation in this case is landscape, rather than portrait. See Figure 114 and Figure 115:

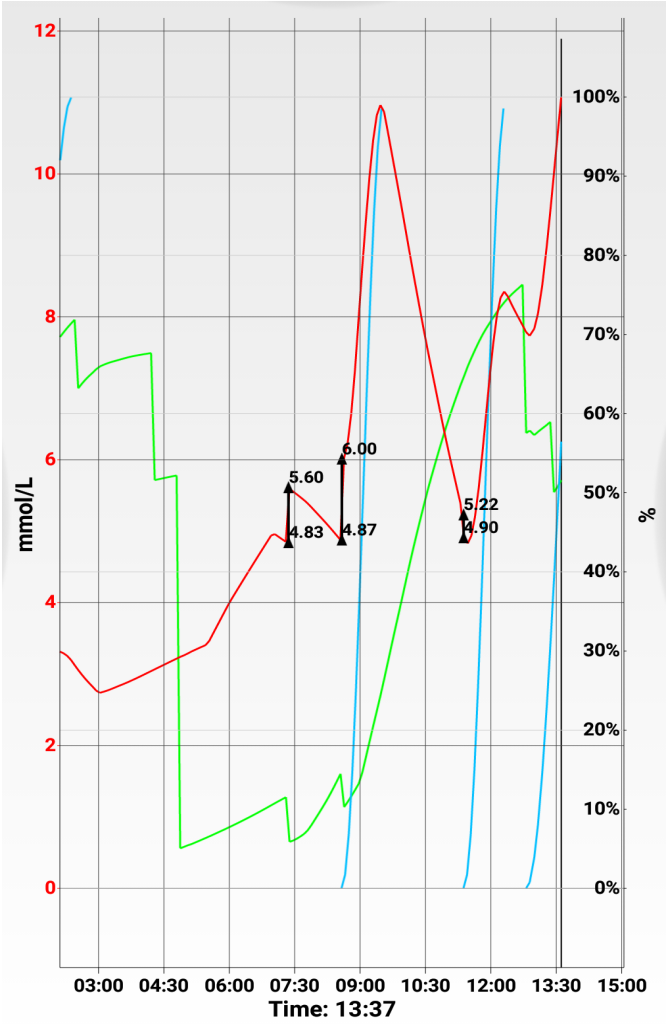


Figure 114: The FullScreen Graph of the prototype v0.010, portrait orientation.

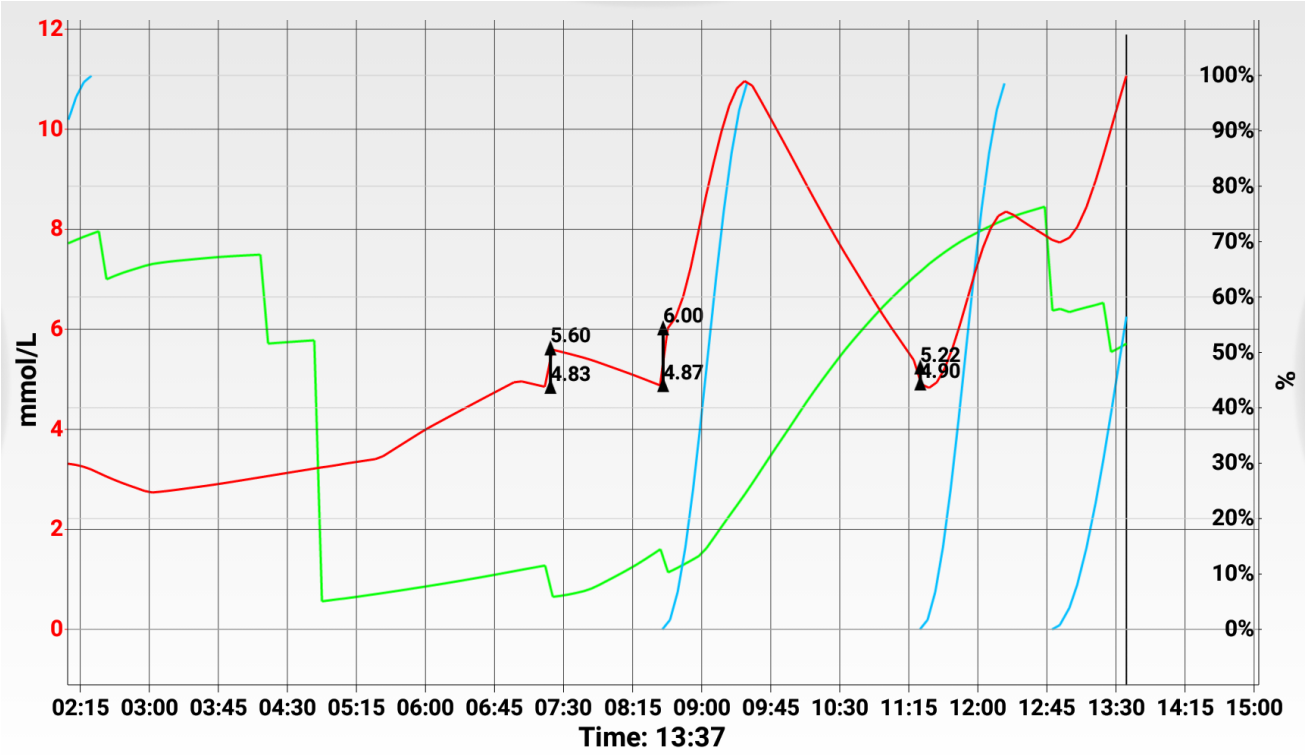


Figure 115: The FullScreen Graph of the prototype v0.010, landscape orientation.

6.6.5. Diabetes Automata Service

Diabetes Automata Service is a very important component of the prototype. It is an Android service (Developer.Android.com, 2015d), which runs in the background, keeps an instance of Engine, runs and keeps all processes that must be always running during the active simulation. In addition, it provides opportunity to implement application notification (Developer.Android.com, 2015c), which shows the application icon in the notification area, and shows full notification message in the notification screen, available when the notification area is pulled down. See Figure 116 – Figure 122.

Figure 116: The notification icon of the prototype v0.010 (the circle on the left) in the Android 5 notification area.

The complete notification looks different when the simulator is launched, when simulation is loaded, running, paused or stopped, and when the simulator is launched after running simulation was interrupted (see section 6.6.6). The notifications for each of these cases are presented below, in Figure 117 – Figure 122:

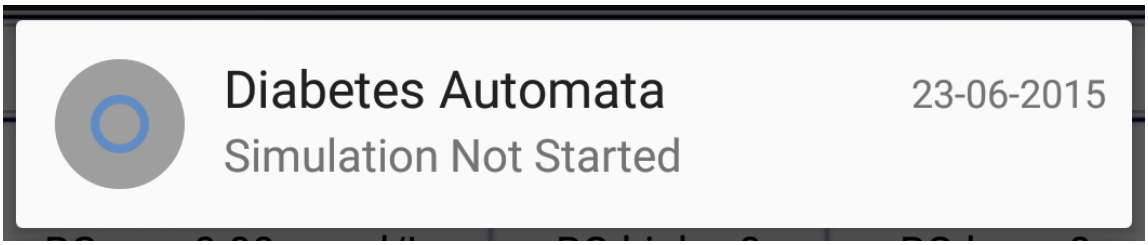


Figure 117: The notification of the prototype v0.010, when launched.

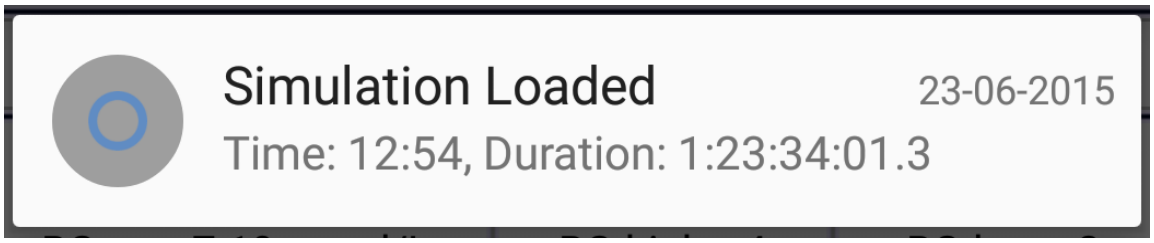


Figure 118: The notification of the prototype v0.010, when simulation is loaded.

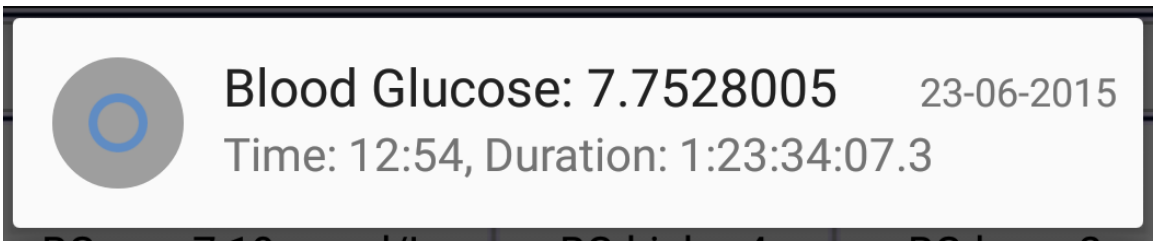


Figure 119: The notification of the prototype v0.010, when simulation is running and not paused.

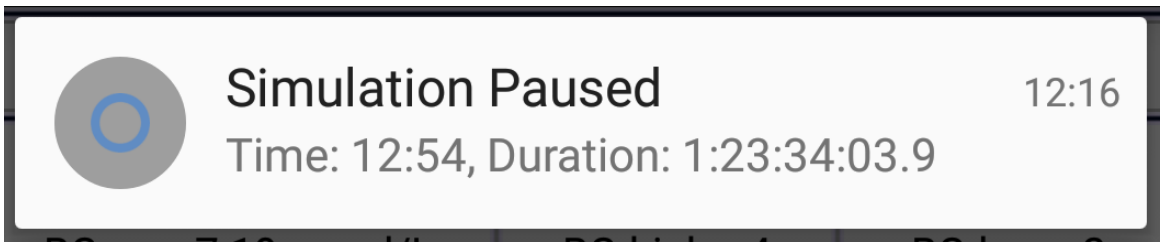


Figure 120: The notification of the prototype v0.010, when simulation is running but paused.

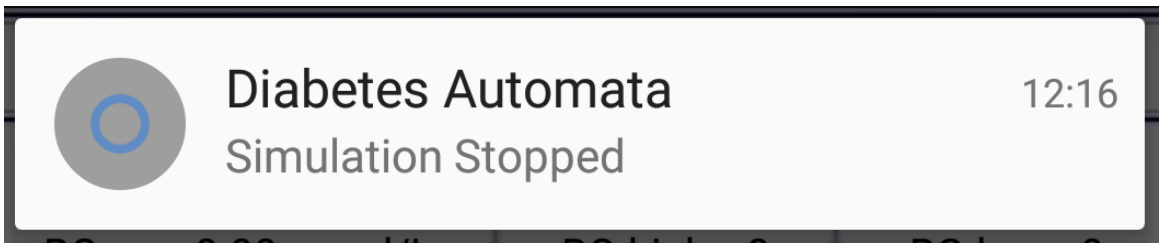


Figure 121: The notification of the prototype v0.010, when simulation is stopped.

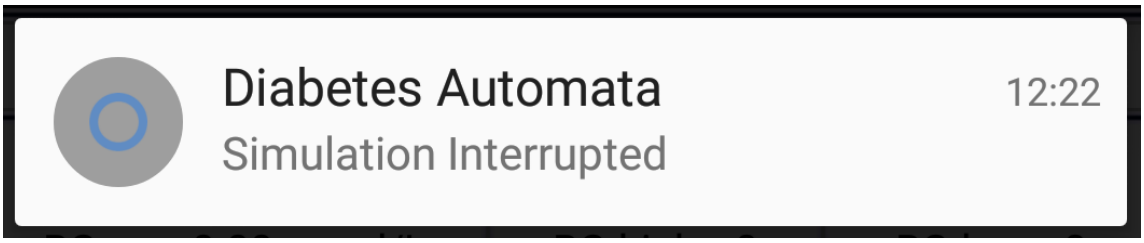


Figure 122: The notification of the prototype v0.010, when launched after previously running simulation was interrupted.

In Figure 118 – Figure 120, Time is the simulation-time of the day between 00:00 and 23:59, and Duration is the total duration of the simulation counted in simulation time.

"RemoteServiceConnection" and "IncomingHandler" (see Figure 93) are the sub-classes used in the simulation modes 2 and 3 for the integration with Diabetesdagboka via its IPC service.

6.6.6. Simulation Interrupted

Situations when running simulation can be interrupted, can possibly occur. For example, if Android stops the background service presented in section 6.6.5 for some reason, or if user forgets to save and stop the simulation before pressing "Exit", or if the phone is about to get powered off soon (battery level is below 2%), or in case of unfortunate application crash.

If one of these cases takes place, the running simulation is automatically saved as interrupted simulation in simulation mode 1 (in case of crash or very low battery, the necessary routines for saving the interrupted simulation are executed in the ExceptionHandler class and Receiver class respectively (see Diabetes Automata Service class in Figure 93)). Later, when the prototype is launched again, the simulation recovery dialog appears on the screen. See Figure 123 (left).

Interrupted simulation can be either deleted (in this case the confirmation dialog is shown, see Figure 123 (right)) or loaded. In the second case, the loaded interrupted simulation is set to pause, can be looked through, and continued in the simulation mode 1.

This interrupted simulation saving feature could be especially useful during testing, in order to not lose the results of several days of testing, in case if the running simulation was interrupted for some reason.

In addition, in case of application crash, the information about the crash is written to a text file with timestamp, and saved on the phone in "`<internal storage>/DiabetesAutomata/CrashLog/`", to make it possible for the author to see what has happened, and find the reason that caused the crash.

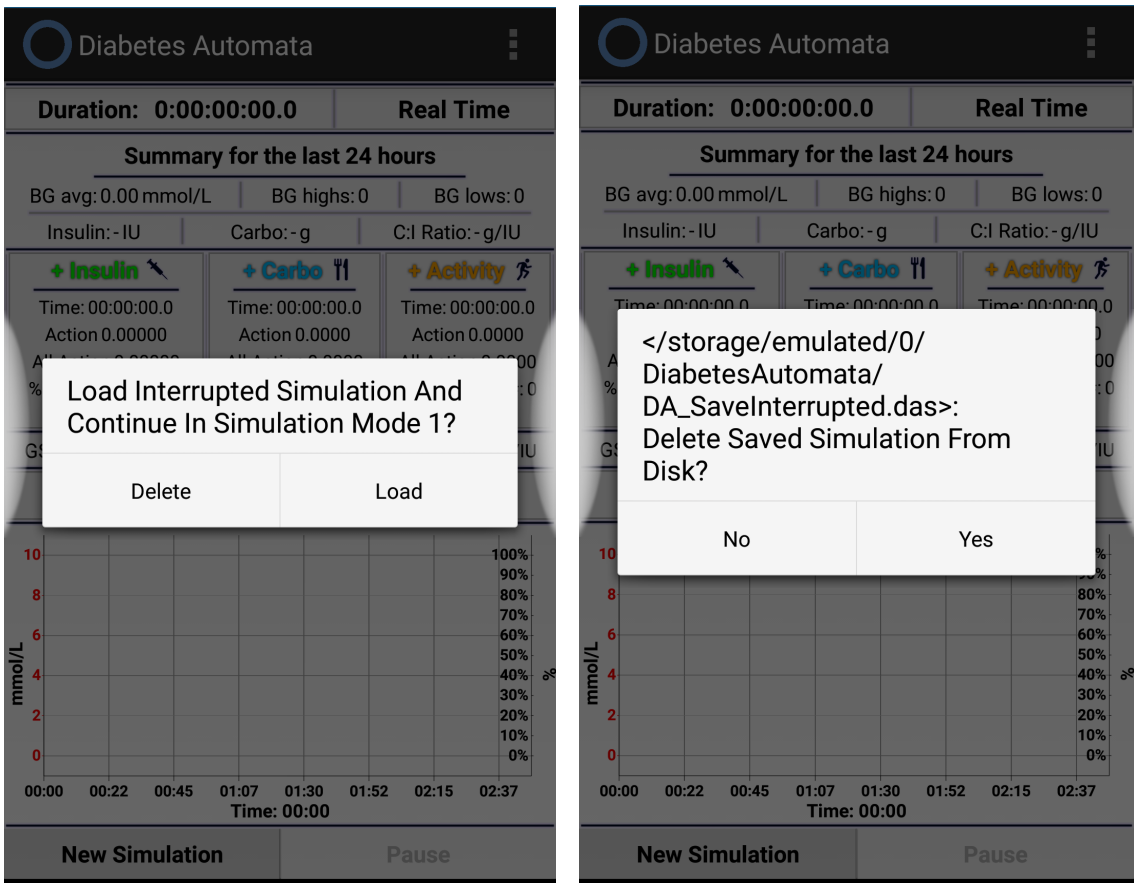


Figure 123: Load Interrupted Simulation dialog (left) and delete interrupted simulation confirmation dialog (right) of the prototype v0.010.

6.6.7. Other

Other elements presented in Figure 93 are About screen, Launcher activity, Utils, Time Preference and Date Preference.

About screen doesn't have any functionality, it just shows the name of the project, the version of the prototype, and author's name. It can be opened from the options menu of the Main, Settings and Default Settings screens.

Launcher activity is the first activity launched when the Demonstrator application is launched. We don't call it "screen" because it doesn't have any interface. Its only function is to start the Diabetes Automata Service (if it is not already running) and open the Main Screen.

Utils is a collection of helping static methods. Most of them are aimed to create and get string in desired format.

Time Preference is the class used for implementation and display of a time setting when it is clicked. The time setting where the class is used, is "Starting Time" parameter of the New Simulation screen (see Figure 97).

Date Preference is the class used for implementation and display of a date setting when it is clicked. The date setting where the class is used, is "Birth Date" parameter of the Settings and Default Settings screens (see Figure 95 and Figure 96).

6.7. Summary

In this chapter, we have discussed how the prototype is implemented. In particular, we have discussed what technologies were used to develop the prototype – Java programming language, mobile OS Android, and Eclipse for Java Developers with Android Developer Tools plugin.

After that, we have thoroughly discussed the implementation of the Diabetes Automata engine in Java, with all its inner components, structures and modules. The Main module provides a part of the API for external applications, implements time management and runs Insulin, Carbohydrate, Activity and Blood Glucose modules, each of which uses various methods provided in the collection of calculation formulas and equations, and important constants defined in corresponding static class. Settings hold the biometric information and general simulation settings, and also provide another part of the API for external applications. State keeps an instance of Settings, as well as the entire collection of the simulation data for the last 30 simulation-days. Database module was designed to keep the inactive simulation data, but unfortunately is not implemented in the current version of the prototype.

In the last sub-chapter, we have thoroughly discussed the implementation of the UI in (currently) last version of the prototype. Three simulation modes are provided – usual simulation (where user make all simulation records), database history simulation using integration with Diabetesdagboka (which is the simulation of the Diabetesdagboka records for the last 30 days), and real-time simulation paired with Diabetesdagboka (where new Diabetesdagboka records are automatically set into the running real-time simulation).

The screens of the UI are: the Main screen (which demonstrates the simulation process, shows detailed information about the running simulation, and provides simulation control), the New Simulation screen (which provides opportunity to choose between 3 new simulation modes and change the new simulation settings), Settings screen (which provides opportunity to see and change the biometric information and new simulation parameters, and to set default values with one click), Default Settings screen (which is used to change the default settings), and About screen (which shows the version number of the prototype).

Also, we have looked at the implementation of the Diabetes Automata Service, which is a very important component in the Demonstrator UI. Finally, we have seen how the simulation interrupt is handled.

7. Testing, Results and Discussion

As said in 3.4, there were three kinds of project testing: feedback from testers during the development, simulation trial by testers, and testing of the engine algorithms with testers' data. This chapter presents the process and results of them.

7.1. Testing

Two test persons with diabetes type 1 have been testing the system. The first person uses Humalog before eating, and 16 IU of Lantus every midnight as long-acting insulin. The second person uses Humalog before eating, and an insulin pump that makes continuous micro-injections of Humalog in the following way: 1 IU per hour during 3:00-7:00, 0.9 IU per hour during 7:00-18:00, 1 IU per hour during 18:00-22:00, and 0.6 IU during 22:00-3:00. The prototype was modified in order to simulate the described behavior of the pump.

The other types of insulin (Novolog, Regular, NPH and Levemir) were not tested.

During the development, the author has been receiving constant feedback from test person 1. During the last month of the development, when test person 2 was found, this person has been also giving feedback about the prototype. The results of this first type of testing are presented in section 7.2.

The second kind of testing was to send a copy of currently last version of prototype to the test persons, ask them to run it during several days; give feedback about how accurate the blood glucose simulation algorithms work and how big the deviations are; save the test simulation and send the saved file to the author.

This kind of testing was done in 2 different ways – using simulation mode 1 and simulation mode 3. In the first case, testers would have to do double work – make new simulation records and Diabetesdagboka records (since they all were actively using Diabetesdagboka for their diabetes self-management); however, it would provide more accurate results, since the testers have choice between different types of insulin, carbohydrate and activity that can be registered in the demonstrator prototype. In the second case, the process would be easier for the testers, since records registered in Diabetesdagboka are automatically imported into a running simulation; however, the test accuracy would be not as good as in the first case, since the process of simulation in mode 3 is automated as follows:

- Diabetesdagboka insulin record is imported as a rapid-/short-acting insulin with the number of IU given in the Diabetesdagboka record. The tester is asked about the type of insulin.
- If tester uses long-acting insulin (and doesn't register it in Diabetesdagboka), the daily injection simulation record is automatically made around midnight. The tester is asked about the number of IU and the type of long-acting insulin.
- If tester uses insulin pump, its micro-injections can be automated as well. The tester is asked about the type of insulin and the number of IU that the pump injects over time. A new simulated micro-injection is made automatically every 30 minutes.
- Diabetesdagboka carbohydrate record is imported as carbohydrate with GI medium and the number of grams taken from the record.

- Diabetesdagboka activity record is imported as aerobic activity, with the duration taken from the record.

Since the correct number of GI is important, and the test persons have been setting GI level almost without knowing the actual GI of food (and in case of simulation mode 3, the GI was always set to medium – GI 60), this kind of testing is aimed more to figure out what in the engine should be calibrated and how – a preparation-testing before the third type of testing.

Finally, in the third type of testing, the test persons were asked to create a 2-day sample database in Diabetesdagboka, while collecting additional detailed information about insulin types they have used, pictures of food they have eaten, and information about the kinds of activity, for these 2 days when the database has been created. After receiving the database and the additional information, the author hardcoded personal parameters so be set automatically during the database simulation (the parameters such as types of insulin and activity, as well as the amount and injection time of long acting insulin the test person 1, and the simulation of insulin pump's action for test person 2), wrote information about carbohydrate GI in the comment field of each carbohydrate record in Diabetesdagboka (the information about GI was found on several sources (Atkinson et al., 2008; Mendosa, 2008; Harvard Health Publications, 2015; Thompson, 2014; Weight Loss Resources, 2015)), and ran the simulation on the sample database, in order to see the accuracy and deviations of blood glucose levels prediction by himself. A blood glucose record in the database is used as blood glucose correction when running simulation of the database, and there should be 1 blood glucose record in 2-8 hours, so that the simulated blood glucose level is not always calibrated, but also it doesn't go too far from the real blood glucose level and is corrected once in this defined period.

This is the most important kind of prototype testing, in comparison to the second kind of testing, and its results show how accurate the simulation results are. The sample database from test person 1 was containing data for a little bit more than 3.5 days. The sample database from test person 2 was containing data for 1.5 days.

During the second and the third kind of testing, the prototype was used as a testing tool, since the prototype works with databases from Diabetesdagboka, can save and load simulations, scale the simulation graph in different ways and show it in fullscreen mode. The fullscreen mode of the prototype was used to make the figures presented in the section 7.2.

In order to organize the second and the third kind of testing, the following text together with the instructions presented in "Appendix A: Instructions" were prepared for the test persons:

Test 1, 2 days: with Diabetesdagboka (Diabetes Diary), without prototype.
Create a 2-day sample database in Diabetesdagboka, with all insulin, carbohydrates and activity records for these 2 days. While making the database, take pictures of each eaten meal (or, in other words, for each Diabetesdagboka carbohydrate record). Two days after, send the database, the pictures and the information about what types of insulin have been used and what types of activity (aerobic or anaerobic) has been done (as alternative, the info about the last 2 could be written into the "comment" field of corresponding Diabetesdagboka record) (as alternative for making pictures of food, the information about GI of the food could be written into the "comment" field of corresponding Diabetesdagboka carbohydrates record). The developer will hard-code the provided information and run simulation with the provided sample database, in order to test the accuracy of the prototype.

Test 2, 2-3 days: with prototype.
Run simulation in the first mode and see how accurate it works.

The user should not change the time speed or press "pause" (in order to keep the simulation in real-time) and make new records manually in the simulator (this will provide choice for types of insulin, carbohydrates and activity, and as the result - better and more accurate testing than in the Test 3). A manual "Blood Glucose" calibration by the tester should be made 2-3 hours after the last one (except the night time of course), so that the predicted blood glucose value will not be always correct and calibrated (the reason for calibrating blood glucose less often than once in 2 hours), and its deviation will not go too far in case of incorrect prediction.
When the testing is finished and the "Stop" button is pressed, the tester will see the dialog "Save current simulation before stopping?": press "Save", otherwise the results will be lost.
Give general message-feedback about the accuracy, and send the saved simulation file(s) that can be found in "<Internal Storage>/Android/data/no.telemed.diabetesautomata/files/data/".

Test 3, 1-2 days: with prototype and Diabetesdagboka.
Run simulation in the third mode and see how accurate it works.
New records made in Diabetesdagboka, will be automatically set into the simulation (so that the tester will not have to do double-work like in the Test 2, however, the results will not be as accurate as in the Test 2).
When the testing is finished and the "Stop" button is pressed, the tester will see the dialog "Save current simulation before stopping?": press "Save", otherwise the results will be lost.
Give general message-feedback about the accuracy, and send the saved simulation file(s) that can be found in "<Internal Storage>/Android/data/no.telemed.diabetesautomata/files/data/".

The example-pictures showing how the first test person was doing testing type 2, can be seen in Figure 124 – Figure 126.

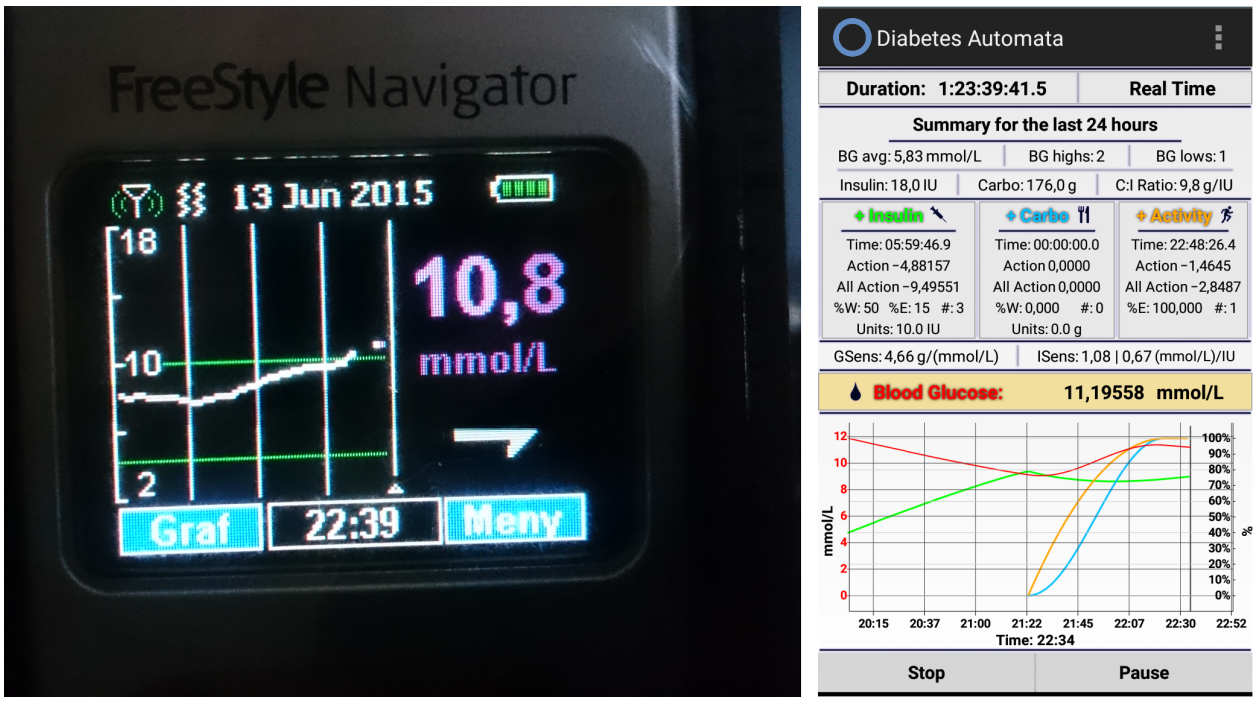


Figure 124: Testing type 2 by the first test person, glucose monitoring gadget on the left side, prototype v0.009a of the right side.

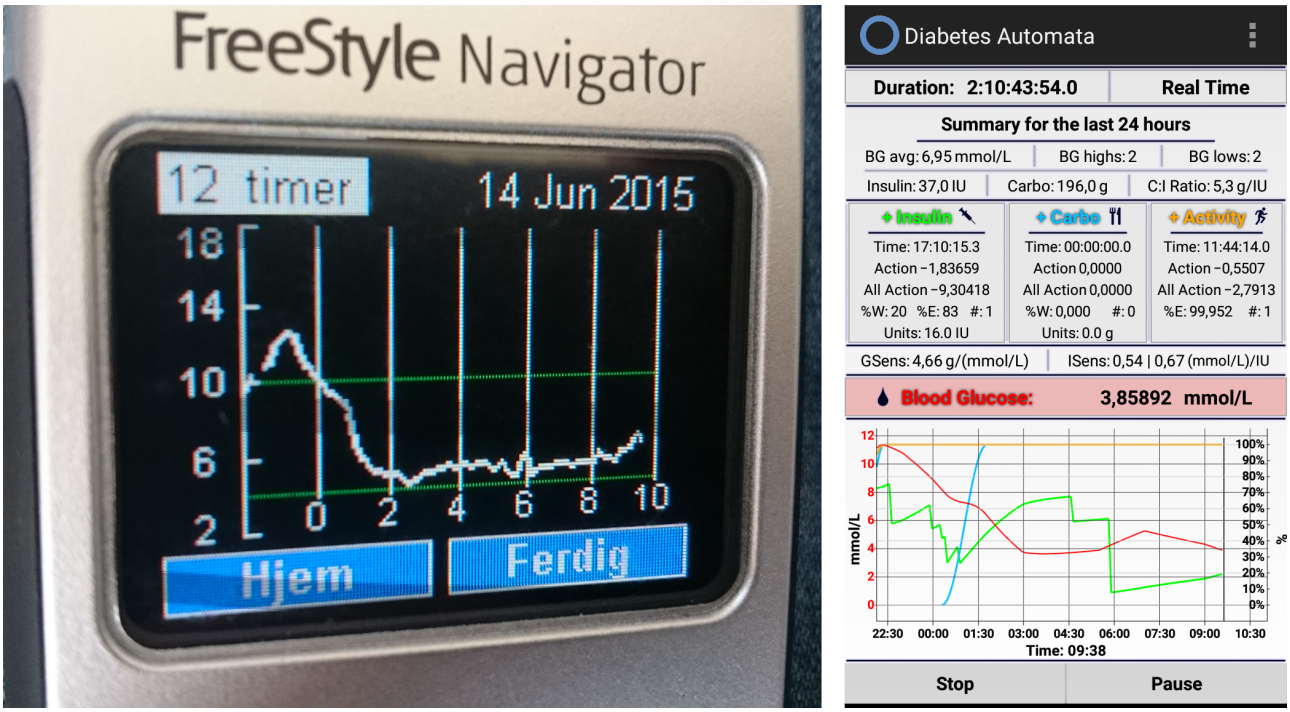


Figure 125: Testing type 2 by the first test person, glucose monitoring gadget on the left side, prototype v0.009a of the right side.

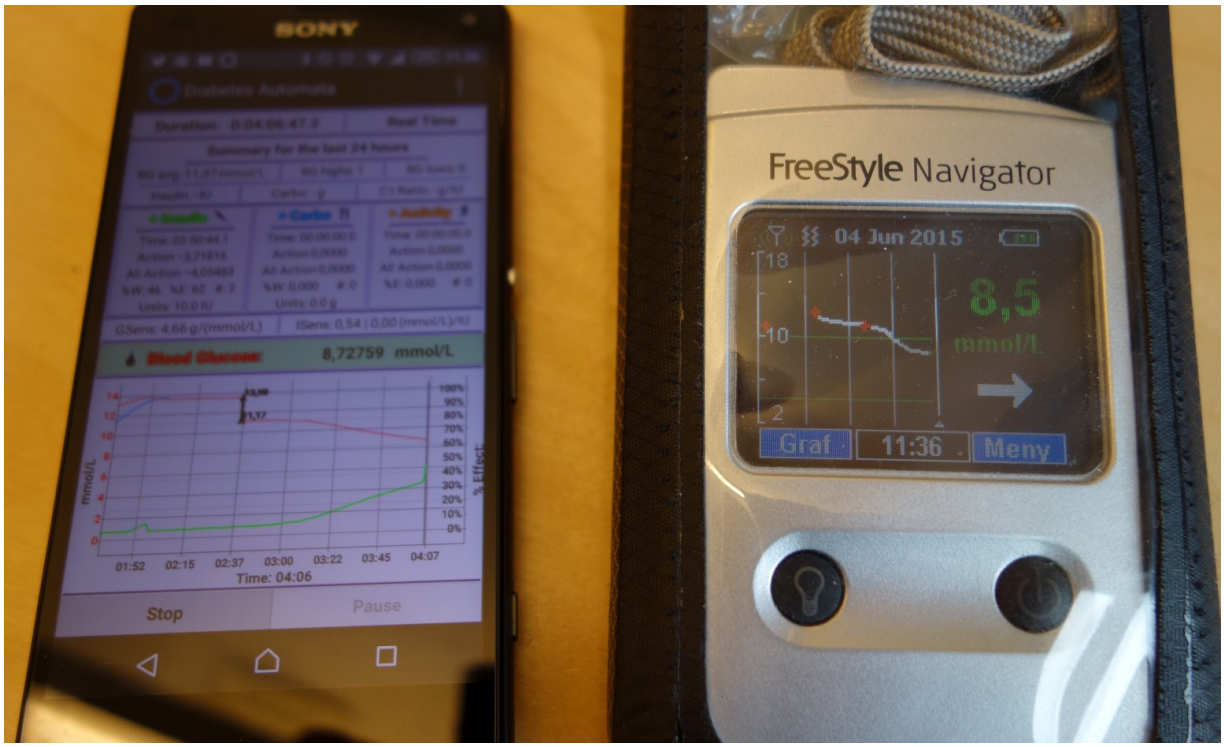


Figure 126: Testing type 2 by the first test person, glucose monitoring gadget on the right side, prototype v0.008g of the left side.

7.2. Results

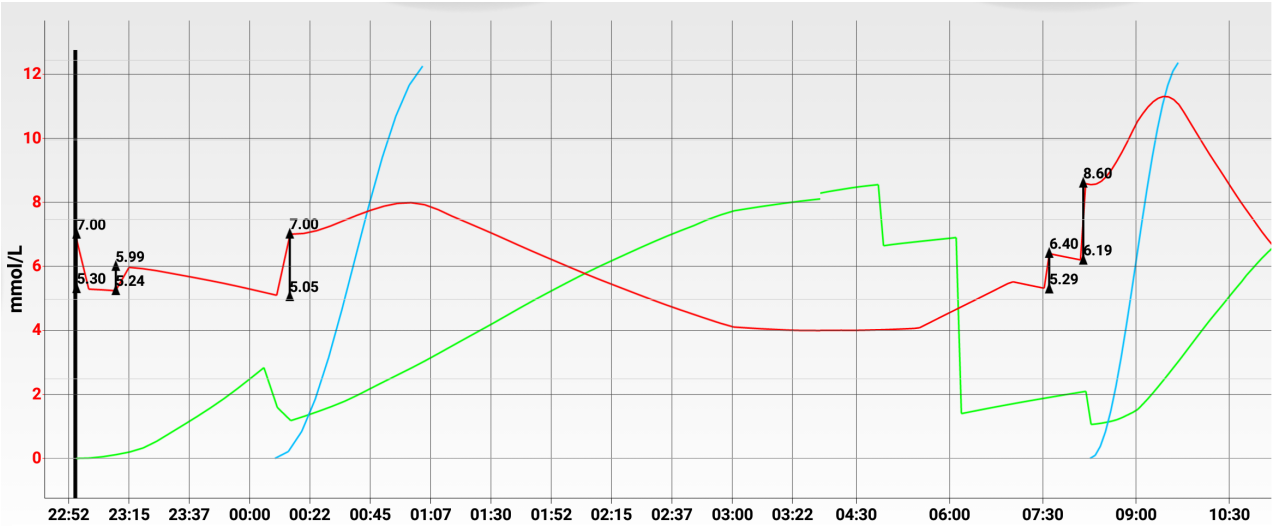
Results of the first kind of testing:

The results were constant improvement and correction of the prototype during the whole development period with help of the feedbacks from the test persons. The algorithms and the demonstrator's UI were becoming better, the prototype was becoming more usable, stable and was getting more and more necessary features and functionality. Among the improvements are:

- The incorrect behavior of insulin module which was reducing blood glucose too much. Big re-implementation of all algorithms in insulin module was done in prototype version 0.007a (the version history draft is presented in "Appendix C: Prototype Version History").
- Various calculation inaccuracies.
- Various general bugs and crashes.
- Bugs, crashes and problems that were occurring on Android 5 and not occurring on Android 4.4.
- Issues occurring in simulation mode 2 and 3, where the integration with Diabetesdagboka via its IPC service is used.
- The use of too much CPU-time and, as the consequence, too big energy consumption that was causing the test person's mobile phone to discharge very fast.
- Issues when the usage of the UI was unclear, of when the usage of some elements of the main screen is not practical. The feedback about that, was the reason for making all clickable elements of the Main screen look more like buttons (see Figure 59 – Figure 66), and for changing the type of new carbohydrate dialog from progress-bar (see Figure 34) to the usual kind of dialog (see Figure 108), like new time speed dialog (Figure 105).

Results of the second kind of testing:

The results from the first try (16.06.2015., using prototype v0.009b, simulation mode 1) by the first test person are presented in Figure 127:



This figure continues on the next page.

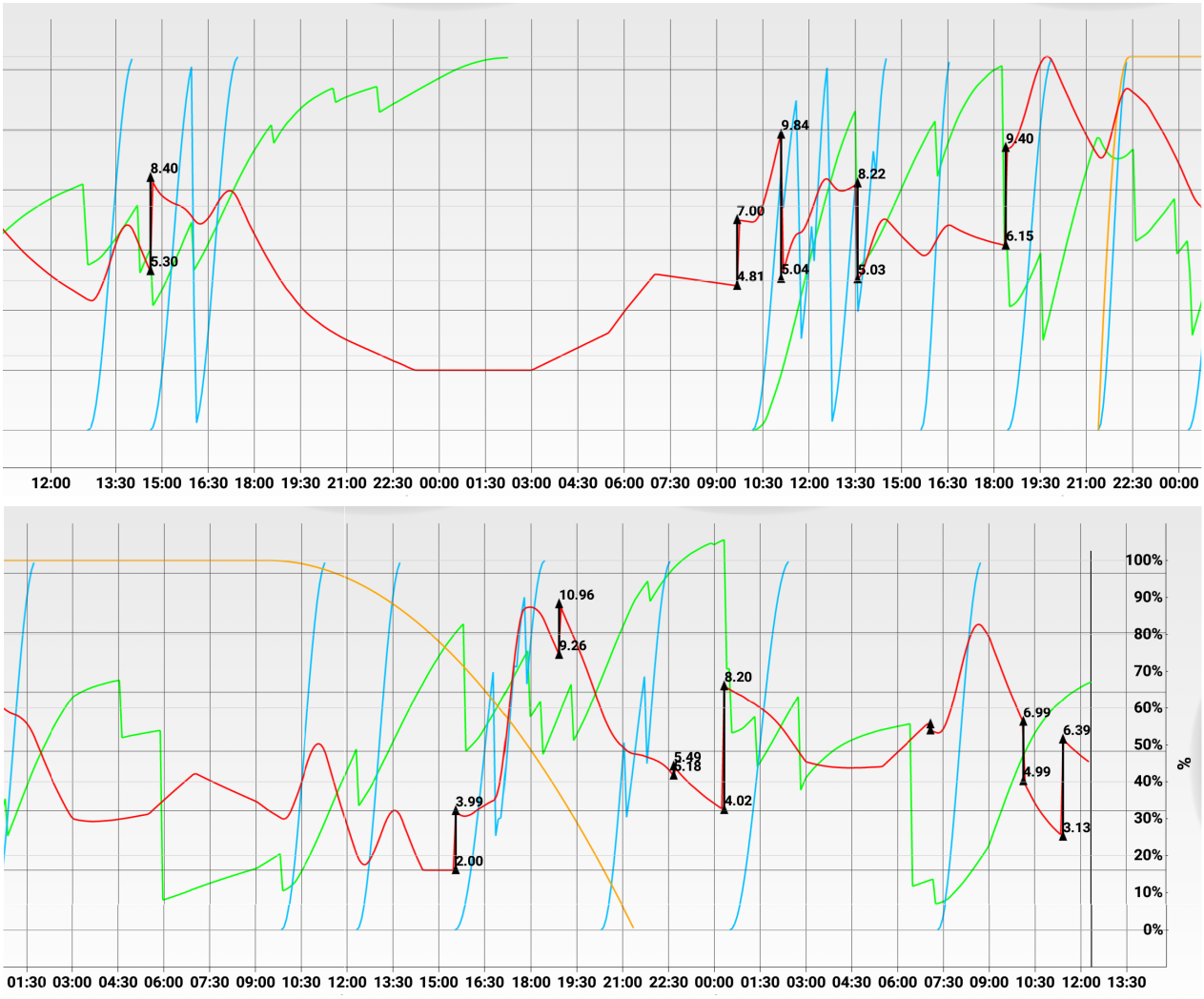


Figure 127: Second kind of testing, test #1 (16.06.2015., prototype v0.009b, simulation mode 1), test person 1.

Total number of blood glucose calibrations: 17.
Number of deviations that are almost 0 (between 0 and 0.3): 1, 6%.
Number of deviations between 0.3 and 0.8: 2, 12%.
Number of deviations between 0.8 and 1.5: 1, 6%.
Number of deviations between 1.5 and 2.5: 7, 41%.
Number of deviations bigger than 2.5: 6, 35%.

The results have shown that insulin sensitivity is too big, the blood glucose increase caused by the dawn phenomenon should be a little bit bigger during time between 3:00 and 5:30, and 50%-reduced insulin sensitivity should last between 2:00 and 8:00 instead of between 2:00 and 7:00. The following changes were made: the Humalog peak-effect constant was decreased, the effect of aerobic activity was reduced from 30% to 15%, the hourly blood glucose increment was increased by 0.1 mmol/L for time between 3:00 and 5:30, and the 50%-reduced insulin sensitivity became 1 hour longer.

The results from the second try (18.06.2015., using prototype v0.009d, simulation mode 1) by the first person are presented in Figure 128:

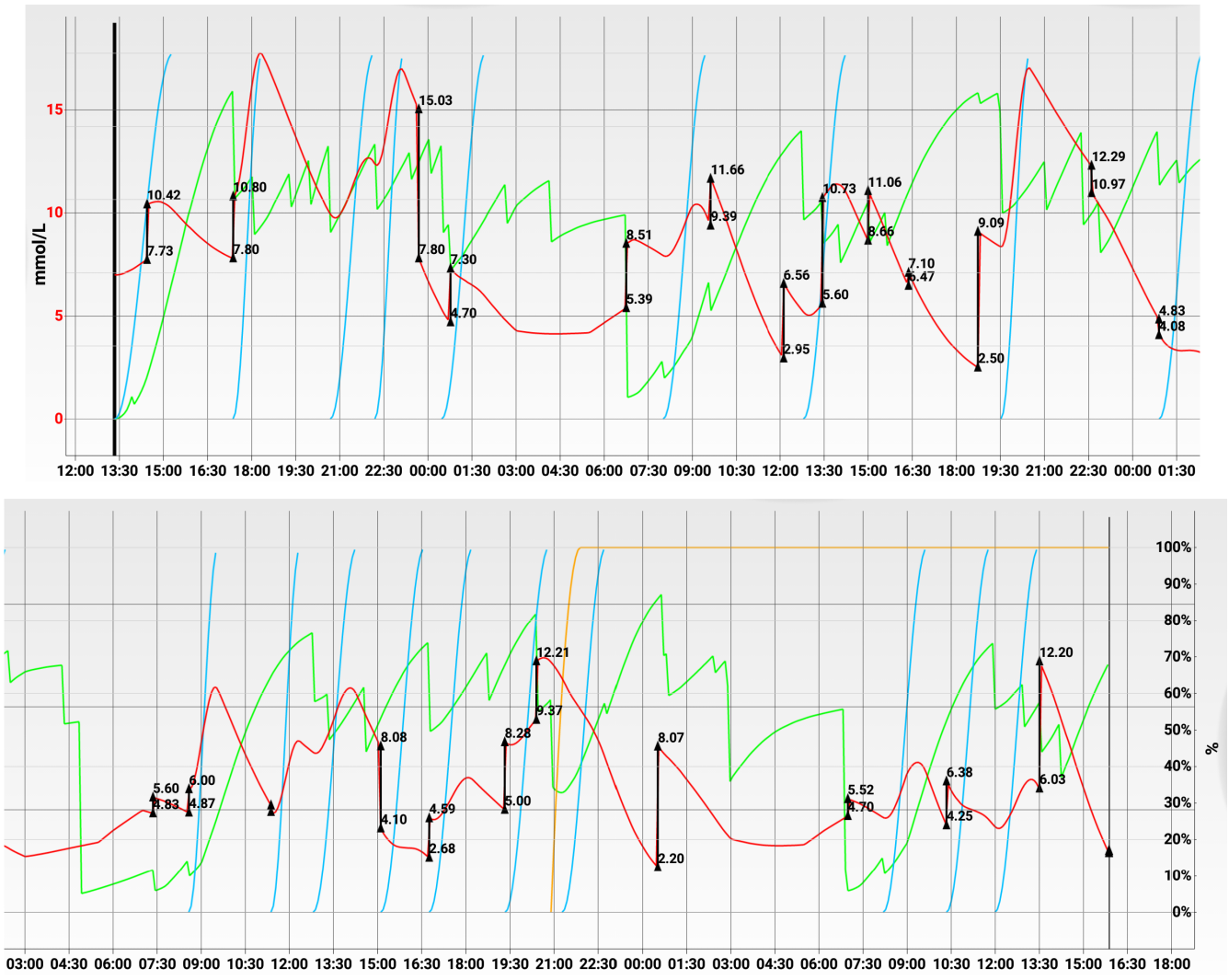


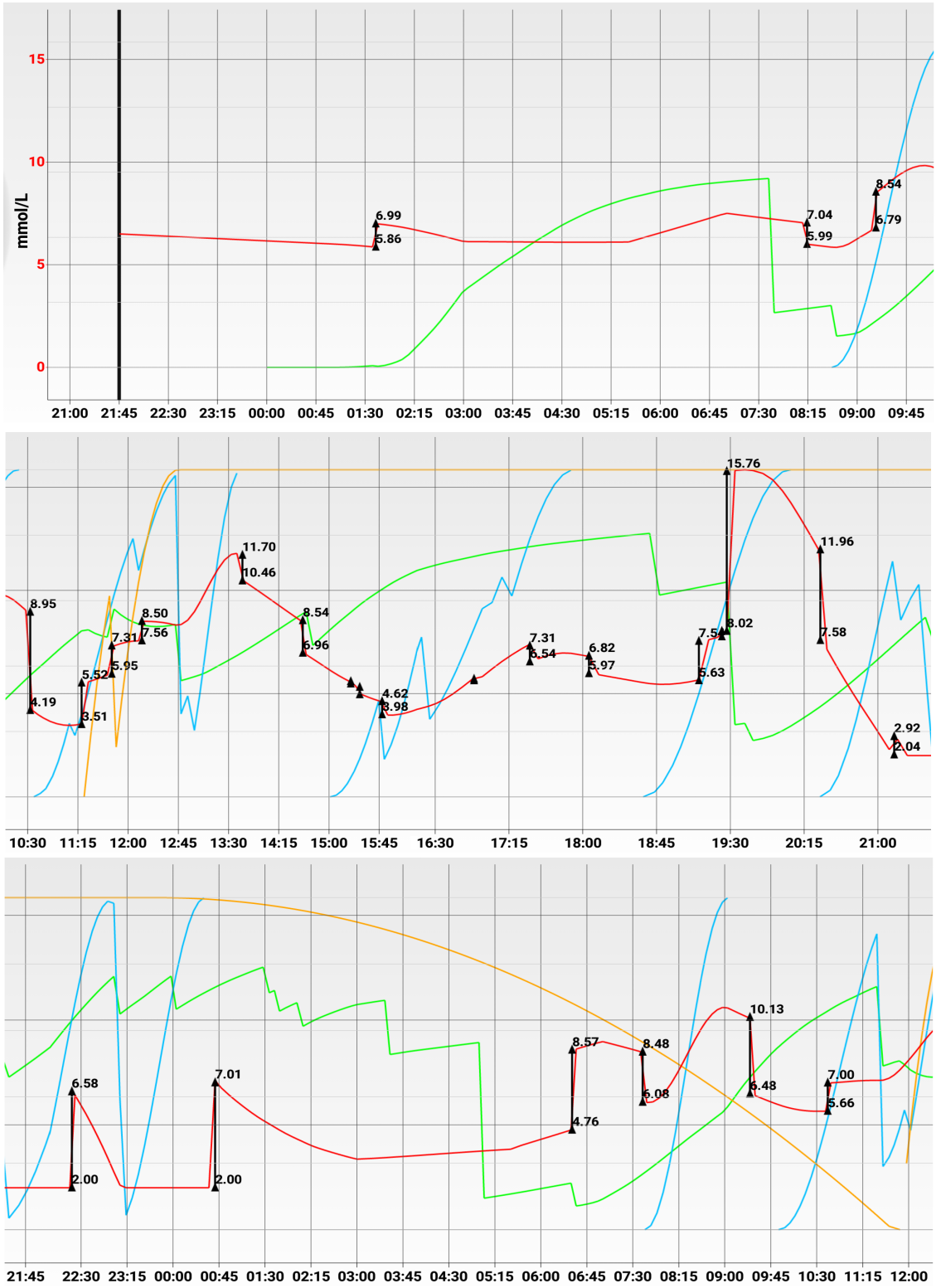
Figure 128: Second kind of testing, test #2 (18.06.2015., prototype v0.009d, simulation mode 1), test person 1.

Total number of blood glucose calibrations: 25.
Number of deviations that are almost 0 (between 0 and 0.3): 2, 8%.
Number of deviations between 0.3 and 0.8: 4, 16%.
Number of deviations between 0.8 and 1.5: 2, 8%.
Number of deviations between 1.5 and 2.5: 4, 16%.
Number of deviations bigger than 2.5: 13, 52%.

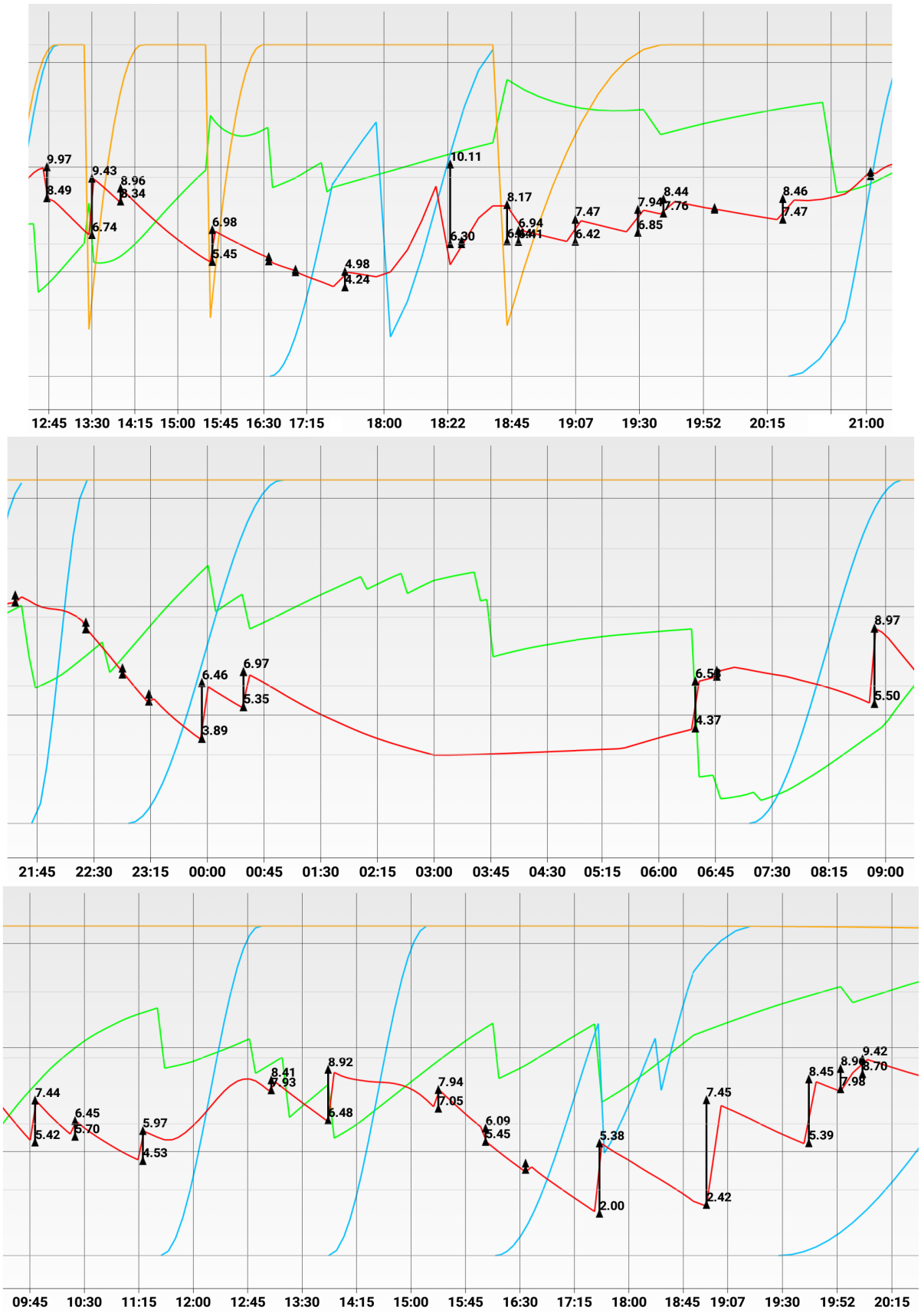
The results showed that insulin sensitivity was still higher than it should be, the behavior of Humalog shown in Figure 80 that was based on Figure 5 is not so accurate, and glucose sensitivity was lower that it should be, which means that calculation based on the data presented in Table 1 underestimates glucose sensitivity. The following changes were made: Humalog action curve was re-implemented according to Figure 8 and is presented in Figure 81. The peak-effect constant for Humalog and Lantus were corrected, glucose sensitivity was increased by 20%.

Results of the third kind of testing:

The results from the first execution of the sample database testing with test person 1 using prototype v0.009b, are presented in Figure 129:



This figure continues on the next page.



This figure continues on the next page.

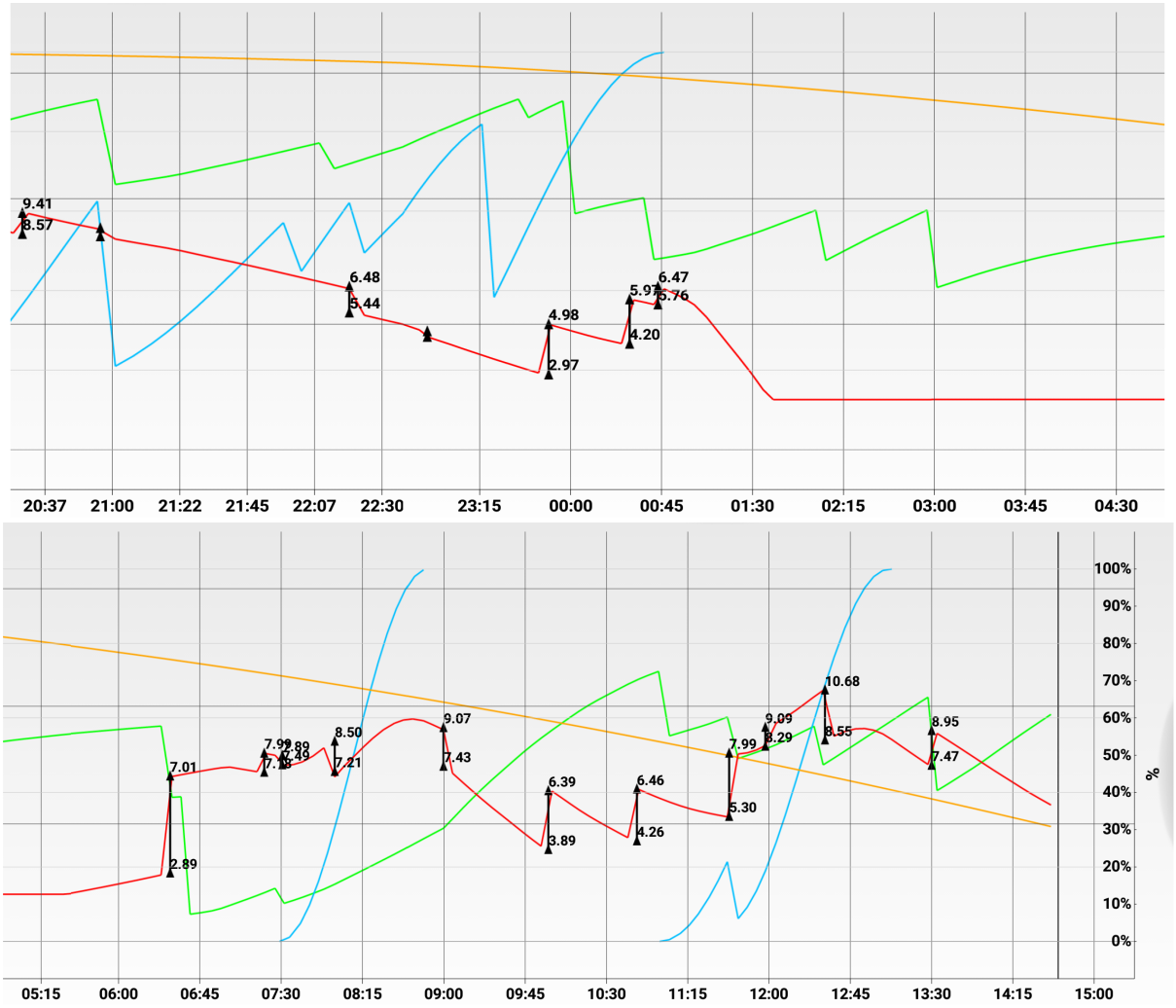
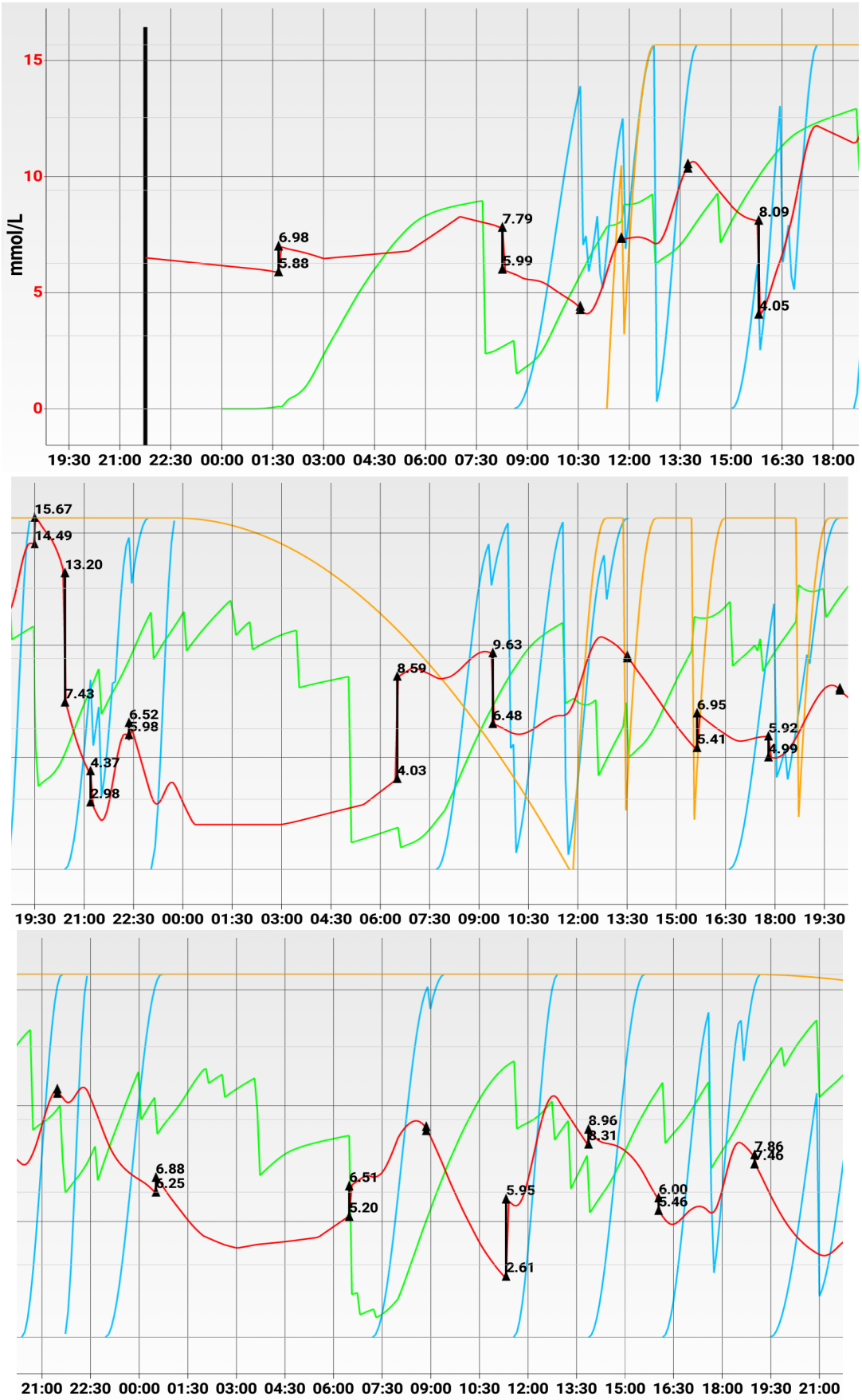


Figure 129: Third kind of testing with test person 1, execution #1 (prototype v0.009b).

Total number of blood glucose calibrations: 83.
Number of deviations that are almost 0 (between 0 and 0.3): 17, 20.5%.
Number of deviations between 0.3 and 0.8: 13, 16%.
Number of deviations between 0.8 and 1.5: 17, 20.5%.
Number of deviations between 1.5 and 2.5: 20, 24%.
Number of deviations bigger than 2.5: 16, 19%.

As we see, the first execution was messed up because of too many Diabetesdagboka blood glucose records – too many blood glucose calibrations, and as the consequence, always corrected blood glucose level, which should not be done too often.

After deleting some blood glucose records from the sample Diabetesdagboka database, so that there is only 1 Diabetesdagboka blood glucose record per 2-8 hours, analyzing the accuracy problems described above in this section, and correcting the GIs in the Diabetesdagboka carbohydrate records, the sample database testing with test person 1 was executed again with prototype v0.009e. The results are presented in below, in Figure 130:



This figure continues on the next page.

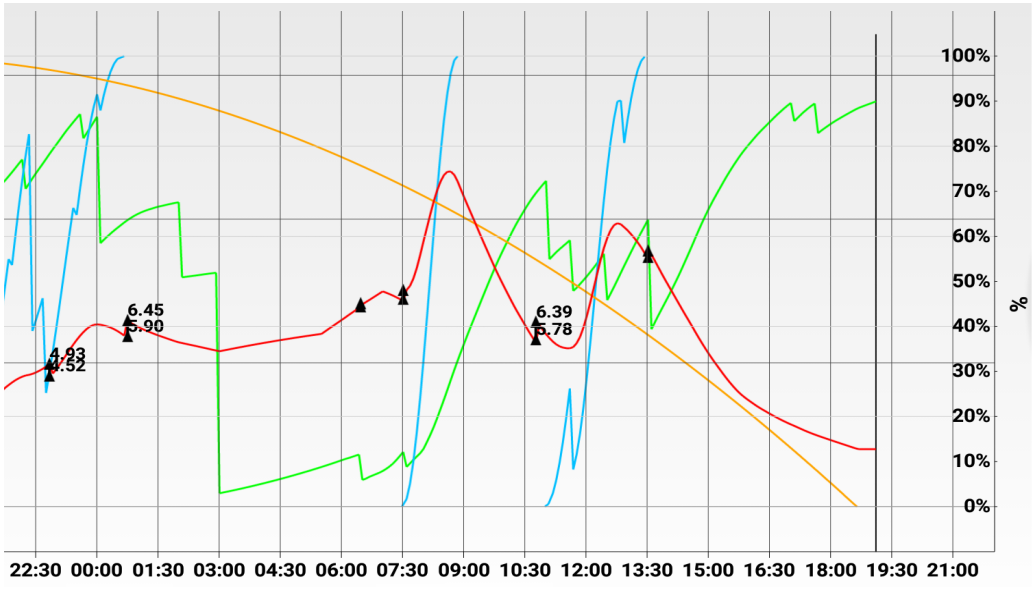


Figure 130: Third kind of testing with test person 1, the last execution (prototype v0.009e).

Total number of blood glucose calibrations: 30.
Number of deviations that are almost 0 (between 0 and 0.3): 10, 33.(3)%.
Number of deviations between 0.3 and 0.8: 8, 27.(6)%.
Number of deviations between 0.8 and 1.5: 5, 16.(6)%.
Number of deviations between 1.5 and 2.5: 2, 6.(6)%.
Number of deviations bigger than 2.5: 5, 16.(6)%.

As we see, it became much easier to understand the results, the blood glucose calibrations was taking place in a limited amount of time, and the results became much more accurate because of the changes between v0.009b to v0.009e, which are described above in this section.

Let us take a look at the testing results for the second test person. As mentioned before, the second person uses insulin pump instead of long-acting insulin.

The results from the first execution of the sample database testing with test person 2 using prototype v0.009e, are presented below, in Figure 131.

Total number of blood glucose calibrations: 14.
Number of deviations that are almost 0 (between 0 and 0.3): 0, 0%.
Number of deviations between 0.3 and 0.8: 0, 0%.
Number of deviations between 0.8 and 1.5: 1, 7%.
Number of deviations between 1.5 and 2.5: 1, 7%.
Number of deviations bigger than 2.5: 12, 86%.

As we see, the results for the second test person are much worse than the results of the last database testing execution for the first test person.

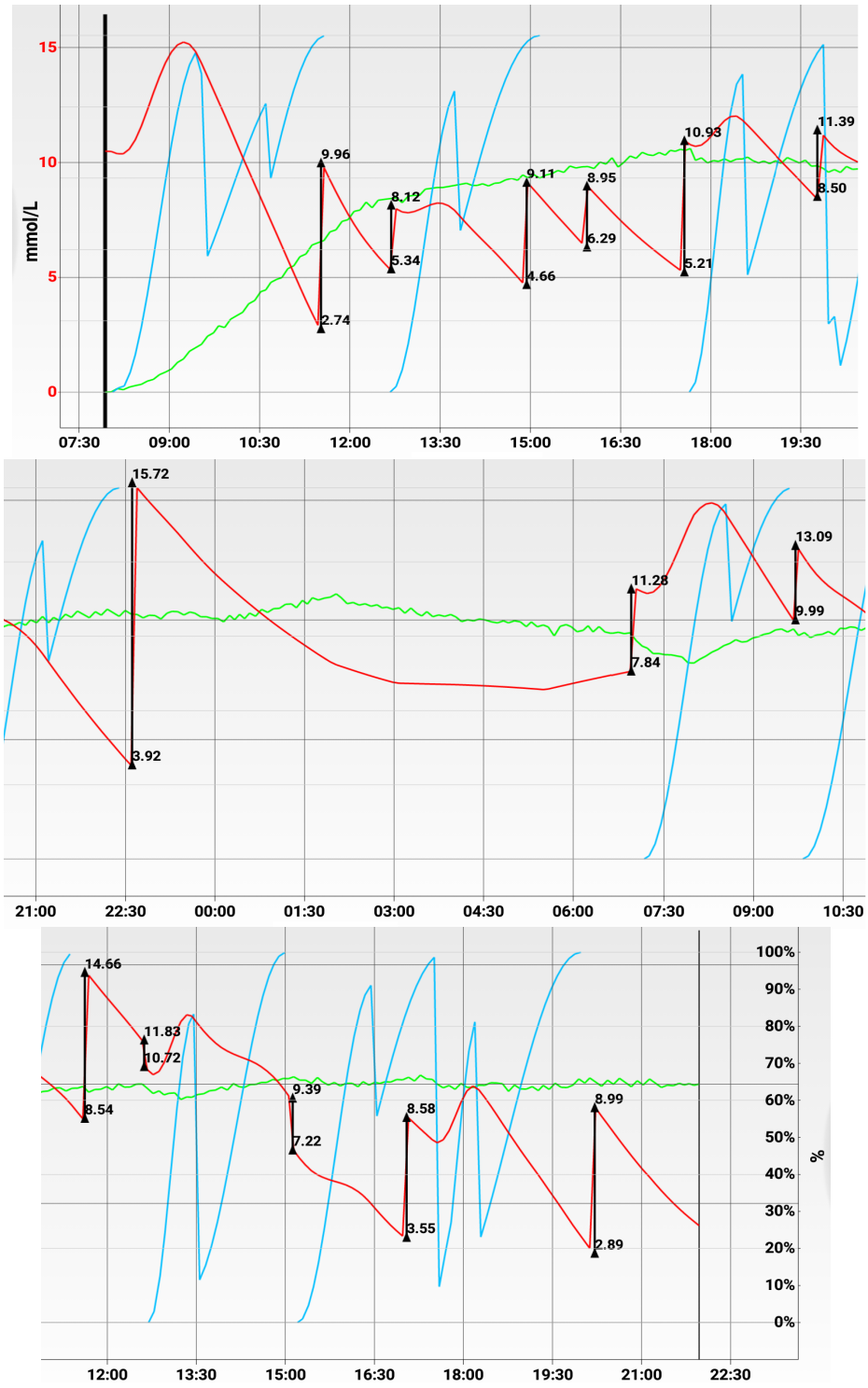


Figure 131: Third kind of testing with test person 2, execution #1 (prototype v0.009e).

In the next try, insulin sensitivity was multiplied by a constant. The constant is 0.6, and it reduces the strength of insulin effect by 40%. The reasons of such big deviations in the first execution of the testing, as well as the reason for applying this constant, are discussed in section 7.3. Discussion. In addition, the GI of the carbohydrate records were corrected a little bit. The results from the new execution of the sample database testing with test person 2 using prototype v0.009h, are presented below, in Figure 132:

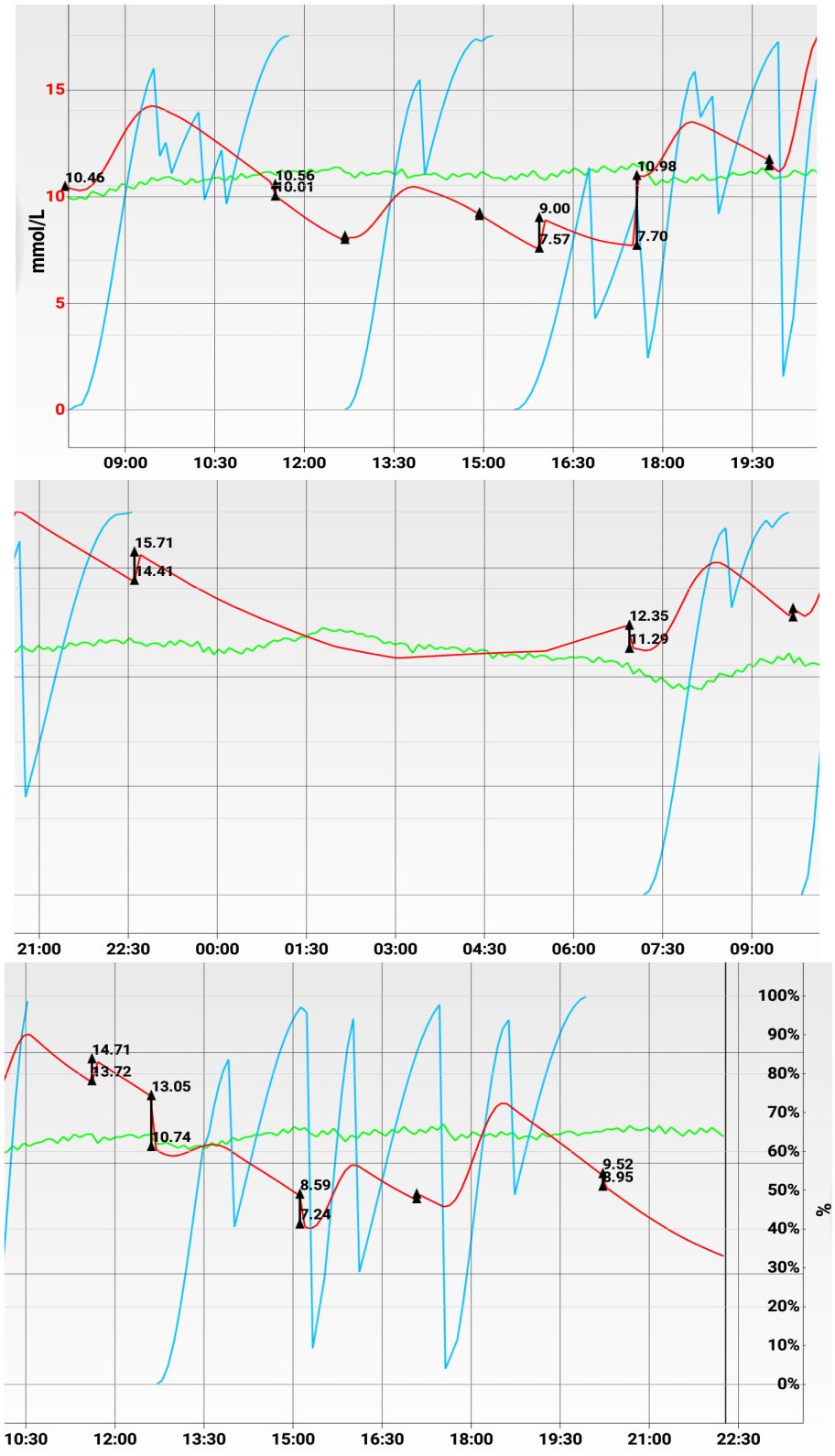


Figure 132: Third kind of testing with test person 2, the last execution (prototype v0.009h).

Total number of blood glucose calibrations: 14.
Number of deviations that are almost 0 (between 0 and 0.3): 5, 36%.
Number of deviations between 0.3 and 0.8: 2, 14%.

Number of deviations between 0.8 and 1.5: 5, 36%.
Number of deviations between 1.5 and 2.5: 1, 7%.
Number of deviations bigger than 2.5: 1, 7%.

As we see, the results are much better now, and the predicted blood glucose levels are now much closer to the real blood glucose levels, than in the results of the first try with the second test person.

7.3. Discussion

A part of discussions was previously presented in section 3.6.

During the project work, the sub-goals presented in section 1.2 were achieved:

1. The methods such as equations, formulas, graphs, to base the software model on, were identified.
2. The engine architecture and the design for the demonstrator UI were created.
3. The prototype that consists of the engine (based on the designed architecture and the set of collected methods) and the demonstrator UI (which is using the engine and demonstrating its features and functionality), was created in the way that it could be used on different platforms, including mobile phone applications, computer applications, web applications, etc.
4. The engine's accuracy was tested, and the testing results were presented.

The project requirements were presented in chapter 4. Are they fulfilled? Table 5 and Table 6 present the status of each functional requirement for the Diabetes Automata engine and demonstrator respectively. The statuses of non-functional requirement for the Diabetes Automata engine and demonstrator are presented in Table 7 and Table 8 respectively.

Req.	*Status description*
1	Fulfilled: the engine runs simulation of blood glucose levels and the blood glucose influence by insulin, carbohydrates and activity, over simulation time
2	Fulfilled: the engine supports various kinds of insulin (Humalog, Novolog, Regular, NPH, Levemir, Lantus), carbohydrates (GI from 0 to 100) and activity (aerobic and anaerobic activity; duration short, medium or long, for aerobic activity); corresponding simulation records can be made by the developed demonstrator during the running simulation
3	Fulfilled: calculations of the simulated blood glucose levels depend on users biometry, the external application must set the biometric values via provided engine API
4	Fulfilled: the engine provides API for external applications
5	Fulfilled: the simulation can be simply started, paused, continued and stopped by the external application that uses the engine, via corresponding API methods.
6	Fulfilled: the engine has "infinite" loop that runs necessary simulation routines and operations with defined frequency
7	Fulfilled: the engine can run simulation on different speeds, the minimal speed is 1

	second per second (real-time)
8	Fulfilled: one of the components of the engine is State class that is responsible for keeping simulation settings and data. State is reset when the simulation is stopped. Settings class is responsible for keeping general simulation parameters and biometric information, and is a part of the engine state. It is not reset when the simulation is stopped.
9	Fulfilled: the engine has an inner routine that resets the engine settings and simulation state in a simple way
10	Fulfilled: the engine is language-independent and operates with language-independent types of values, such as integer, float, Enum, etc.
11	Fulfilled: the current blood glucose level can be changed/corrected whenever the external application sends request for that while the simulation is running
12	Fulfilled: the engine has default values that are used to reset the settings with one request
13	Fulfilled: all API methods have argument, in order to avoid crashes and incorrect behavior of the engine
14	Not fulfilled: Database module supposed to hold the simulation records that are no longer affecting blood glucose levels; the module was not implemented due to the time limitation and the complexity of the project; the priority given to the implementation of this module was low, since the engine can function and be tested without it

Table 5: Status of the functional requirements for Diabetes Automata engine in the last version of prototype (v0.010).

Req.	*Status description*
1	Fulfilled: the demonstrator interacts with the engine via the API provided by the engine
2	Fulfilled: the demonstrator demonstrates the engine's features and functionality
3	Fulfilled: the demonstrator contains minimal set of screens; the necessary screens are Main screen, New Simulation screen, Settings screen; the additional screens that shouldn't have made the demonstrator more complicated, are Default Settings screen and About screen
4	Fulfilled: the Main screen of the demonstrator contains clickable buttons for changing the simulation time speed, making new insulin/carbohydrate/activity records, changing/correcting blood glucose level, launching New Simulation screen if simulation is not running, pause/continue/stop the simulation if it is running
5	Fulfilled: the Main screen displays current simulation time and speed, summary information for the last 24 simulated hours, information about the current blood glucose influence of insulin/carbohydrate/activity, current blood glucose level, graph with the curve representing the levels of blood glucose level over the simulated time (as well as information about insulin, carbohydrates and activity), control buttons to start/stop/pause/continue simulation
6	Fulfilled: while the simulation is running, the demonstrator updates the elements of the Main screen (if the Main screen is currently being displayed); the update frequency is different for different groups of screen elements, and depends on simulation speed, in order to consume less CPU-time
7	Fulfilled: when the simulation is paused, the buttons of "add a new record" type and

	the blood glucose correction button become not clickable
8	Fulfilled: Settings screen contains editable application and engine settings, including biometric parameters
9	Fulfilled: if the demonstrator is launched the first time after installation, or if the settings are corrupted, the demonstrator imports the standard settings from the Engine and resets the values in Settings and Default Settings screens
10	Fulfilled: if the case is opposite to the case in the requirement 9, the demonstrator updates the engine's settings using the values defined in the demonstrator
11	Fulfilled: the demonstrator is able to restore the default settings with one click
12	Fulfilled: the demonstrator has Default Settings screen that contains editable default settings
13	Fulfilled: the demonstrator's graph on the Main screen has changeable zoom and moves automatically along the simulation time
14	Fulfilled: when the simulation is continued from the pause state, the graph is set back to the last point of the simulation time and application-defined zoom
15	Fulfilled: when the running simulation is paused, the graph can be moved and zoomed in/out
16	Fulfilled: the demonstrator allows only correctly formatted data to be set in the editable fields
17	Fulfilled: the demonstrator is able to save the current simulation/engine state and restore it later, using corresponding API methods provided by the engine
18	Fulfilled: the demonstrator shows various dialogs and pop-up messages when necessary

Table 6: Status of the functional requirements for Diabetes Automata demonstrator in the last version of prototype (v0.010).

Req.	*Status description*
1	Fulfilled: the engine is implemented as a project and also exported as a library that can be used by external applications
2	Fulfilled: the engine has modular architecture and runs some of its modules in parallel
3	According to the author, his supervisor and project testers, the engine works stably, responsively and reliably
4	Fulfilled: the engine doesn't provide access to the methods, mechanisms, variables and constants that should not be accessed from outside the Engine; the only operations that external applications can do are to create a new Engine object, to create a new Settings object, and to control the engine via API
5	According to the author, his supervisor and project testers, the engine does not consume too much CPU-time; however, the CPU usage is relatively high due to the routines and calculations required for the simulation

Table 7: Status of the non-functional requirements for Diabetes Automata engine in the last version of prototype (v0.010).

Req.	*Status description*
1	According to the author, his supervisor and project testers, the demonstrator works stably and reliably; also, no problems with integration with Diabetesdagboka were seen, as far as the requirements for the integration (see section 6.6.1) are fulfilled
2	This requirement conflicted with requirement 4. The requirement 4 was fulfilled, which cause problems with fulfilling the requirement 2 – according to the feedback from both test persons, the main screen was too technical and complex. In addition, they mentioned that it is not easy to understand how to use it. This caused the author to additionally write different types of instructions and manuals, some of which are presented in "Appendix A: Instructions". The re-implementation of the main screen of the demonstrator, as well as the implementation of a simpler version of the demonstrator, were considered but not done due to the time limitations
3	According to the author, his supervisor and project testers, the demonstrator usually does not consume too much CPU time; however, the CPU usage is relatively high due to the engine routines and calculations required for the simulation; moreover, frequent update of the Main screen of the demonstrator consumes relatively much CPU time (the screen update procedures were optimized during the development of the prototype, however, further optimization should be considered); in the feedback from the second test person, she mentioned that the battery consumption was acceptable in most of the cases, and quite high sometimes
4	The demonstrator is informative, since it displays detailed information from Insulin, Carbohydrate and Activity modules of the engine, current blood glucose level, summary for the last 24 simulated hours, simulation duration and speed, and graph that has changeable scale and graphically displays information about blood glucose levels and the action of insulin/carbohydrate/activity over simulated time.

Table 8: Status of the functional requirements for Diabetes Automata demonstrator in the last version of prototype (v0.010).

The overview of the last results of the testing type 3 with both test persons (see Figure 130 for the first test person and Figure 132 for the second test person) is presented in Table 9 and shows the current accuracy of the last version of the prototype.

Test person / *Testing*	*Test person 1*		*Test person 2*	
Deviations between 0 and 0.3	10	33.(3)%	5	36%
Deviations between 0.3 and 0.8	8	27.(6)%	2	14%
Deviations between 0.8 and 1.5	5	16.(6)%	5	36%
Deviations between 1.5 and 2.5	2	6.(6)%	1	7%
Deviations bigger than 2.5	5	16.(6)%	1	7%
Sample database timeframe	3.5 days		1.5 days	
Total number of blood glucose corrections	30		14	

Table 9: The last results of testing type 3 with both test persons.

If we consider only blood glucose and insulin constituents of the Diabetes Automata engine, the engine simulates the fall in blood glucose over time caused by an insulin injection, and the magnitude of the fall depends on the type of insulin and the number of injected insulin units. In this case, the general idea of the engine is quite close to insulin calculators, which are software tools aimed to calculate the insulin dose required to keep the person's blood glucose levels in acceptable range. But how reliable are these tools? According to the systematic assessment of 46 existing mobile-based insulin dose calculators (Huckvale et al., 2015), many of them are not reliable and have serious issues. Table 10 presents some of them, as well as the status of the presence of the issues in the prototype v0.010 (in most of the cases this status was explained above in this section, in Table 5 – Table 8).

Issue	*% of apps*	*Does the prototype have this issue?*
Inputs unconstrained by physiologically or logically plausible limits, for example, negative values accepted	91%	No
No validation of any input, for example, textual values allowed in all numeric fields	52%	No
Calculation despite missing inputs	59%	No
User data cannot be entered faithfully or are not stored correctly for calculation	13%	No
Precision issues, for example, a data field for measurements in mmol/L that accepts only whole integers.	9%	When registering new information on the main screen of the prototype, blood glucose value correction can be done with step 0.1 mmol/L, insulin can be set with step 0.5 IU, carbohydrate can be set with step 1 gram; it was considered to be acceptable since the testers didn't report problems with such implementation
Data not stored correctly, data change unexpectedly, for example, transposed into a different data field	4%	No
Calculation does not conform to principles of underlying clinical conceptual model, for example, by assuming an omitted blood glucose implies a measured value of zero	48%	The engine as well as the demonstrator have simulation parameter that doesn't allow blood glucose level to be below 2 mmol/L, it can be turned off
Calculation does not proceed in accordance with the stated formula	14%	No
Calculator output is not consistently synchronized with changing user inputs	37%	No
A calculator which normally refreshes output automatically in response to changing inputs does not under certain circumstances	7%	Have not been reported
Software unexpectedly crashes or becomes unresponsive during normal use	24%	Have not been reported

Table 10: Issues of mobile-based insulin dosage calculator apps and their relation to the prototype v0.010. Based on: (Huckvale et al., 2015, Table 1)

The presented issues are serious, since they can affect the reliability, accuracy and usability.

In addition, David Klonoff states in his article (Klonoff, 2012) that 6% of the dose determinations during the testing of insulin calculators was incorrect.

It is worth noting that a friend of the second test person had a dangerous situation caused by the incorrect insulin dose calculated by an insulin calculator. The friend of the second test person gave the calculator a try and trusted the calculated dose. The result of this was a severe hypoglycemia. The hypoglycemia was managed, however, the impact on the blood glucose levels was lasting until the next day after the incident.

What results can be considered as correct (or acceptable) or not correct (or not acceptable)? The author found the standard DIN ISO 15197:2003 by the International Organization for Standardization (ISO), which is a standard for blood glucose meters (McCarren, 2015a; McCarren, 2015b). In Europe, this standard is met by the blood glucose meters with a Conformité Européenne label (Schnell et al., 2013).

Diabetes Automata is not a blood glucose meter, however, the standard could be used as an example of the results evaluation.

According to the standard, in case if blood glucose is lower than 75 mg/dl (~4.1625 mmol/L), 95% of the results must be within ±15%; in case if blood glucose is 75 mg/dl or higher, 95% of the results must be within ±20% (Schnell et al., 2013; McCarren, 2015a; McCarren, 2015b).

If we count the percentage of deviations that are within ±15% for blood glucose levels < 4.1625 mmol/L and ±20% for blood glucose levels >= 4.1625 mmol/L in the results presented in Figure 130 and Figure 132, we get the results presented in Table 11.

Test	*Results within ±15% for blood glucose < 4.1625 mmol/L and within ±20% for blood glucose levels >= 4.1625 mmol/L*		*Total number of blood glucose corrections*
	Number	*%*	
Last testing type 3 with test person 1 (Figure 130)	22	73.(3)	30
Last testing type 3 with test person 2 (Figure 132)	12	85.7	14

Table 11: Example evaluation of the last results of testing type 3 with both test persons using the standard DIN ISO 15197:2003.

As we see in Table 11, the accuracy of the last test results didn't reach 95%, like it is required for blood glucose meters. However, the results were considered to be good. It is worth noting that due to the time limit and the experimentalism and the complexity of the project, the author's expectations about the results were worse than the results presented in Table 9 and Table 11.

There is a wide field of future work in this project, both in implementation and testing. The potential future works are discussed in chapter 8. According to feedback from both test persons, the UI should be re-implemented in the way to make is more user-friendly and more easy to use and less technical. The suggestion to implement two versions of the demonstrator was considered. One version supposed to be the version presented in this thesis, another version supposed to be much more simplified and contain less technical information. The second version was not implemented due to the time limitations.

The activity module of the engine should be re-implemented in order to better reflect the effect of various types of activity on blood glucose levels. Also, the engine should support the input of historical data (not only the data for the current moment in simulated time), as well as the ability to change previously made simulation records. The full overview of future work can be found in chapter 8.

Due to the time limitation and other limitation described in section 1.3, only few people were involved in testing of the prototype. Additional reasons were that not many potential testers had time for testing, and not many people wanted to provide personal data, such as their diabetes self-management history from Diabetesdagboka.

In order to make testing more comprehensive, more people with diabetes of different gender and age should be involved in testing type 3 defined earlier in section 7.1. Among these people should be both people using long-acting insulin and people using insulin pump.

Testing type 2 defined earlier in section 7.1 is not accurate enough. If a test person runs simulator in simulation mode 3, all eaten carbohydrates are set as GI Medium, and all activities are set as aerobic activities, since Diabetesdagboka doesn't provide such information in its records. These causes the testing to be not accurate. If a test person runs simulator in simulation mode 1, the probability of better accuracy is higher, since the person sets all records in the simulator manually, and can choose between different types of insulin, carbohydrates and activity. However, test persons don't know the exact and even approximate (low, medium, high, 100) GI of food they have eaten, which means that the person can choose a wrong carbohydrate GI type and get bigger deviations than if the GI was set correct. These are the reasons why more time and attention was given to testing type 3.

During the testing type 3 with the second test person, an additional constant was applied to insulin in order to get more accurate results. The reason and the explanation for this is as follows.

The second test person was using insulin pump instead of long-acting insulin injections, whereas the prototype is directed to usual insulin injections. When the testing type 3 with this test person was done for the first time (Figure 131), the author has noticed that the reason for inaccurate results was too high insulin sensitivity for this person. There were two assumptions for that – either the biometric personalization of blood glucose simulation doesn't work correctly, or the usage of insulin pump is the reason for incorrectly calculated insulin sensitivity. In order to find out which of these two cases took place, the author had a meeting with professor Georg Sager (Department of Medical Biology, UiT). During the discussion with the professor, the reason for incorrect insulin sensitivity was found: the usage of insulin pump affects insulin sensitivity, and the magnitude of the effect differs from person to person. The confirmation of the fact that insulin pump affects insulin sensitivity was also found in literature (Donga et al., 2013). In order to find the way to calculate the effect, a new research had to be done, which was not possible due to the time limitation, so the application of a constant was suggested. The applied constant appeared to be 0.6 (which means that after multiplication with results from insulin module, the results are reduced by 40%) and was found experimentally while running the testing type 3 with the database from the second test

person several times. When the constant was found and applied, the testing results became much more accurate (Figure 132).

Making research about differences between the behavior of pumped insulin and classically-injected insulin, and how the usage of insulin pump affects insulin sensitivity, is one of the next steps of the project further works (all further works are presented in chapter 8).

7.4. Summary

In the beginning of this chapter, the testing procedures were presented and described. Among the three defined types of testing are constant feedback that the author has been receiving during the prototype development, simulation trial by test persons (during which the testers run blood glucose simulation during several days and give feedback about the usability, reliability and accuracy), and engine accuracy testing made by the author using the sample Diabetesdagboka databases (which contain personal diabetes self-management data from the testers) as well as the additional information that cannot be registered in Diabetesadagboka (such as GI of meals).

After, the testing results were presented. Among the presented information are the resulting improvements of the demonstrator; the results of the second kind of testing that was used to improve the engine; the first execution of the accuracy testing, the engine calibrations caused by its results, and the final execution of the accuracy testing. In the end of each testing results presentation, the results were grouped by the deviation magnitude.

In the end, we have discussed about the testing and its results, as well as other aspects of the project that were worth being mentioned, discussed or explained.

8. Further Works

Due to the complexity of the project, the limitations discussed in section 1.3, and the feedback from test persons, the project has several further works that should be done in the future.
First of all, Diabetes Automata should be comprehensively tested with people of different gender, age and body type, in order to:

- Test the biometric parametrization of simulation and the customization/adaptation of the simulation results to different people;
- Improve/calibrate the implemented algorithms;
- Consider further algorithm changes and improvements.

Also, there are many potential improvements of the implemented prototype, as well as new functionality and features that can be implemented. Some examples of them are presented below in sections 8.1 and 8.2.

8.1. Possible Improvements of the Implemented Solution

- Engine: make a deeper research about the effect of activity and re-implement Activity module;
- Engine: calibrate algorithms in Insulin, Carbohydrate, Activity, and Blood Glucose modules, in order to get better and more accurate results.
- Engine: make a better implementation of dawn phenomenon, preferably the one that would depend on the biometric information.
- Android Demonstrator: improve the IU, make it more easy to use and user-friendly.
- Android Demonstrator: improve the integration with Diabetesdagboka.

8.2. New Algorithms, Features, Functionality

- Engine: implement insulin pump simulation with the opportunity to set the plan for the micro-injections, which contains the time periods and the amount of insulin that is injected during these periods. Before that – make research and implement the algorithm for calculation of the coefficient that must be applied to insulin sensitivity for people with diabetes type 1 that use insulin pump. The calculation shall depend on user's biometric information.
- Engine: implement simulation of hypoglycemia unawareness.
- Engine: implement "rolling-back-in-time" feature.
  To make it possible to change what has happened during simulation and see the results.
  For example, to change the amount/type of injected insulin or consumed carbohydrates some time before, in order to see how the new blood glucose level prediction would differ. Or just to correct/delete a previously made record if it was not correct.
- Engine: implement automatic engine calibration.

It would be implemented as parsing of some amount of sample data (for example from Diabetesdagboka database) from an external application to the engine that would analyze it and set estimated time-depending multipliers to the results of Insulin/Carbohydrate/Activity modules, in order to make the results more accurate and more personalized.

- Engine: implement Database module.
- Engine: implement Diabetes Automata engine for diabetes type 2.
- Develop a simple diabetes-related example-game.
- Engine: implement engine algorithms based on an existing mathematical model (see 2.2).
- Android Demonstrator: implement a simpler version of demonstrator.
- Android Demonstrator: implement compression of simulation save files.
- Android Demonstrator: implement export of an entire simulation graph as a picture in PNG/JPG format.
- Android Demonstrator: implement the translation of the application texts to other languages.
- Android Demonstrator: implement sending of crash-reports to the author's email.

9. Concluding Remarks

During the project research, a prototype consisting of the Diabetes Automata software engine and the demonstrator, was designed, implemented, tested and discussed. The prototype meets the idea and the goals of the project. However, there are several future works to be done in order to improve the engine's accuracy and functionality, and the usability of the demonstrator. Some of the further works presented in chapter 8 were planned during the development but not implemented due to the time limitation, another works were considered by the author or suggested by testers.

The author was surprised about the results presented in Table 9 and Table 11, since, as mentioned in section 7.3, his expectations about the results were worse than the achieved results due to the experimentalism and the complexity of the project.

Diabetes Automata is a research project that is aimed to present a software solution, a software module that could be used in a wide range of software for people with diabetes and specialists. According to the literature review presented in chapter 2, the project is the first attempt to develop a software engine that is able to simulate blood glucose behavior and calculate blood glucose levels in people with diabetes type 1 with help of input information such as user's biometry and information about insulin, carbohydrates and physical activity, and could be used in external diabetes-related serious games, systems and applications, which are aimed to calculate, mimic or continuously simulate blood glucose levels over time depending on input data and various parameters.

The software engine has great potential to be the blood glucose simulation heart of various diabetes-related systems and applications on different platforms, among which are serious games aimed to improve skills and learn about the disease, mobile and web-based applications (among which can be self-management tools and serious games), decision support systems, and other systems and applications.

Appendix A: Instructions

Detailed version (manual):

- Main Screen:

  - When simulation not running:
    - Press "New Simulation" to open "New Simulation" screen.
    - Press on the graph to change graph mode:
      - Mode 1: Blood Glucose is represented in mmol/L, Insulin and Carbohydrate are represented in % word done, Activity is represented in effect strength.
      - Mode 2: Blood Glucose is represented in mmol/L, Insulin and Activity are represented in effect strength, Carbohydrate is represented in % word done.
      - Mode 3: Blood Glucose is represented in mmol/L.
    - Long press on the graph to change the graph size.
    - Press "..." options menu to see options:
      - Press "Settings" to open "Settings" screen.
      - Press "Load Simulation" to see "Load Saved Simulation" dialog:
        - Press "Cancel" to cancel the dialog.
        - Select one of the saved simulations in the list, the dialog will re-appear:
          - Press "Cancel" to cancel the dialog.
          - Press "Delete" to delete the saved simulation from disk.
        - Press "Load" to load the selected simulation. It can take some time for the simulation to get loaded. While loading, "Loading" dialog will be on the screen.
      - Press "Enable/Disable BG < 2.0 mmol/L" to change the state of this simulation parameter.
      - Press "Enable/Disable Glucose Consumption By Brain" to change the state of this simulation parameter.
      - Press "Enable/Disable Dawn Phenomenon" to change the state of this simulation parameter.
      - Press "Exit" to exit. All application processes will be stopped, including the background service.

  - When simulation mode 1 is running:
    - Press "Pause" to pause the simulation.
    - Press "Stop" to stop the simulation:
      - Press "Cancel" to cancel the dialog.
      - Press "Don't Save" to stop the simulation without saving.
      - Press "Save" to save and stop the simulation.
        - All saved simulation are placed in "<internal storage>/DiabetesAutomata/".
    - Press on the line with "Duration" and "Real Time" or "x Min / 1 Sec" or "x Sec / 1 Sec" to change the simulation time speed.
      - Between 1 Sec / 1 Sec (real-time) to 99 Min / 1 Sec.
    - Press on the "Insulin" square button to make a new insulin simulation record.
      - Choose type of insulin.
      - Set number of IU.
    - Press on the "Carbo" square button to make a new carbohydrate simulation record.
      - Choose type of GI.

- Set number of grams.
- Press on the "Activity" square button to make a new activity simulation record.
  - Choose type of activity.
  - For Aerobic Activity, choose the duration of activity.
- Press on "Blood Glucose" to calibrate (manually set new value of) the current blood glucose level.
- Press on the graph to change graph mode:
  - Mode 1: Blood Glucose is represented in mmol/L, Insulin and Carbohydrate are represented in % word done, Activity is represented in effect strength.
  - Mode 2: Blood Glucose is represented in mmol/L, Insulin and Activity are represented in effect strength, Carbohydrate is represented in % word done.
  - Mode 3: Blood Glucose is represented in mmol/L.
- Long press on the graph to change the graph size.
- Press "..." options menu to see options:
  - Press "Settings" to open "Settings" screen.
  - Press "Save Simulation" to save the current simulation as simulation mode 1.
    - All saved simulation are placed in "<internal storage>/DiabetesAutomata/".
  - Press "Full Screen Graph" to switch to the fullscreen mode.
  - Press "Enable/Disable BG < 2.0 mmol/L" to change the state of this simulation parameter.
  - Press "Enable/Disable Glucose Consumption By Brain" to change the state of this simulation parameter.
  - Press "Enable/Disable Dawn Phenomenon" to change the state of this simulation parameter.
  - Press "Exit" to exit. All application processes will be stopped, including the background service.

❖ When simulation mode 2 is running:
- Press "Pause" to pause the simulation.
- Press "Stop" to stop the simulation:
  - Press "Cancel" to cancel the dialog.
  - Press "Don't Save" to stop the simulation without saving.
  - Press "Save" to save and stop the simulation.
    - All saved simulation are placed in "<internal storage>/DiabetesAutomata/".
- Press on the line with "Duration" and "Real Time" or "x Min / 1 Sec" or "x Sec / 1 Sec" to change the simulation time speed.
  - Between 1 Sec / 1 Sec (real-time) to 99 Min / 1 Sec.
- Press on the graph to change graph mode:
  - Mode 1: Blood Glucose is represented in mmol/L, Insulin and Carbohydrate are represented in % word done, Activity is represented in effect strength.
  - Mode 2: Blood Glucose is represented in mmol/L, Insulin and Activity are represented in effect strength, Carbohydrate is represented in % word done.
  - Mode 3: Blood Glucose is represented in mmol/L.
- Long press on the graph to change the graph size.
- Press "..." options menu to see options:
  - Press "Settings" to open "Settings" screen.
  - Press "Save Simulation" to save the current simulation as simulation mode 1.
    - All saved simulation are placed in "<internal storage>/DiabetesAutomata/".
  - Press "Full Screen Graph" to switch to the fullscreen mode.
  - Press "Enable/Disable BG < 2.0 mmol/L" to change the state of this simulation parameter.

- Press "Enable/Disable Glucose Consumption By Brain" to change the state of this simulation parameter.
- Press "Enable/Disable Dawn Phenomenon" to change the state of this simulation parameter.
- Press "Exit" to exit. All application processes will be stopped, including the background service.

❖ When simulation mode 3 is running:
- o Press "Stop" to stop the simulation:
  - Press "Cancel" to cancel the dialog.
  - Press "Don't Save" to stop the simulation without saving.
  - Press "Save" to save and stop the simulation.
    - All saved simulation are placed in "<internal storage>/DiabetesAutomata/".
- o Press on the "Insulin" square button to make a new insulin simulation record.
  - Choose type of insulin.
  - Set number of IU.
- o Press on the "Carbo" square button to make a new carbohydrate simulation record.
  - Choose type of GI.
  - Set number of grams.
- o Press on the "Activity" square button to make a new activity simulation record.
  - Choose type of activity.
  - For Aerobic Activity, choose the duration of activity.
- o Press on "Blood Glucose" to calibrate (manually set new value of) the current blood glucose level.
- o Press on the graph to change graph mode:
  - ➢ Mode 1: Blood Glucose is represented in mmol/L, Insulin and Carbohydrate are represented in % word done, Activity is represented in effect strength.
  - ➢ Mode 2: Blood Glucose is represented in mmol/L, Insulin and Activity are represented in effect strength, Carbohydrate is represented in % word done.
  - ➢ Mode 3: Blood Glucose is represented in mmol/L.
- o Long press on the graph to change the graph size.
- o Press "..." options menu to see options:
  - Press "Settings" to open "Settings" screen.
  - Press "Save Simulation" to save the current simulation as simulation mode 1.
    - All saved simulation are placed in "<internal storage>/DiabetesAutomata/".
  - Press "Full Screen Graph" to switch to the fullscreen mode.
  - Press "Enable/Disable BG < 2.0 mmol/L" to change the state of this simulation parameter.
  - Press "Enable/Disable Glucose Consumption By Brain" to change the state of this simulation parameter.
  - Press "Enable/Disable Dawn Phenomenon" to change the state of this simulation parameter.
  - Press "Exit" to exit. All application processes will be stopped, including the background service.

❖ When simulation is paused:
- o Press "Continue" to continue the simulation.
- o Press "Stop" to stop the simulation:
  - Press "Cancel" to cancel the dialog.
  - Press "Don't Save" to stop the simulation without saving.
  - Press "Save" to save and stop the simulation.

- All saved simulation are placed in "<internal storage>/DiabetesAutomata/".

- (Except from Simulation Mode 3) Press on the line with "Duration" and "Real Time" or "x Min / 1 Sec" or "x Sec / 1 Sec" to change the simulation time speed.
  - Between 1 Sec / 1 Sec (real-time) to 99 Min / 1 Sec.
- Pan the graph to scroll the graph.
- Zoom the graph with 2 fingers to change the graph's scale.
- Press "..." options menu to see options:
  - Press "Settings" to open "Settings" screen.
  - Press "Save Simulation" to save the current simulation as simulation mode 1.
    - All saved simulation are placed in "<internal storage>/DiabetesAutomata/".
  - Press "Full Screen Graph" to switch to the fullscreen mode.
  - Press "Enable/Disable BG < 2.0 mmol/L" to change the state of this simulation parameter.
  - Press "Enable/Disable Glucose Consumption By Brain" to change the state of this simulation parameter.
  - Press "Enable/Disable Dawn Phenomenon" to change the state of this simulation parameter.
  - Press "Exit" to exit. All application processes will be stopped, including the background service.

- New Simulation Screen:
  - Press "Mode 1" to start a new simulation in a usual mode.
  - Press "Mode 2" to start a new simulation of the Diabetesdagboka records for the last 30 days.
    Requirements:
    - Diabetesdagboka V1.5.3 (ee7a27fb) with IPC.
    - Records in the Diabetesdagboka database.
  - Press "Mode 3" to start a new real-time simulation paired with Diabetesdagboka.
    Requirements:
    - Diabetesdagboka V1.5.3 (ee7a27fb) with IPC.
  - Press "Starting Blood Glucose Level" to set the starting blood glucose level.
  - Press "Starting Time" to set the starting time of the day.
  - Press "Allow Dawn Phenomenon" to change the state of this simulation parameter.
  - Press "Allow Glucose Consumption By Brain" to change the state of this simulation parameter.
  - Press "Allow BG < 2.0 mmol/L" to change the state of this simulation parameter.
  - Press "..." options menu to see options:
    - Press "Settings" to open "Settings" screen.

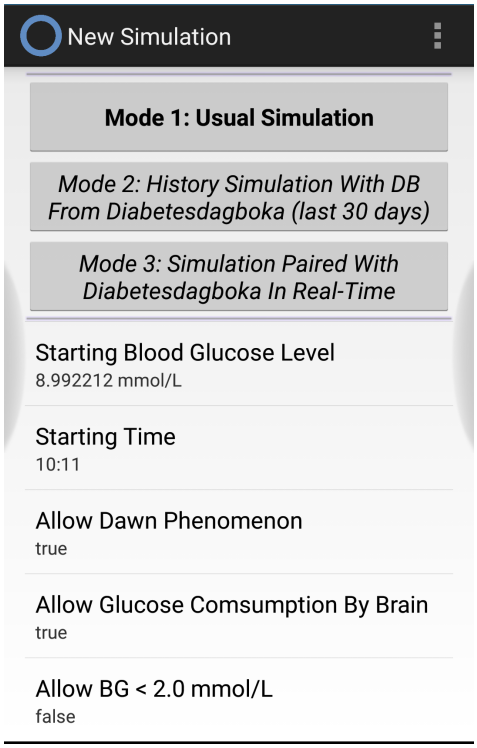
- Settings Screen:
  - Change the settings according to user's personal parameters and needs.
  - Press "Default Settings" to reset the values.
  - Press "..." options menu to see options:
    - Press "Change Default Values" to open "Default Settings" screen.
    - Press "About" to open "About" screen.

- Default Settings Screen:
  - Change the settings according to user's personal parameters and needs.
  - Press "..." options menu to see options:
    - Press "About" to open "About" screen.

Short version with figures:

Settings:

Change the simulator's settings according to user's biometrical information and personal needs.



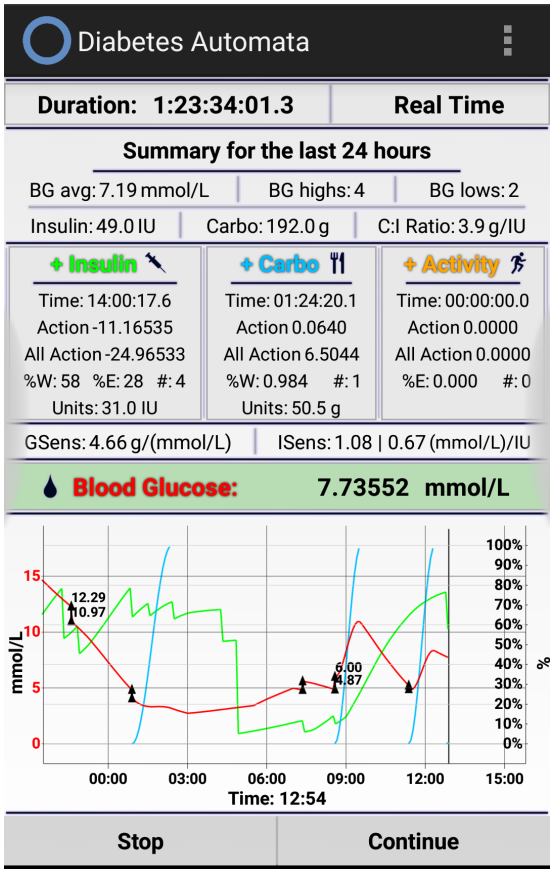
New Simulation:

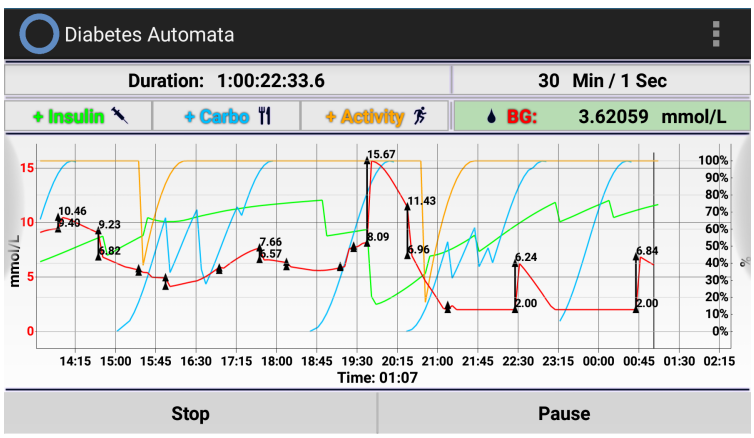
Press "Mode 1" to start a new simulation in a usual mode, where user makes new insulin/carbohydrate/activity records and blood glucose corrections manually.
Press "Mode 2" to start a new simulation of the Diabetesdagboka records for the last 30 days.
Press "Mode 3" to start a new real-time simulation paired with Diabetesdagboka – all new Diabetesdagboka records will be automatically set into the running simulation. Requirement for "Mode 2" and "Mode 3" is Diabetesdagboka V1.5.3 (ee7a27fb) with IPC.
Set current blood glucose level in the "Starting BG Level" setting.

Main Screen:

Use Start/Stop and Pause/Continue buttons for simulation control.
Simulation time speed can be changed by pressing on the line with "Duration" (except from Simulation Mode 3).

New insulin/carbohydrate/activity record can be made by pressing on corresponding square buttons (except from Simulation Mode 2).
Blood glucose can be manually calibrated by pressing "Blood Glucose" (except from Simulation Mode 2).

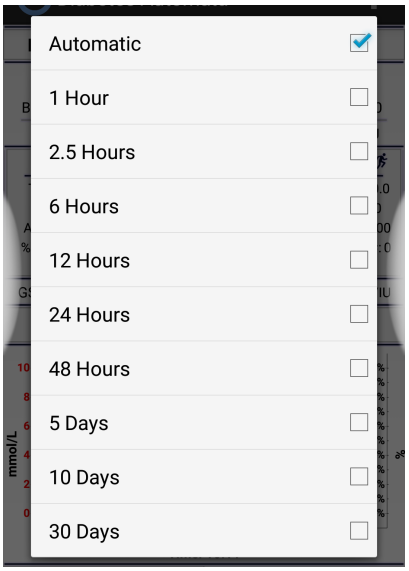




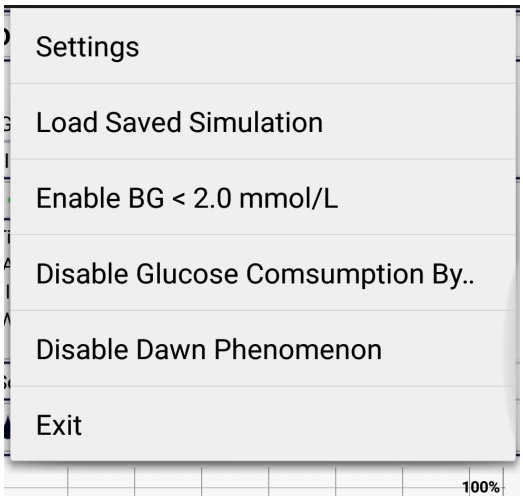
Graph mode can be changed by a single short click on the graph.

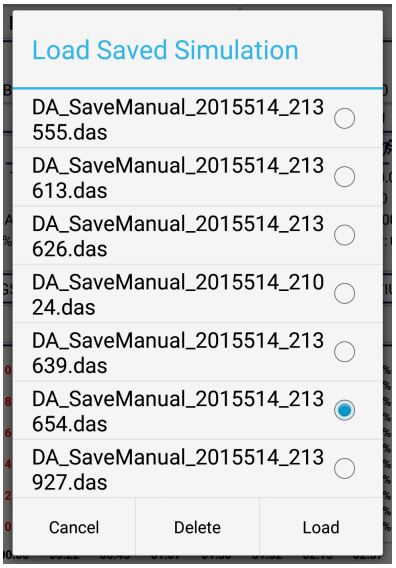
Graph Mode 1: Blood Glucose is represented in mmol/L, Insulin and Carbohydrate are represented in % work done, Activity is represented in % effect strength.
Graph Mode 2: Blood Glucose is represented in mmol/L, Insulin and Activity are represented in % effect strength, Carbohydrate is represented in % work done.
Graph Mode 3: Blood Glucose is represented in mmol/L.

Graph Scale can be changed by long click on the graph.

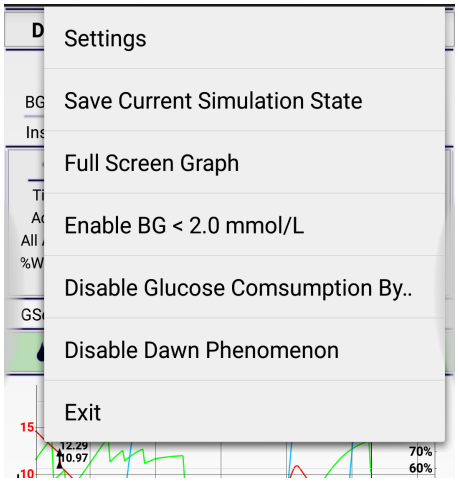


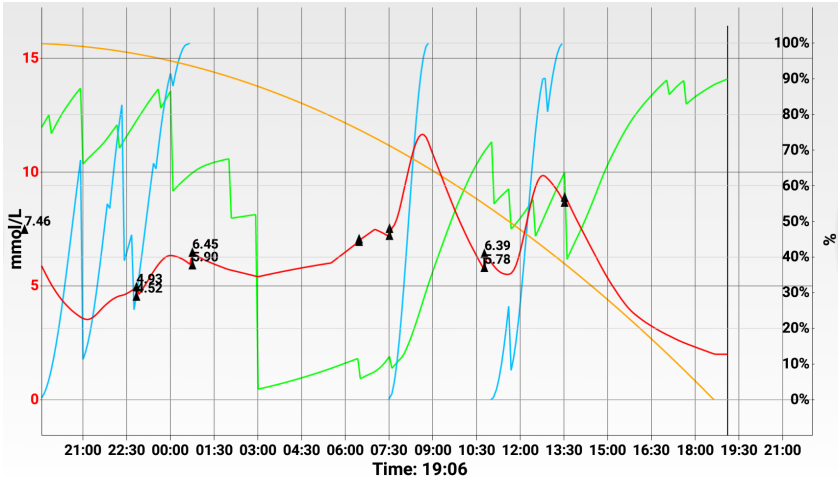
When simulation is not started, user can load or delete saved simulations from the options menu.



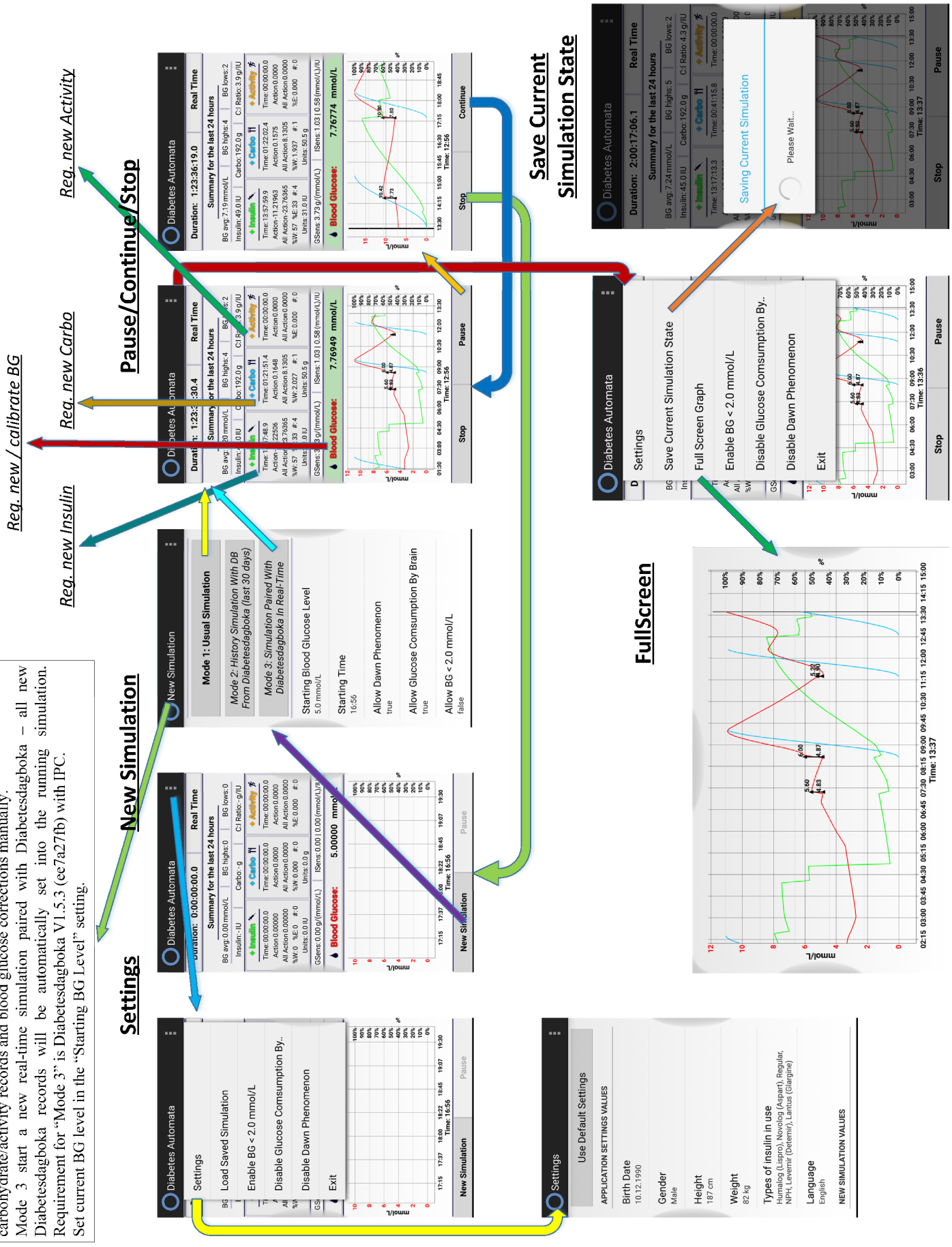


When simulation is started, user can save current simulation (in any mode) as a Simulation Mode 1 from the options menu, as well as switching to the fullscreen mode by pressing “Full Screen Graph”.





Quick Guide For Testers:



Mode 1: start a new simulation in a usual mode, where user makes new insulin/ carbohydrate/activity records and blood glucose corrections manually.
Mode 3 start a new real-time simulation paired with Diabetesdagboka – all new Diabetesdagboka records will be automatically set into the running simulation. Requirement for "Mode 3" is Diabetesdagboka V1.5.3 (ec7a27fb) with IPC.
Set current BG level in the "Starting BG Level" setting.

Appendix B: Article For Conference SHI 2015 (15-17 June)

Diabetes Automata: Software Engine for Blood Glucose Level Simulation

Aleksandr Agafonov[a], Eirik Årsand[a,b], Alexandra Makhlysheva[b], Håvard Blixgård[b], Meghan Bradway[b], Gunnar Hartvigsen[a,b]

[a] *Department of Computer Science, University of Tromsø - The Arctic University of Norway, Norway*
[b] *Norwegian Centre for Integrated Care and Telemedicine, University Hospital of North Norway*

Abstract

For individuals with diabetes to live a healthy life, they must balance carbohydrate intake, insulin injections, physical activity, and monitoring of blood glucose levels. For adolescents with diabetes, this can be challenging – measuring blood glucose and injecting insulin often feels awkward, especially in public areas. In the long run, the result of not managing the factors affecting their diabetes can cause serious consequences. In order to motivate this particular group with diabetes to maintain recommended levels of blood glucose, serious games can be used. The Diabetes Automata project is a part of the research being done on serious games for diabetes at the Norwegian Centre for Integrated Care and Telemedicine. In the project, we have developed a prototype version of a software engine on which diabetes-related serious games, simulators and other tools for diabetes management on various platforms can be based. The Diabetes Automata calculates blood glucose levels based upon relevant patient-gathered data such as insulin, carbohydrates, physical activity, and the user's biometry. The prototype is not yet evaluated.

Keywords:

Blood Glucose Simulation, Diabetes, Mobile Application, Software engine

Introduction

Diabetes mellitus is a group of metabolic diseases characterized by high blood glucose (BG) levels resulting from defects in insulin secretion, its action, or both. It is a chronic disease that occurs either when the pancreas does not produce enough insulin, or when the body cannot effectively use the insulin it produces. WHO estimates that 347 million people worldwide have diabetes [1]. Type 1 diabetes (T1D) is characterized by absolute deficiency of insulin production, daily administration of insulin, and occurs in about 10% of the cases [1]. The cause of type 1 diabetes is not known and it is not preventable, according to current knowledge about the disease [1].

Good management of one's BG levels makes living with diabetes much easier. Most people with T1D use a blood glucose meter for BG monitoring and insulin pen or pump for insulin injections. Through frequent measurements and injections of different types of insulin, people with diabetes can generally maintain a BG level within the recommended range. For adolescents with T1D, managing this important element can be a challenge. Measuring BG and taking insulin can be perceived by adolescents as stigmatising activities. Sometimes patients forget to administer insulin injections or underestimate the effect of the amount of carbohydrates they have eaten.

However, neglecting the disease leads to serious long-term consequences. Therefore, adolescents must be made aware of the long-term consequences of poor blood glucose management and be motivated to adequately manage their diabetes– for example, using their own media channels. One such channel for achieving this is serious games.

Diabetes Automata is a master's project under development that will be finished during the summer 2015. In this project, we have developed a software engine prototype that can be used in diabetes-related serious games, simulators and other tools on various platforms. The engine calculates and provides the blood glucose level based upon relevant patient-gathered background data such as insulin, carbohydrates, physical activity, along with the user's biometry such as age, gender, height and weight.

Materials and Methods

The design of the Diabetes Automata has been divided into two parts: the engine itself, and a demonstrator application (app) that is used to test the engine's functionality and illustrate its potential use. The engine is developed in Java to make it possible for external applications to use it through the engine's API (Application Programming Interface).

The Diabetes Automata offers the opportunity to run blood glucose simulation at various speeds that can be set through the demonstrator application's user interface: from 1 second per second (real-time) to 99 minutes per second. The demonstrator app is an Android-based simulator with a simple design, which displays various types of output information in a form of text, numbers, and graphs. The prototype can run a customized simulation with the opportunity to save the current state of the simulation and then load and continue it later. In addition, simulations can be based on records from the Diabetes Diary, a mobile self-management application for people with diabetes developed by the Norwegian Centre for Integrated Care and Telemedicine (NST) [2-4]. See Figure 8.

The Diabetes Automata has a modular architecture, which provides certain advantages. For example, it is easier for the engine developers to add, change, and edit code to play with different formulas and equations, and to run the modules in parallel. The architecture is shown in Figure 1.

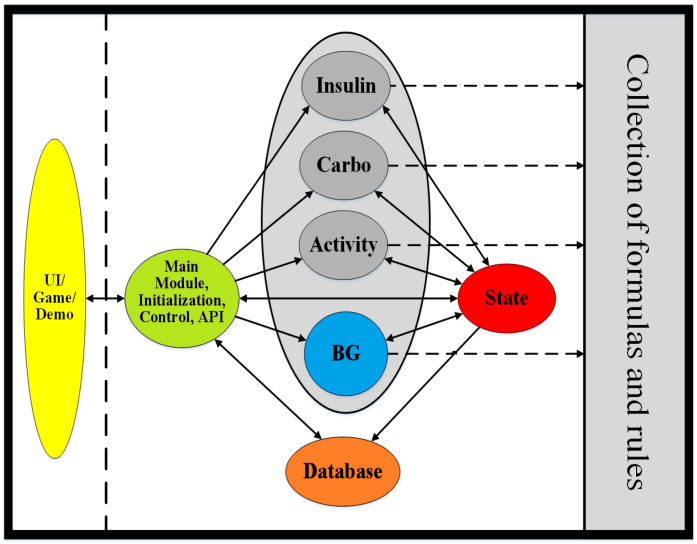


Figure 1. Diabetes Automata Engine Architecture

The engine contains several modules that are described below.

Carbohydrates Module

First, the engine calculates the user's glucose sensitivity, which is the number of grams of glucose (or carbohydrate with glycaemic index (GI) = 100) that increase the blood glucose level by 1 mmol/L. The calculation of Glucose Sensitivity is based upon Table 1 and the set of functions represented in Figure 2.

Table 1. BG rise by 1 g glucose depending on body weight. (Source [5])

Body weight (kg)	1 gram glucose will raise BG by (mmol/L):
16	1.11
32	0.56
48	0.39
64	0.28
80	0.22
95	0.18
111	0.17
128	0.14
143	0.12

Glucose Sensitivity (g/mmol) is calculated as follows (see Figure 2). The engine calculates the number of mmol increased by 1g of glucose (y) for the user's weight (x), and then divides 1 by the result.

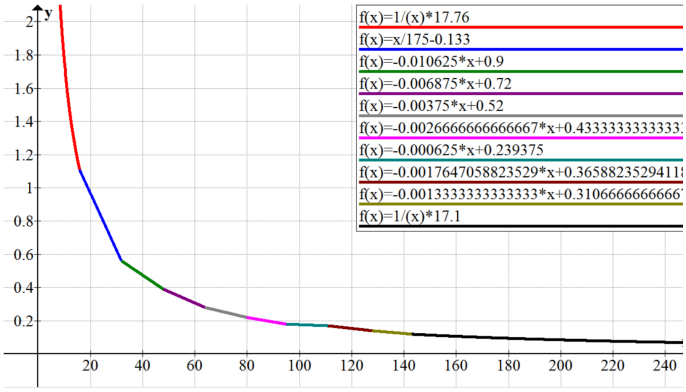


Figure 2. Glucose Sensitivity in mmol/L per gram of glucose, where x is weight and y is a number of mmol increased by 1 gram of glucose. Figure is created by the authors and based on the data from Table 1 (the data is used as (x,y)-coordinates joined to each other).

Further the duration of action is calculated. The duration depends on GI (see Figure 4) of the engine record and is calculated by the function represented in Figure 3.

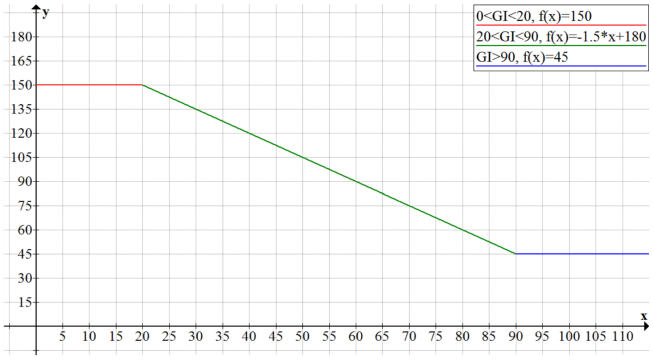


Figure 3. Calculation of the duration of action of a carbohydrate simulation record. X-axis represents GI, Y-axis represents minutes.

Figure 4 shows the difference between the duration and the effect of high GI and low GI, for a non-diabetic person.

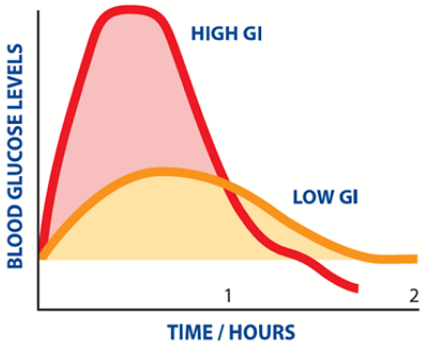


Figure 4. Duration and action of high GI and low GI, generally. (Source [6])

Next step of the algorithm is to calculate the influence of work done for the current moment in time, in percent. The result

depends on the duration, the example curves for 45, 60, 90 and 120 minutes are represented in Figure 5.

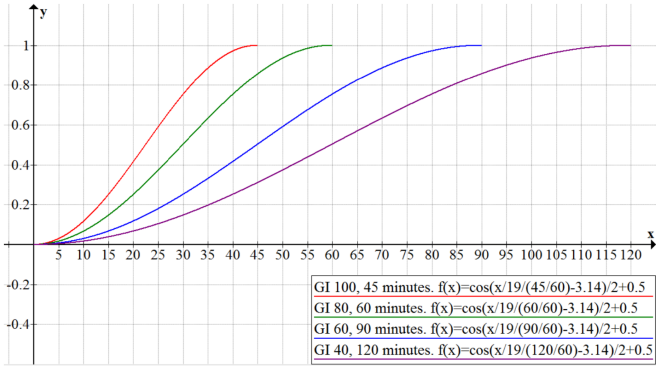


Figure 5. Percentage BG increase from high GI food over time, where x is time in minutes and y is the effect. 1 on Y-axis represents 100%.

The last step done by the Carbohydrates Module is to calculate the rise in blood glucose in the current moment in time. The calculation is as follows.

```
BG Rise (mmol/L) =
Number of carbohydrates in the record (g) *
GI / 100 (%/100) *
Work done for the moment in time (%/100) /
Glucose sensitivity (g per 1 mmol/L)
```

Before the module finishes its execution, it calculates the additional module values, such as the total module current influence on blood glucose level for the current moment in time, the total module influence for the whole activity time, the relation those two values in percent, the number of currently active records, the remaining module action time, and the history of BG influence value for future purposes. All this is stored to the simulation state. If the action time of the current record is over, it is marked as deleted.

Insulin Module

First, durations of action of an insulin simulation record are defined, which varies between different types of insulin.

The next step is to define the peaking value (1) in mg/min/kg for the defined number of IU/kg (2). Authors found curves representing glucose utilization rate for Humalog, Novolog, Regular, NPH, Levemir, Lantus (see Figure 6) from information sources, and took the maximal glucose utilization rate values for each mentioned insulin type. That was 6.5 mg/min/kg for 0.2 IU/kg for Humalog, 6.8 mg/min/kg for 0.2 IU/kg for Novolog, 5.(6) mg/min/kg for 0.2 IU/kg for Regular, 3.4 mg/min/kg for 0.3 IU/kg for NPH, 3.0 mg/min/kg for 0.8 IU/kg for Levemir, and 0.95 mg/min/kg for 0.3 IU/kg for Lantus.

These values were used to calculate respective coefficients, which will be used in the next step.

```
Coeff (mg/kg/min) = number of IU in the
record * (1) / ((2) * weight)
```

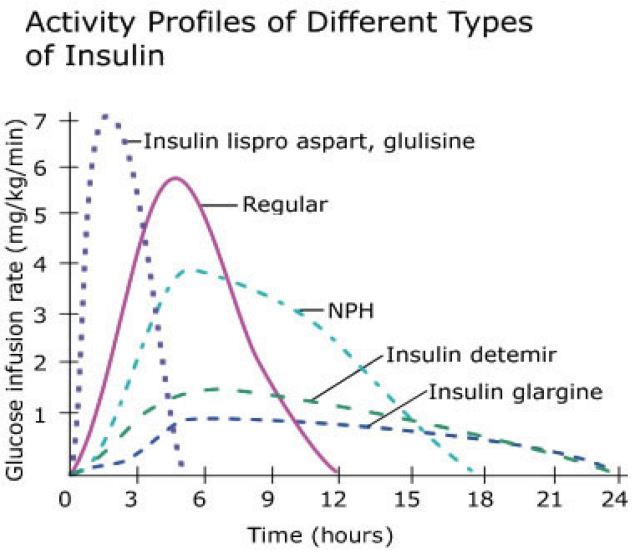


Figure 6. Glucose utilization rate for different types (examples) of insulin. Rapid-acting insulin is represented by Lispro and Aspart. Short-acting insulin is represented by Regular. Intermediate-acting insulin is represented by NPH. Long-acting insulin is represented by Detemir and Glargine. Source [7]

The next step is to find the current influence and the full influence of the record, by calculating the area under the action curve for particular type of insulin. This is implemented as a loop that iterates through the whole duration line: from zero, when the record was made, until the defined duration, and outputs a sum in mg/kg. The influence is measured in mg/kg/min, so each iteration step is defined as 1 minute.

Each iteration is as follows.

```
Res += Coeff (mg/kg/min) * Influence by
the moment in time (%/100)
```

The functions used for calculating the influence over time build the action curves, which are estimated versions of the curves described in the previous step (see Figure 6 as general example). An example of such estimated curve for the insulin type Humalog is shown in Figure 7.

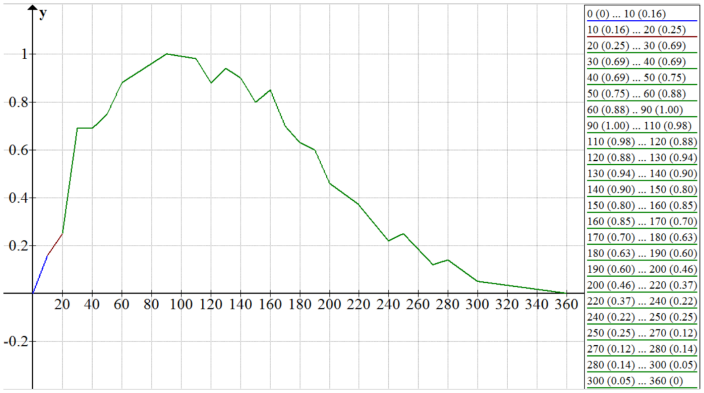


Figure 7. Estimated action curve for Humalog. X-axis represents minutes; Y-axis represents action from 0 to 1.

The result of this loop is the total influence during the given time in mg/kg. Current influence is calculated in the same way but with one difference – the loop runs not until the end of the duration, but until current moment.

The last step is to convert the results into mmol/L. We know that 1 mole of glucose is 180.15588 g of glucose, or 1 mmol of glucose is 180.15588 mg of glucose. In addition, both insulin and glucose are distributed in the extracellular space in the body. Extracellular space consists of interstitial space (~16% of weight) and blood plasma (~55% of blood volume). So the following conversion was implemented:

```
# (mmol/L) = # (mg/kg) * weight (kg) *
(0.16 + (0.55 * blood volume / weight)) /
      (180.15588 * blood volume (L))
```

Now, we have the current influence and the total influence that will be made by an engine insulin record. Before the module finishes its execution, the same algorithm as in the end of Carbohydrates Module runs. All calculated values are written to the state, and the records which action is finished are marked as deleted.

Activity Module

Aerobic and anaerobic activities have different effects on BG level. According to Bacchi and coauthors [14], aerobic activity causes 30% increase in Insulin Sensitivity because muscles absorb more glucose (which causes a drop in BG) [9] [10] [11]. This increase lasts for the next 24-72 hours after the exercise, depending on the duration of the exercise. During the first 1-3 hours, anaerobic (resistance) activity increases BG by 2 to 4 mmol/L [8], and sometimes even more [13], which is caused by adrenalin [12]. After this, the 15% increase of Insulin Sensitivity [14] takes effect and lasts about 14-20 hours [8]. This information is the basis of the Activity module algorithm.

Blood Glucose module

This module calculates the resulting blood glucose from the Insulin, Carbohydrate and Activity modules.

Additionally, every time the module is called, it subtracts a small value representing the constant glucose consumption by brain (about 100g per day, or about 4g per hour [15]).

Next steps will be to implement at least two other important elements that the Engine should have: Hypoglycemia unawareness option, and Dawn Phenomenon simulation (the rise in blood glucose during the night caused by the hormones). If they are implemented in the future, they will be developed in Blood Glucose module.

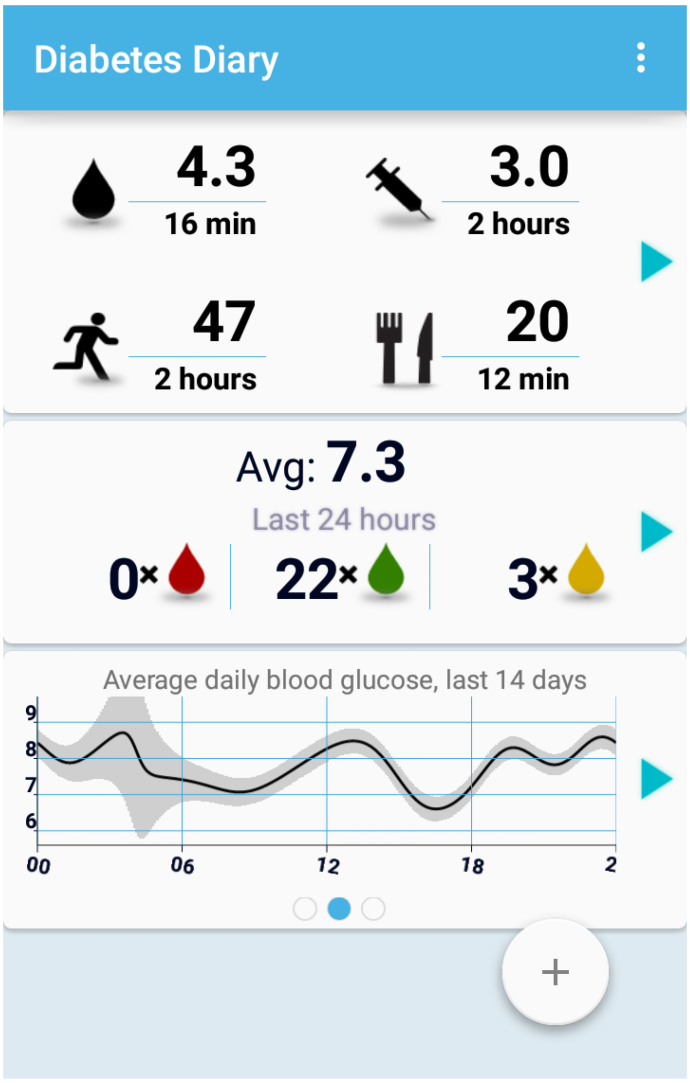


Figure 8. Diabetes Diary, a mobile self-management application for people with diabetes by NST.

Main Module

The main Diabetes Automata module initializes the engine and runs the simulation loop repeatedly where it calls Insulin, Carbohydrates, Activity and Blood Glucose modules with the user-specified frequency. For example, it is also possible to run one simulation-hour per minute. Additionally, it provides the API (Application Programming Interface) for external applications to use the Engine code in e.g. serious games or simulation-based self-management applications.

Results

The current version of the Diabetes Automata is a functioning prototype, presented in Figure 9 - Figure 16. Figure 9 presents the page for engine customization (settings), which includes age, gender, height, weight, insulin types used, preferred language, and other parameters.

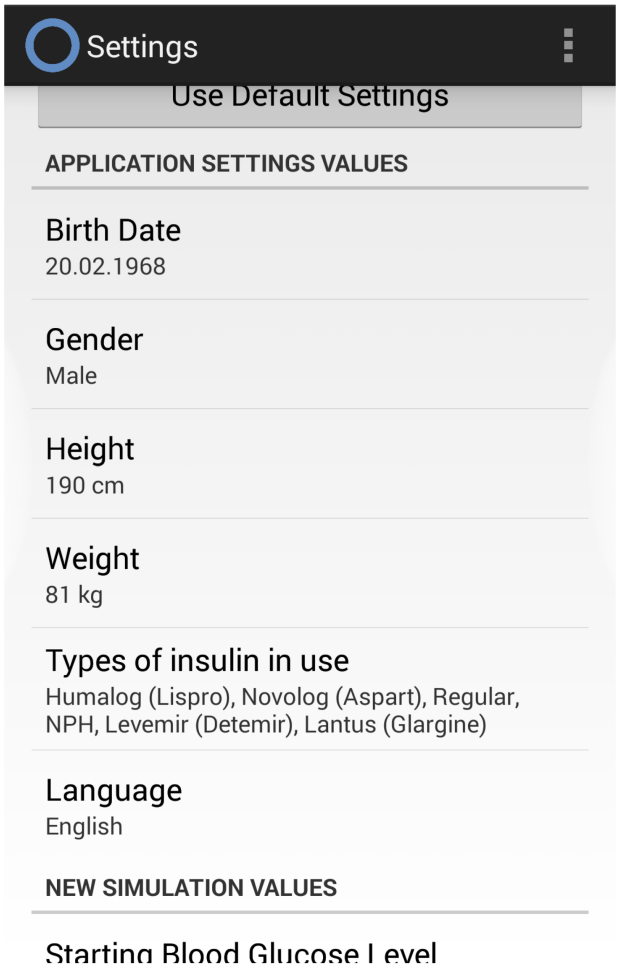


Figure 9. Settings screen

The prototype provides three simulation modes (see Figure 10). The first mode is the usual simulation where the user inputs new records into the Engine by him/herself. The second mode is the simulation that uses records from Diabetes Diary database by fetching the database records and inputting them into the engine automatically, during the simulation time. This mode is currently personalized for one of the test persons, and has the following default settings: the Engine takes records from Diabetes Diary, use of insulin, long-lasting insulin is Lantus, carbohydrates have medium GI, and physical activity is aerobic. The third mode is a real-time simulation paired with Diabetes Diary. It starts from scratch without the opportunity to change the simulation speed or pause. From this, every time user makes a new record in Diabetes Diary, the record is automatically added to the running simulation.

Figure 11 and Figure 12 show the main screen of Diabetes Automata with started simulation, offering all information necessary to understand the simulation. We can see that the simulation is set to 10 Min / 1 Sec. This means that for each second the result for a ten minute interval is presented. The duration field presents the progress in simulation-time. Time 01:39 (see graph) is the current simulation time between 00:00 and 23:59.

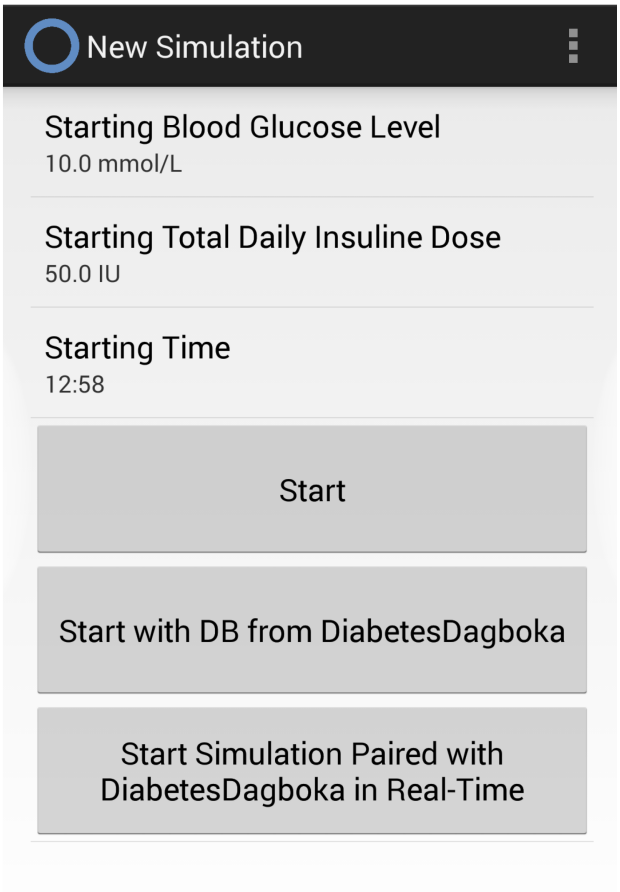


Figure 10. New Simulation screen

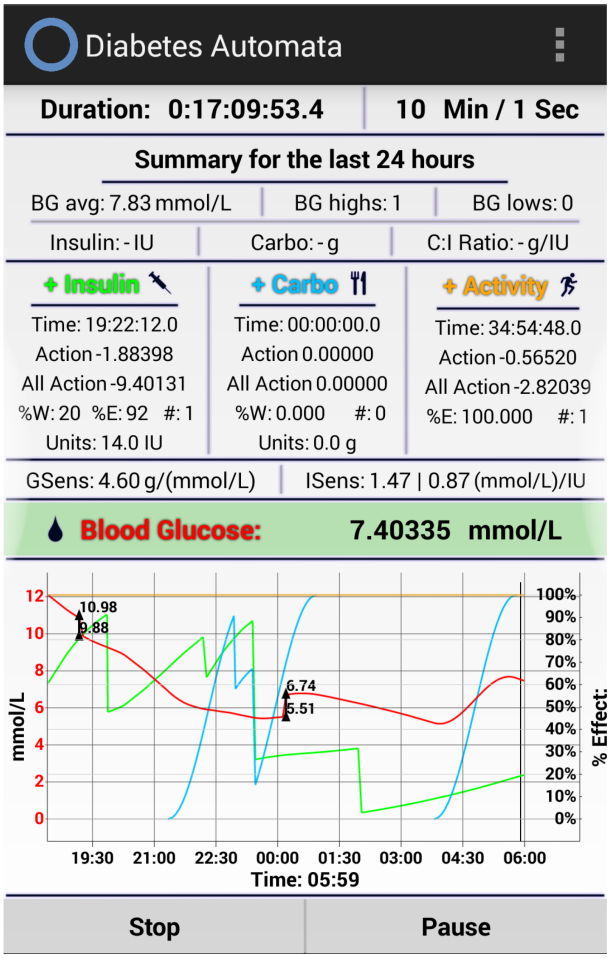


Figure 11. Simulation. Main Screen

The meaning of presented Insulin, Carbohydrates and Activity values in three columns on the main screen follows these calculations. The first value in each column is the time until that the longest effect of simulation records will last. The second value shows current effect of these simulation records on BG level. The third value is the expected maximal effect of these simulation records on BG. The set of values after, displays the work (%W) and the current effect (%E) of these simulation records on BG level in percentage, and the number (#) of currently active simulation records. The last value in Insulin, Carbohydrates modules is the number of currently active module units.

Blood glucose is represented in mmol/L, and insulin, carbohydrates, and activity curves are represented in percentage of influence. The small black triangles with values (see graph) mean that a user manually calibrated the BG level.

Several functionalities are available for the user on the main screen. One can make a new simulation record by pressing on Insulin, Carbohydrates, and Activity, manually calibrate BG level by pressing Blood Glucose, change the simulation speed by pressing on the time field above Summary, change the graph mode by pressing on the graph, and stop or pause the simulation by pressing the buttons below the graph.

The user can also save the entire current simulation, load it later and continue. This can be done via the options menu in the upper right corner of the screen (see Figure 11). This option is available only for the first simulation mode.

The graph can be set to display the last 1, 2.5, 6.5, 12, 24, 48 simulation-hours, or set to the automatic drawing mode which sets the zoom to one of these values depending on the simulation speed. The graph also has different curve modes. First mode (see Figure 11) represents blood glucose levels, together with action progress (in %) for Insulin and Carbohydrates, and the action strength for Activity. The second mode (see Figure 12) differs with the insulin representation. It is represented as the action strength over time (in %). The third mode is just for blood glucose levels. The user may adjust the graph size by selecting a size menu, presented after holding a long-tap during the simulation (see Figure 13).

When the Pause button is pressed, the graph on the screen contains all information from the beginning of the simulation, which can then be zoomed in or out and scrolled within. The default pause graph shows the last 24 hours. The “Summary” shows the number of blood glucose lows, highs and average level, as well as the total amount of grams of carbohydrates, IU of insulin, and the relation between the latter two, for the last 24 hours.

One of the effects that can be studied with the help of Diabetes Automata is the different types of insulins’ influences on the blood glucose level (Figure 14). The challenge for this project was to handle the complexity and variety of insulin types.

The Diabetes Automata aims to emulate the body’s metabolism. The key input parameters for the engine are insulin, blood glucose, carbohydrates and physical activity. Each of these parameters has a set of alternatives, as illustrated in Figure 14 - Figure 16.

The final version is expected to be completed during the summer 2015.

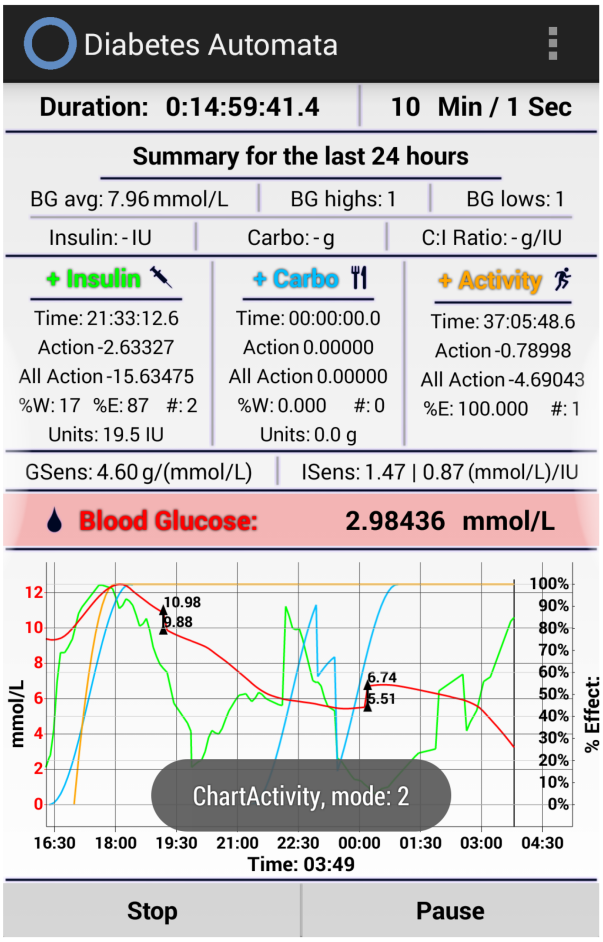


Figure 12. Main Screen, Simulation

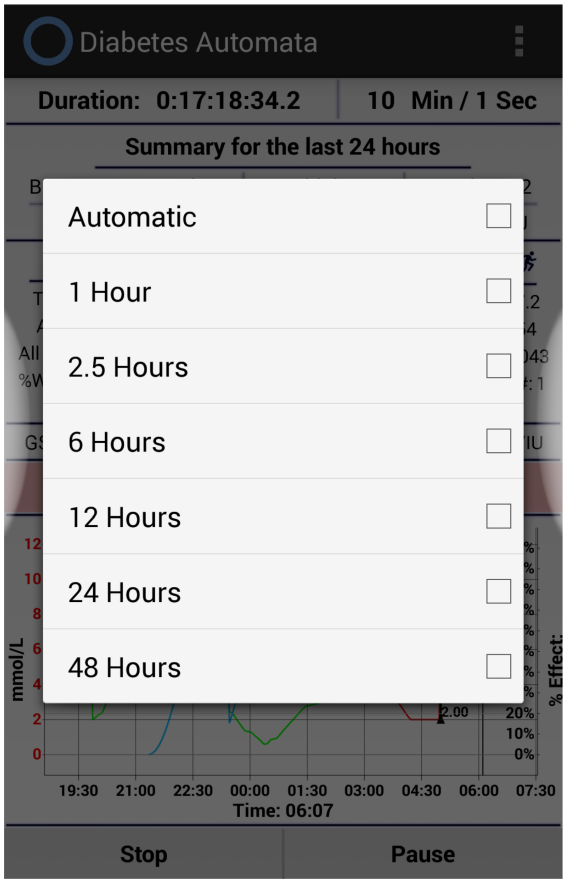


Figure 13. Choosing the graph zoom modes

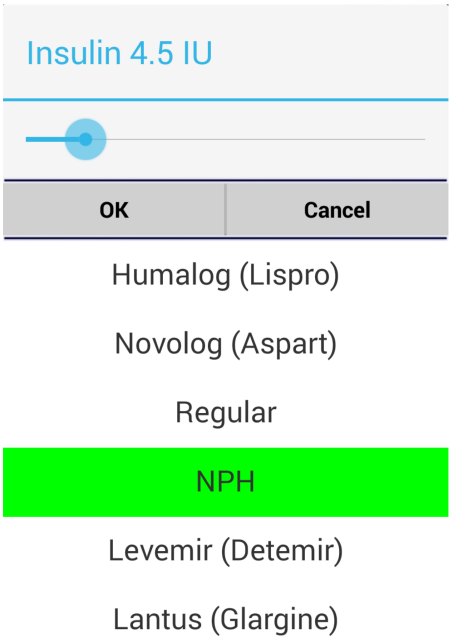


Figure 14. Make a new insulin registration during simulation

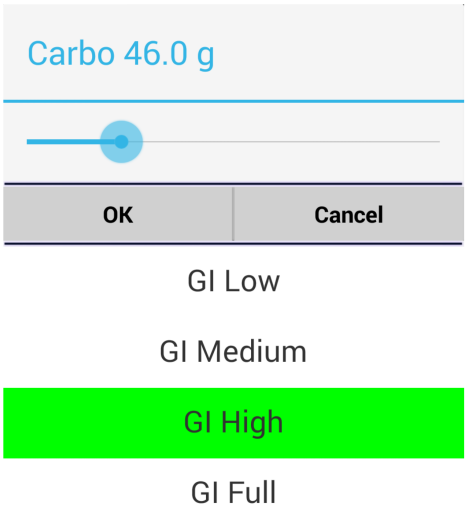


Figure 15. Make a new carbohydrate registration during simulation

Conclusion

We have developed Diabetes Automata, a software engine that offers the opportunity to run blood glucose simulation at user-defined time-speed.

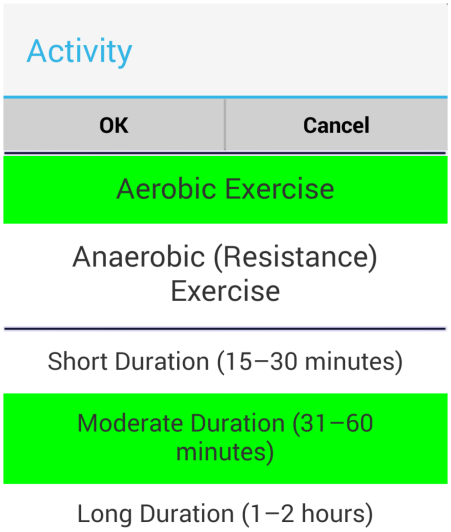


Figure 16. Make a new activity registration during simulation

The main challenge with the simulator is to make it as realistic as possible. The Diabetes Automata aims to emulate the body's metabolism. We expect that our simulator can be used as part of serious games and contribute through this to a better understanding of the interplay between insulin, carbohydrates, physical activity, and blood glucose levels. This could make games closer to reality and, therefore, more motivational.

The system is not yet evaluated or tested, and is currently under development.

Acknowledgments

This project is partly supported by the Research Council of Norwegian, grant no. 174934.

References

[1] World Health Organization. *Diabetes, Fact sheet N°312*. [Accessed 2015 26 April]; Available from: http://www.who.int/mediacentre/factsheets/fs312/en/

[2] Nasjonalt senter for samhandling og telemedisin. *Diabetesdagboka*. [Accessed 2015 26 April]; Available from: http://telemed.custompublish.com/better-quality-of-life-for-patients-with-diabetes.5208977-287081.html

[3] Årsand, E., Tatara, N., Østengen, G., Hartvigsen, G. Mobile-Phone-based Self-Management Tools for Type 2 Diabetes – The Few Touch Application. *Journal of Diabetes Science and Technology,* March 2010, 4(2), 328-336.

[4] Årsand, E., Frøisland, D.H., Skrøvseth, S.O., Chomutare, T., Tatara, N., Hartvigsen, G., Tufano, J.T. Mobile Health Applications to Assist Patients with Diabetes: Lessons Learned and Design Implications. *Journal of Diabetes Science and Technology*, September 2012, 6(5), 1197–1206.

[5] Bernstein, Richard K., *Dr. Bernstein's diabetes solution : the complete guide to achieving normal blood sugars*, 2007, 325

[6] GlycemicIndex.com, *About Glycemic Index. [Accessed 2015 29 April]; Available from: http://www.glycemicindex.com/about.php*

[7] DiabetesEducationOnline, *Types of Insulin [Accessed 2015 29 April]; Available from: http://dtc.ucsf.edu/types-of-diabetes/type2/treatment-of-type-2-diabetes/medications-and-therapies/type-2-insulin-rx/types-of-insulin/*

[8] Adams, O. Peter, The impact of brief high-intensity exercise on blood glucose levels. *Diabetes, Metabolic Syndrome and Obesity: Targets and Therapy,* 2013, 6, 113-122

[9] AmericanDiabetesAssociation, *Factors Affecting Blood Glucose [Accessed 2015 29 April]; Available from: http://www.diabetes.org/living-with-diabetes/treatment-and-care/blood-glucose-control/factors-affecting-blood-glucose.html*

[10] Diabetes.co.uk, *What Affects Blood Sugar Levels [Accessed 2015 29 April]; Available from: http://www.diabetes.co.uk/blood-glucose/what-affects-blood-glucose-levels.html*

[11] AmericanDiab, etesAssociation, *Blood Glucose Control and Exercise [Accessed 2015 29 April]; Available from: http://www.diabetes.org/food-and-fitness/fitness/get-started-safely/blood-glucose-control-and-exercise.html*

[12] Kris Berg, *Exercise Often Raises Blood Glucose in Type 1 Diabetes [Accessed 2015 29 April]; Available from: http://www.diabeteshealth.com/news/exercise-often-raises-blood-glucose-in-type-1-diabetes/*

[13] Fd Turner, D., Luzio, S., Gray, B. J., Dunseath, G., Rees, E. D., Kilduff, L. P., Campbell, M. D., West, D. J., Bain, S. C., Bracken, R. M., Impact of single and multiple sets of resistance exercise in type 1 diabetes, *Scandinavian Journal of Medicine & Science in Sports,* 2015, 1, 25, E99-E109

[14] Bacchi, E., Negri, C., Zanolin, M. E., Milanese, C., Faccioli, N., Trombetta, M., Zoppini, G., Cevese, A., Bonadonna, R. C., Schena, F., Bonora, E., Lanza, M., Moghetti, P., Metabolic Effects of Aerobic Training and Resistance Training in Type 2 Diabetic Subjects A randomized controlled trial (the RAED2 study), *Diabetes Care,* April 2012, 35, 676-682

[15] Diapedia, *Brain glucose metabolism [Accessed 2015 29 May]; Available from:* http://www.diapedia.org/metabolism/brain-glucose-metabolism

Address for correspondence

Aleksandr Agafonov, Department of Computer Science, University of Tromsø - The Arctic University of Norway, Norway
email: agych.alexander@gmail.com, tlf: +4794103421

Appendix C: Prototype Version History

During the development, the author has been writing a log of changes coming with every new version of the prototype. This information looks like a draft and has been written for himself (in order to remember what has been done over time), for supervisors (in order to show how the development is going) and for testers (to make them know which changes the new-coming prototype version has). The log is presented below.

26.12.2014., v0.001a:

- The first version.
  Implemented insulin module, based on the TBW, I:C ratio, modified formulas from the description of "0.12" rule, and very estimated self-implemented curve for work done over time, with the blood glucose level coming back to where it was before the action of insulin.
- Implemented Carbohydrate module (with 4 types of GI), based on the TBW, Insulin: Carbohydrate ratio, modified formulas from the description of "0.12" rule, and estimated self-implemented curve for work done over time. The blood glucose level coming back to where it was before the action of carbohydrate.
  Comment: Engine working, with implemented Insulin module (only Humalog) and Carbohydrate module (with 4 types of GI).

29.03.2015., v0.001b:

- Re-implemented Insulin (only Humalog) and Carbohydrate modules - the blood glucose level doesn't go back anymore.

05.04.2015., v0.002a:

- Implemented activity module, for both aerobic and anaerobic.
- Implemented algorithm for Lantus.
- When pressing "+Insulin" "+Carbo" "+Activity" "Blood Glucose" or Time Speed, the simulation is paused.
- The Engine is now language-independent. Before, there were some English definitions like "GI High" or "Male" for example. The definitions are re-implemented as Enums.

11.04.2015., v0.002b:

- Now, if "insulin in use" setting is empty, the simulation can't be started; and if the user presses "new simulation", he/she sees error message.

13.04.2015., v0.002bb:

- X-axis now shows hours instead of minutes (the last 12 hours are visible).
- Graph drawing is now more optimized.
- Added insulin effect % curve.

14.04.2015., v0.002c:

- 2 graph modes:
  - Mode 1: like it was before, just blood glucose curve;
  - Mode 2: 4 curves: Blood Glucose, Insulin, Carbohydrate, Activity: with different colors (same colors are assigned to the corresponding buttons).

Tap on the graph to change the mode (not during the pause, pause is for zooming/moving the graph).
Comment: after 2 (real) hour of simulation the performance in the mode 2 becomes quite slow, much slower that in the mode 1.

- Even more optimized graph drawing.

16.04.2015., v0.002d:

- Graph drawing now very good optimized (for both modes):
  The application's response doesn't become slower after many (real) hours of simulation and doesn't get overloaded (sometimes it does in the mode 2 a little bit, but this effect disappears).
  The program can be running during a very long (real) time now.
- Mode 2: insulin, carbohydrate, activity curves improvement:
  Before: they have been drawn always, even when not active (so during inactive time drawn as line(s) x=0).
  Now: they are like separated curve-pieces with separated datasets, with each of them starting from 0% and finishing with 100% (or 0% if it is an activity), which means that they are drawn only when active - currently affecting blood glucose.
- Visualization of hours on x-axis is now better and displayed in a normal way when zooming in/out during pause.

17.04.2015., v0.002e:

- Added option to enable/disable negative blood glucose values (press "...").
- X-axis now represents time instead of hours (00:00, 00:30, 01:00, etc.).
- Various changes made in the graph representation.

18-19.04.2015., v0.003a:

- Added blood glucose consumption by brain value that is constantly subtracted from the current blood glucose level.
- Added option to enable/disable the constant subtraction/addition described above (press "...").
- Added new simulation parameter: starting time. Default starting time equals to the current time:
  if for example the time now is 16:08, the simulation time is set to the same time, but the user can change it when starting a new simulation;
- Improved Insulin:Carbohydrate ratio formula:
  instead of 500-rule (which actually overestimates the ratio), 300-450-400-rule is used, where the dividend changes from 300 to 450 depending on the current (simulation) time:
  (breakfast) between 6:00-9:00 - dividend = 300;
  between 9:00-12:00 - dividend increasing from 300 to 450;
  (lunch) between 12:00-14:00 - dividend = 450;
  between 14:00-17:00 - dividend decreasing from 450 to 400;
  (dinner/supper) between 17:00-21:00 - dividend = 400;
  between 21:00-6:00 - dividend decreasing from 400 to 300.
- Implemented algorithm for Novolog-insulin
- Implemented algorithm for Regular-insulin
- Implemented algorithm for NPH-insulin
- Implemented algorithm for Levemir-insulin

21.04.2015., v0.003b:

- UI updates twice more frequently during the simulation.
- "Set time speed" dialog: the number of digits is now limited to 2
  (the maximum speed that can be set is 99 min/sec instead of over9000 min/sec).
- Added one more information field for insulin, carbohydrate, activity modules - # of currently active records;
  (for carbohydrate and activity there are usually 1 active record per time, but for insulin it can be 2, for example if the user uses Humalog and Lantus at the same time).
- The time management is now better, more precise:
  before, if the speed was for example 10 sec/sec, the actual speed was around 10.1-10.2 because the duration of the modules' calculations was not taken into account and subtracted, this problem could be observed very well in the "Duration" field during the simulation;
  now the engine calculates its iteration duration and subtracts it, so the time is much more accurate now;
  but not super-ideal (can be visible on the high speed, 60 min/sec for example: sometimes 1-2 extra deciseconds are added).

21.04.2015., v0.004a:

- Added manual and recovery "Save game", or in the project's case, "Save state".
  - Manual save:
    When the simulation is running, the state of the entire current simulation can be saved manually when user wants (in the menu "...").
    When the simulation is stopped, or the application is closed and restarted, or the phone is restarted, or etc., the manually saved simulation can be loaded and continued like it was just paused.
  - Recovery save:
    When the user swipes out the application from the android task manager, or in other cases when "onDestroy" method is called, the prototype automatically saves the simulation with the latest simulation data.
    When the prototype is launched again, the user sees a recovery dialogue and must choose either to recover the simulation, or to delete the saved recovery state.

  Manual and Recovery save are 2 different files.
  The size of the save file, and the time needed to save/load state, depends on how long (real execution time) the simulation has been running.
  Currently, there are only manual (in the "..."-menu) and recovery state save; in one of the next versions, automatic frequent save (for example once in a (real) minute) can be added.
  Currently, only one file with some hardcoded name can be saved manually.
- Added foreground program service, which:
  - Keeps the application and the simulation always running.
  - Prevents the application from being closed and the simulation from being interrupted and stopped in case of the lack of free RAM.
  - Makes the application be available from the system sliding menu (notifications).
  - Displays the current blood glucose level, simulation time and simulation duration in the system sliding menu (notifications), updated every 2 seconds.
- Implemented the handling of the main Engine simulation thread fatal error that was causing the simulation crash in the previous versions:

The simulation crash was happening because sometimes 1 loop iteration was taking more time than the engine update frequency step.

- The main Engine simulation thread priority is set to "High" when the simulation is running, in order to make the error described above happen rarer and to handle it rarer.
- "Back" button action changes:
  If the simulation is not running and "Back" button is pressed, the program gets closed, as usually.
  If the simulation is running and "Back" button is pressed, the program doesn't get closed and doesn't interrupt the simulation, it gets minimized and remains in the background, just like if "Home" button was pressed.
- Added blood glucose field background color/blinking:
  - When the current blood glucose level is in the normal range, the background is green;
  - When the current blood glucose level is between 10 and 15, the blood glucose field background starts blinking yellow;
  - When the current blood glucose level is higher than 15, the blood glucose field background starts aggressively blinking yellow;
  - When the current blood glucose level is between 3 and 4, the blood glucose field background starts blinking red;
  - When the current blood glucose level is lower than 3, the blood glucose field background starts aggressively blinking red;
- Added logo.

22.04.2015., v0.004b:

Further changes are sub-divided into Engine and Android UI.

Engine:

- Implemented parallel execution in the Engine:
  Insulin and Carbohydrate modules run in parallel, the engine waits until all 3 threads complete the calculations, and then finally runs the blood glucose module.
  The execution of the set of these 2 modules is now approx. 0-30% faster than before.
  Activity module is currently excluded from the parallel group because it affects insulin sensitivity, which is used by the Insulin module, and the Activity module must run only after Insulin module. Including the Activity module in the parallel group causes unnormal behavior of Insulin sensitivity, and therefore also unnormal behavior of the effect of Insulin on blood glucose and the blood glucose level.
- In case of real-time simulation (1 sec/sec): when a new record dialog or set time speed dialog is called during the simulation, the simulation doesn't get paused automatically. It happens only when the time speed is more than 1.

Android UI:

- Notification update is changed from 2 seconds to 1 second.

24.04.2015., v0.005a:

Android UI:

- When the blood glucose level is corrected, this change is displayed in the graph as a black line, in which (t, y1) coordinate = previous blood glucose level, and (t, y2) = new calibrated blood glucose level; both coordinates are marked with triangles and their blood glucose values.
- Various graph improvements.
- Implemented simulation of records from the Diabetesdagboka:
  press "New Simulation" -> start DB simulation.

The timestamp of the first blood glucose database record is taken as 0 in time, as the beginning of the simulation time.
The "while" loop, which runs in a separate thread, checks if the time for next future record (blood glucose calibration, insulin, carbohydrate, activity) has come, and in positive case sets the record into the engine.
Information from the DB-owner: the insulin data in the DB is only about Humalog injections; Lantus injections of 16 IU should be added (so they are added) additionally and take place between 00:00 and 01:00 each simulation-day.
It seems like IPC doesn't work with huge databases (10-15 MB), and works with Demo-database (less than 0.5 MB, about 45 days between the first and the last blood glucose record, which means about 45 simulation-days).
(It is not possible to save/load a DB Simulation, only a usual simulation)
Requirements (otherwise DB Simulation will not work):
- Diabetesdagboka with IPC.
- Demo database imported into Diabetesdagboka.

25.04.2015., v0.005b:
Android UI:
- Reorganization of elements on the main screen.
- Added new information field for Insulin and Carbohydrate: sum of units/grams in currently active records.
- Added helping pop-up messages for the following elements on the main screen: Summary, Insulin, Carbohydrate, Activity, Blood Glucose.

30.04.2015., v0.006a:
Engine:
- Implemented dynamic calculations of Insulin sensitivity.
  Insulin sensitivity depends on the relation between the sum of insulin units and the sum of carbohydrate -grams eaten during the last 24 hours.
  During the first 24 simulation-hours, when such data is not available yet, the "Starting TDD" and 300-450-400 Formula are used instead.
- New method for glucose sensitivity calculation (the previous one was bad).
- Implemented calculations for 24h-summary.
- Changed duration and functions/equations for GI High, Medium and Low, the effect duration is set to 60, 90, 120 respectively (GI 100 is still 45 minutes).
- Changed duration and functions/equations for Humalog and Novolog, the effect duration became shorter than before.
- Glucose and Insulin sensitivity values are now calculated in the State class and are available for the external UI.

Android UI:
- Reorganization of elements on the main screen.
- Implemented summary 24h: blood glucose average, highs and lows; insulin units, carbohydrate grams, C:I ratio - for the last 24 simulation hours.
- Added new information field - Glucose and Insulin sensitivity.
- Dynamic calculations of Insulin sensitivity can be enabled/disabled in the "..."-menu.

04.05.2015., v0.006b:
Android UI:
- Fixes.

- The prototype supports phones with different DPI scale factor.
  Now the size of these Graph-elements depend on pixel density and will have the same good scale on phones with different pixel density.

21.05.2015., v0.007a:

Engine:

- Big re-implementation of all Insulin algorithms in general, and for Humalog, Novolog, Regular, NPH, Levemir, Lantus in particular.
  The algorithm is based not on the C:I for the last 24h and Glucose Sensitivity, but on Glucose Utilization Rate, the calculations of the area under the effect strength over time curve.
- Improved calculation of the constant resting blood glucose decrement, based on the solution that was found for insulin. The constant is not randomly-taken anymore, it is exactly 4 grams of blood glucose consumed by brain per hour.
- New second Insulin Sensitivity variable - insulin sensitivity for long-acting insulins.
  (assuming that user uses only one long-acting type of insulin per time).
- New calculation of old Insulin Sensitivity variable (for Humalog, Novolog and Regular)
  (assuming that user uses only one rapid/short-acting type of insulin per time).
- Calculations for the new state variable - effect strength (in addition to the variable of work done).
- When the simulation is paused, the engine completely stops processing and doesn't take any CPU-time.
  Before: during the pause the engine was running in an empty (passive) loop and doing nothing.

Android UI:

- New simulation mode - real-time simulation paired with Diabetesdagboka:
  Set new records in Diabetesdagboka, and they will appear in the simulation.
  Pause and Time Speed are disabled, since the simulation must be real-time.
  It is possible to set new records without Diabetesdagboka (like in regular simulation), if you need to make records that are different from automatic Humalog, Lantus, Carbohydrate Medium GI and Aerobic Activity.
  If the Diabetesdagboka record remains in Edit mode longer than 1 minute, it will not be considered by the simulator.
  (1 minute should be enough to set in a new record in Diabetesdagboka).
  IPC mistakenly reads the records that are deleted. For this case, manual blood glucose calibration in the simulator is now activated for all simulation modes.
  If you delete a record from Diabetesdagboka, it will be set in the Simulator like a newly created Diabetesdagboka record, so don't delete Diabetesdagboka records.
  Requirements (otherwise it will not work):
  - Diabetesdagboka with IPC.
  - The Diabetesdagboka DB must be less than over9000 MB, since the IPC doesn't work with huge databases.
- 3rd Graph mode. The difference from the Graph mode 2: representation of the insulin effect strength instead of insulin work done.
- Automatic Graph zoom.
  Speed <= 30 Sec / Sec --> Graph 2.5 hours.
  Speed <= 3.(3) Min / Sec --> Graph 6 hours.
  Speed <= 10 Min / Sec --> Graph 12 hours.
  Speed <= 30 Min / Sec --> Graph 24 hours.

Speed > 30 Min / Sec --> Graph 48 hours.

- Manual Graph zoom: 1, 2.5, 6, 12, 24, 48 hours.
- Zoom mode menu. Long tap on the graph: choose between Automatic and Manual graph zoom modes.
  Zoom menu can be called only when the simulation is not started and when the simulation is running; it is not available during Pause.
- Added "Loading" mini-window with progress circle in the following cases:
  When rendering full simulation graph right before pause; when starting a new DB simulation; when saving or loading saved simulation.
- One more insulin sensitivity variable is displayed - the one for long-acting insulin.
  (assuming that user uses only one long-acting and one rapid/short-acting type of insulin per time)
- UI changes to make the application more comfortable for the user:
  - Default simulation speed is set from 2 minutes to 1 second (real-time).
  - "Min / 1 Sec" and "Sec / 1 Sec" in the simulation time dialog is changed to "x Min / 1 Sec" and "x Sec / 1 Sec".
  - Improvements of the input field in the simulation time dialog.
- BG < 2.0 parameter is set to "disabled" by default.
- Various fixes.

26.05.2015., v0.007b:

Engine:

- Changes in Carbohydrate module algorithms.

Android UI:

- Simulation mode 2 and 3 now set the records in the right time and can work with huge databases.
  Requirements:
  - Diabetesdagboka V1.5.3 (ee7a27fb) with IPC.
- In the 2nd and 3rd simulation mode: fixed problem when the time of making a new simulation record was wrong.
- In the 3rd simulation mode: simulation time speed automatically set to real-time (should have been done in v7a).
- Reorganization of elements in the Ins/Carbohydrate/Act buttons.

27.05.2015., v0.007c:

Android UI:

- Fixes.
- When setting to pause, the full graph is always zoomed to the last 24 hours.

28.05.2015., v0.007d:

Engine:

- Multiple additional iterations on high speeds:
  - Pros: for providing better calculations and much more detailed simulation information when the speed is high (1-99 minutes per second).
  - Cons: much bigger CPU consumption.

Android UI:

- On high speeds, the simulation curves are detailed and not cornered like before.

02.06.2015., v0.008a:

Engine:

- Simple implementation of the Dawn Phenomenon. Can be turned on and off.
- Improvements, changes and restructure of the algorithms in the Insulin module.
- Changes in the Main module.

Android UI:

- All clickable elements on the main screen are now drawn and animated as buttons:
  The following clickable fields: Time, Insulin, Carbohydrate, Activity, Blood Glucose, Chart.
- Dawn Phenomenon on/off option in the "..."-menu.
- When setting a new time speed value, if "x Sex / Sec" is selected and the user enters value bigger than 60, it automatically becomes 60.

03.06.2015., v0.008b:

Android UI: big bug-fixing for Android 5 (since the development has always been going on Android 4.4)

- Compatibility with Android 5.

Android UI:

- Implemented Landscape orientation for the main screen.
- Now the simulation can be running even if the user swipes the program out from the task manager.
  If simulation was running for the moment of swiping out, it will be still running and updating the notification.
  Ones the user touches the notification or launches the program again, he/she will see the re-opened program with the ongoing simulation that he/she has previously started.

04.06.2015., v0.008c:

Engine:

- Dawn Phenomenon calibrated.

Android UI:

- Fixes.

05.06.2015., v0.008d:

Android UI:

- Attempt to optimize the usage of processor for simulation modes 2 and 3.
- Attempt to fix the integration with Diabetesdagboka that stops working in few hours on some devices.
- Fixes.

06.06.2015., v0.008e:

Android UI:

- The usage of CPU is well-optimized for simulation modes 2 and 3.
  According to profiler, in comparison to v8d, the CPU usage of the Android UI is now 10% smaller when the UI is on the foreground, and 250-300% smaller when the UI is in the background (when the UI is minimized or when the mobile screen is locked).

07.06.2015., v0.008f:

Engine:

- Implemented multithreading inside Insulin module.
  This module has the heaviest computations in the Engine.

Several active records significantly increase the amount of computation and CPU time in comparison to 1 active record.
Now, calculations for each active insulin record run in a separate thread.

- Counting of carbohydrates and insulin for the last 24 hours now starts immediately, not in 24 hours.
- Reimplementation of multithreading in the Main module.

Android UI:

- Fixed serious issue - the engine's modules were not running in the background when the phone screen was locked.
  Now, when the simulation is started, the simulator acquires a "PowerManager.WakeLock" with flag "PARTIAL_WAKE_LOCK", in order to keep the CPU running when the screen is turned off.
  The Lock is released when the simulation is paused or stopped.
  Acquiring this lock can increase battery usage during the simulation.
- In the simulation mode 3, the Diabetesdagboka record fetching timer is changed from 1 to 5 minutes.

08.06.2015., v0.008g:

Engine:

- Priority of all threads in the Engine is changed from 10 (High) to 5 (Medium) because of high CPU-usage in the previous version.

Android UI:

- Code restructure and re-implementation of some parts.

11.06.2015., v0.009a:

Engine:

- The state keeps history data only for the last simulation-month.
  Before: since Database module is not implemented, the state is holding all the history data, and the size of this data didn't have any limit.
- Fixed bug when the simulation duration was reset or not correct after loading saved simulation.
- Implemented argument check for the API methods (all public methods in Engine and Settings classes), where it was necessary.
- Reorganization of methods and removed unused methods.
- Fixes.

Android UI:

- Added "Exit" in the "..."-menu of the main screen.
  Completely quits the program and stops the "forever"-living background service.
  Automatic interrupted simulation save (for simulation mode 1) saves the simulation if it was running for the moment.
- Changed New Simulation Screen.
- Removed "Starting TTD" from New Simulation screen, Settings screen and Default Settings screen.
- Added "Allow BG < 2 mmol/L", "Allow dawn phenomenon" and "Allow BG consumption by brain" to the New Simulation screen and Settings screen.
- Added new graph zoom sizes: 5 days, 10 days, 30 days.
- Fixes.
- Code restructure.

14.06.2015., v0.009b:

Android UI:

- Manual simulation save now creates multiple simulation files with filenames containing date and time.
- New dialog - Load Simulation:
  When pressing "Load Saved Simulation", the dialog appears with the list of all simulation save files that are found in the directory.
  User can select one file, and either load selected simulation, or delete it.
  In the second case, confirmation dialog with "Delete" and "Cancel" buttons appears.
- Simulation save files are now located in "<internal storage>/Android/data/no.telemed.diabetesautomata/files/data/".
  Save and Load simulation are both using this path now.
  Unlike the old path, the new one can be easily accessed in order to copy or paste the simulation file in a file manager.
- Temporary save-file-moving feature:
  If there is a manually saved simulation file from the previous prototype versions in the old directory, the file is moved to the new directory and deleted from the old one.
- Changed dialog "Stop Simulation":
  Now it is called "Save Simulation Before Stopping?" and has 3 buttons: Cancel, Don't Save, Save.
- Added confirmation dialog for deleting interrupted simulation file, if there is one.
- Enabled (but not implemented) simulation saving for simulation mode 2 and 3 - they will be saved as simulations in the mode 1.
  Later, the simulation can be loaded, seen, and continued in the mode 1.
- Changed buttons on the New Simulation Screen.
- Changes and improvements of Toast-messages.
- Fixes.

17.06.2015., v0.009c:

Engine:

- Increase in insulin sensitivity caused by aerobic activity is changed from 30% to 15%.
- Improvements of activity module.

Android UI:

- Now Saving Interrupted Simulation works also when an application crash occurs.
  If the prototype has crashed for some reason, the simulation is saved on the disk and can be recovered when launching the prototype again.
  The procedure is the same as loading an interrupted simulation when starting the prototype.
  However, in some cases the crashed simulation is not saved.
- In case of application crash, the information about the crash is stored on the disk, in order to make it easier and faster to find and fix bugs.
  In case of crash, the file should be sent to the developer. The crash-logs can be found with help of a file manager.
  Path: "<internal storage>/Android/data/no.telemed.diabetesautomata/files/data/CrashLog/"
- Fixes.
- For Testing:
  - Hardcoded algorithm that simulates behavior of the insulin pump of test person #2.

- Added code for switching between testing person #1 and #2.
- Fixed carbohydrate-record auto-setting in testing-mode.

18.06.2015., v0.009d:

Engine:

- Changes in the Dawn Phenomenon implementation.
- Humalog strength is decreased.

20.06.2015., v0.009e:

Engine:

- Re-implemented effect strength percentage calculation for Humalog.
- Corrected Humalog peak-effect constant.
- Corrected Lantus peak-effect constant.
- Glucose sensitivity results are increased by 20%, since testing has shown that the Bernstein's book (2007) underestimates glucose sensitivity.

Android UI:

- Added Full Screen Graph activity, available from the "..."-menu when the simulation is running, both in portrait and landscape mode.
- Added flushing to all file operations.

22.06.2015., v0.009f:

Android UI:

- Simulation Save/Load path changed from "<internal storage>/Android/data/no.telemed.diabetesautomata/files/data/" to "<internal storage>/DiabetesAutomata/".
- Power consumption reduced.
  - Graph update frequency depends on speed, varies from 1 time per minute when real-time, to 1 time per second when 1 min / 1 sec or faster.
  - Fixed disabling of frequently repeating operations when engine is on pause and user opens "Time Speed" dialog.
  - Notification update frequency changed from 1 to 1.5 sec.
  - Project cleaned from unused stuff and garbage.
- Fixes.

25.06.2015., v0.009g:

Android UI:

- Currently running simulation is automatically saved as interrupted simulation in the mode 1, before the phone's battery is empty.

  When the battery level is below 2%, Diabetes Automata automatically saves running simulation as "Interrupted Simulation", stops the entire application (all processes and services).
- Changed Carbohydrate dialog: progress bar is replaced by usual number-input dialog that can contain maximum 3 digits.
- Default chosen (green) GI in the Carbohydrate dialog is changed from "GI Full" to "GI Medium".
- Added notification message "Simulation Paused", updated notification message "Simulation Interrupted".

26.06.2015., v0.009h:

Android UI:

- Implemented "About" screen, can be accessed through the options-menu of the Main, Settings and Default Settings screens.
- For Testing:
  - Glucose infusion by insulin is reduced by 40% for test person #2 that uses insulin pump.

29.06.2015., v0.009i:

Engine:

- Code cleaning.

Android UI:

- Code restructure and cleaning.
- Fullscreen graph: update frequency depends on speed, varies from 1 time per minute when real-time, to 1 time per second when 1 min / 1 sec or faster.

30.06.2015., v0.009j:

Android UI:

- "GI Full" changed to "GI 100", to make it clearer.
- For Testing:
  - Fixes for testing in simulation mode 1.

03.07.2015., v0.009k:

Android UI:

- Fixing of issues.

04.07.2015., v0.010, The Last Version:

Engine:

- Code cleaning.
- Deleting of unused files.

Android UI:

- Code cleaning.
- Deleting of unused files.

Appendix D: Previous Implementation Of Insulin Module

The older implementation of Insulin module was similar to the implementation of Carbohydrate module (see 6.5.7), was based on the current work done (influence percentage for the current moment in time), total body water and insulin sensitivity calculated as the division of C/I ratio by Glucose Sensitivity, and was calculated as follows:

```
Current influence (mmol/L) =
# IU * work done (%/100) * Insulin Sensitivity.

Insulin Sensitivity (mmol/L per IU) = C/I / Glucose
Sensitivity.

Glucose Sensitivity (g per mmol/L) = 0.18 * TotalBodyWater
(%) / 100 * weight.

where 0.18 is equivalent to 1 mmol/L.
```

The "work done over time" curves were based on Figure 4 – Figure 13, same as Figure 81 – Figure 86, but they were having much bigger level of approximated, same as the whole older implementation. The last implementation from section 6.5.6 makes much more accurate and proper calculations, since it is based on the estimated versions of the original curves (Figure 4 – Figure 13). The examples of the curves are presented in Figure 133 – Figure 138:

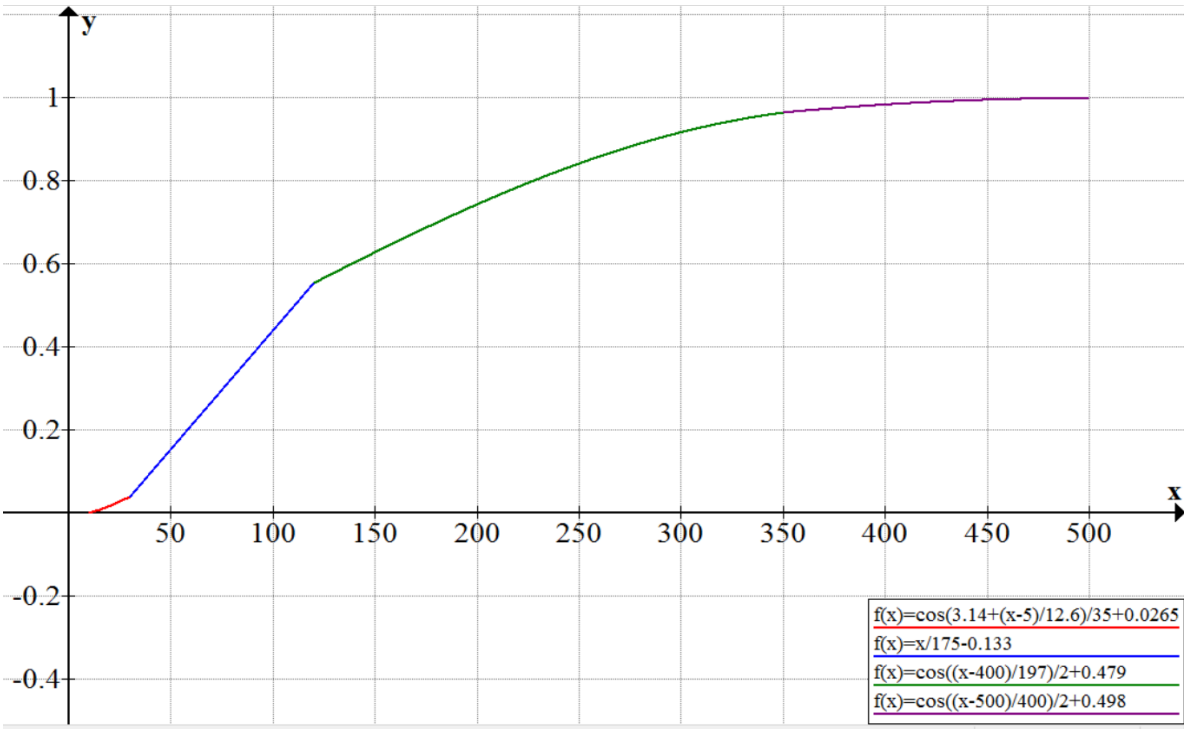


Figure 133: Curve used in the previous Insulin module implementation. The work done over time by Humalog.

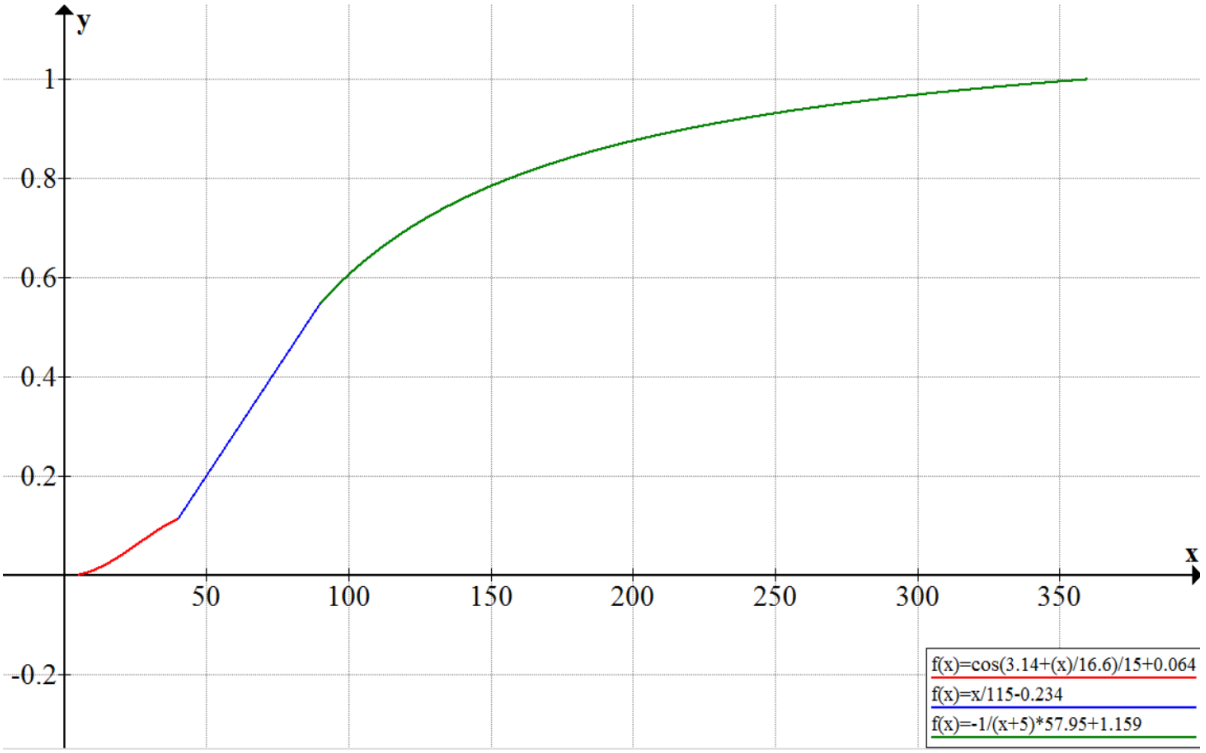


Figure 134: Curve used in the previous Insulin module implementation. The work done over time by Novolog.

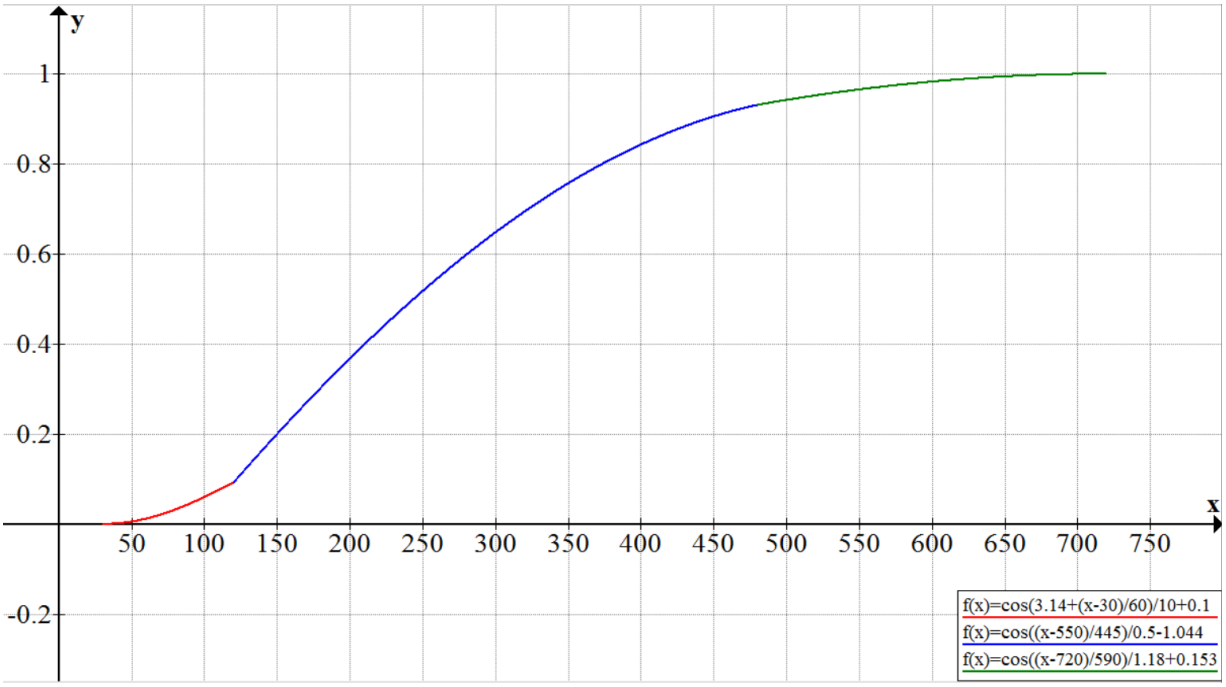


Figure 135: Curve used in the previous Insulin module implementation. The work done over time by Regular.

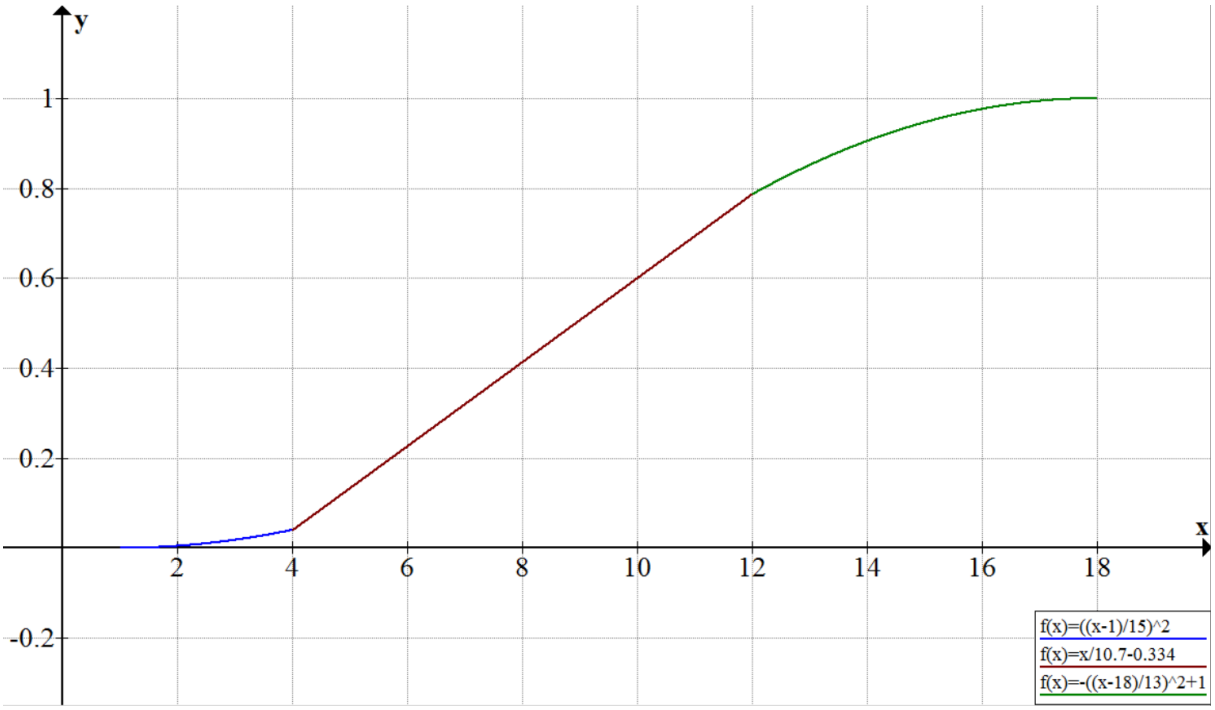


Figure 136: Curve used in the previous Insulin module implementation. The work done over time by NPH.

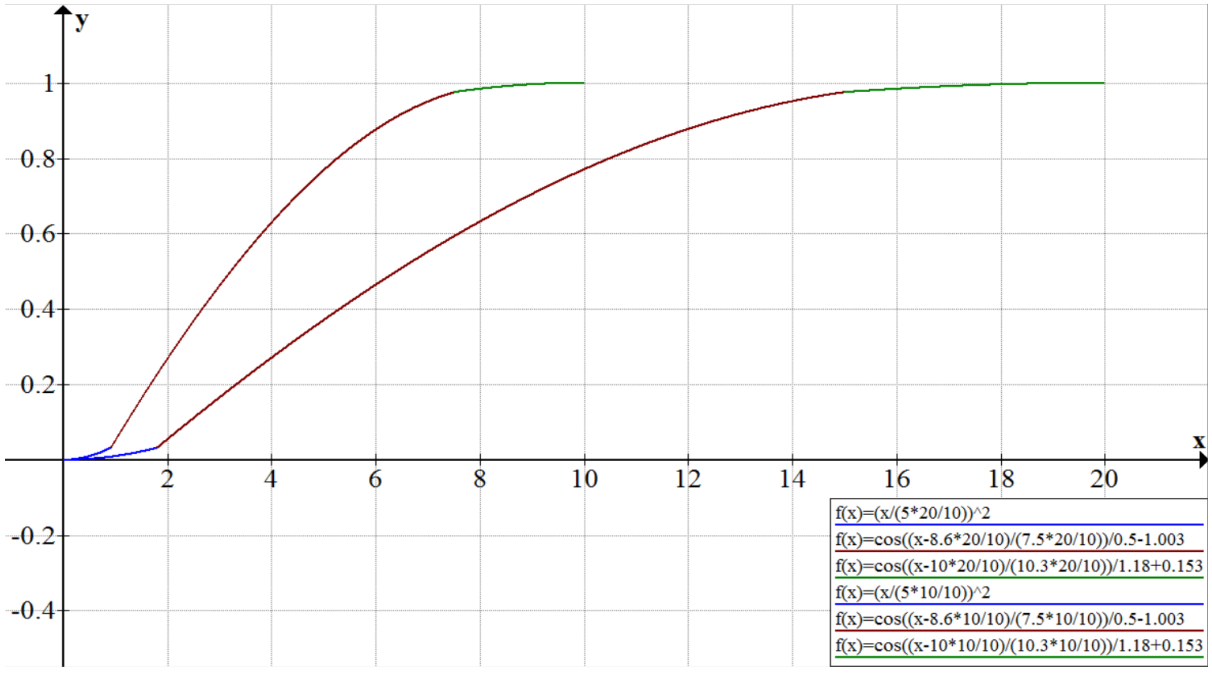


Figure 137: Curve used in the previous Insulin module implementation. The work done over time by Levemir, for t=10h and 20h.

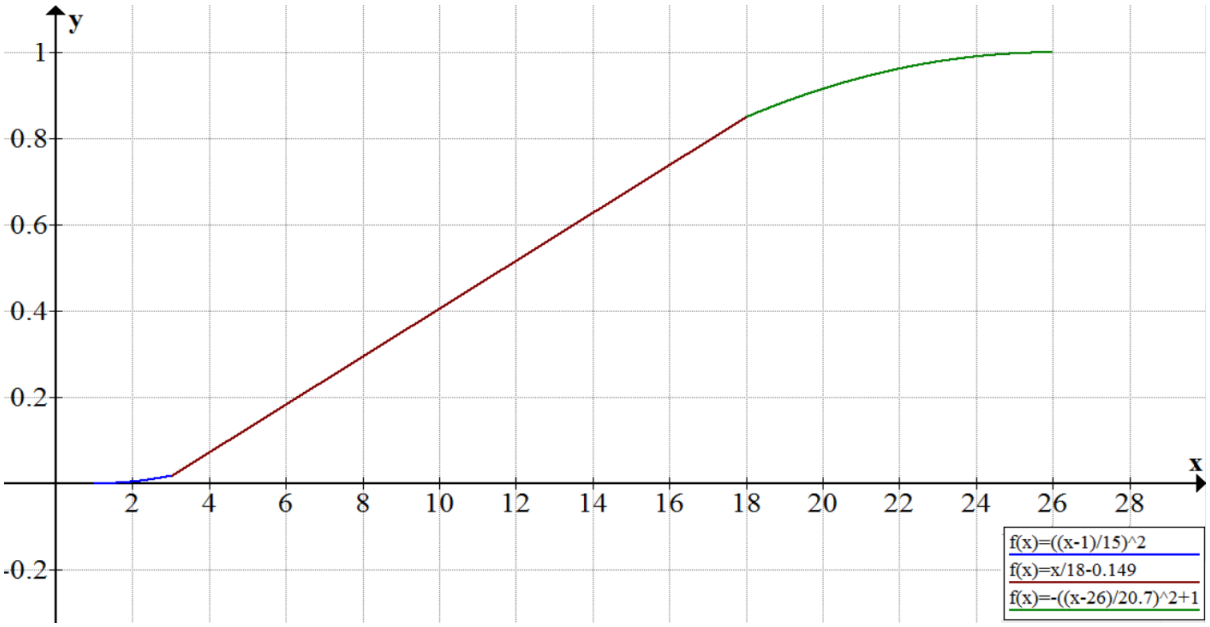


Figure 138: Curve used in the previous Insulin module implementation. The work done over time by Lantus.

References

4VIEWSOFT. 2013. *AChartEngine, website* [Online]. Available: http://www.achartengine.org/index.html [Accessed 23 June 2015].

ABBOTT DIABETES CARE. 2005. *Blood Glucose Measurement Units* [Online]. Available: https://web.archive.org/web/20110706100159/http://www.abbottdiabetescare.com.au/diabetes-faq-measure-units.php [Accessed 5 March 2015].

ADAMS, O. P. 2013. The impact of brief high-intensity exercise on blood glucose levels. *Diabetes, Metabolic Syndrome and Obesity: Targets and Therapy,* 6**,** 113-122.

ADOCIA.FR. 2014. Available: http://www.adocia.fr/WP/ [Accessed 23 May 2015].

AGAR, B., EREN, M. & CINAR, A. 2005. Glucosim: educational software for virtual experiments with patients with type 1 diabetes. *Conf Proc IEEE Eng Med Biol Soc,* 1**,** 845-8.

AIDA. 2000. *AIDA - A freeware educational simulator program of glucose-insulin interaction and insulin dosage & dietary adjustment in diabetes mellitus* [Online]. Available: http://www.2aida.net/aida/index.shtml [Accessed 14 March 2015].

AIDA LOGSTATS. 2000. *AIDA Logstats* [Online]. Available: http://www.2aida.net/aida/logstats.htm [Accessed 14 March 2015].

ALBERTI, K. G. & ZIMMET, P. Z. 1998. Definition, diagnosis and classification of diabetes mellitus and its complications. Part 1: diagnosis and classification of diabetes mellitus provisional report of a WHO consultation. *Diabet Med,* 15**,** 539-53.

AMERICAN DIABETES ASSOCIATION. 2013a. *Blood Glucose Control and Exercise* [Online]. Available: http://www.diabetes.org/food-and-fitness/fitness/get-started-safely/blood-glucose-control-and-exercise.html [Accessed 1 April 2015].

AMERICAN DIABETES ASSOCIATION. 2013b. *Dawn Phenomenon* [Online]. Available: http://www.diabetes.org/living-with-diabetes/treatment-and-care/blood-glucose-control/dawn-phenomenon.html?referrer=https://www.google.no/ [Accessed 4 June 2015].

AMERICAN DIABETES ASSOCIATION. 2014. *Factors Affecting Blood Glucose* [Online]. Available: http://www.diabetes.org/living-with-diabetes/treatment-and-care/blood-glucose-control/factors-affecting-blood-glucose.html [Accessed 1 April 2015].

AMERICAN DIABETES ASSOCIATION. 2015. *Insulin Pumps* [Online]. Available: http://www.diabetes.org/living-with-diabetes/treatment-and-care/medication/insulin/insulin-pumps.html?referrer=https://www.google.no/ [Accessed 22 June 2015].

AMERICAN SOCIETY OF HEMATOLOGY. 2015. *Blood Basics* [Online]. Available: http://www.hematology.org/Patients/Basics/ [Accessed 25 May 2015].

ANDERSEN, G., ALLUIS, B., MEIFFREN, G., RANSON, A., SOULA, O., SOULA, G., SOULA, R., FISCHER, A., NOSEK, L., SCHLIESS, F. & HEISE, T. 2014. The ultra-rapid biochaperone insulin lispro shows a faster onset of action and stronger early metabolic effect than insulin lispro alone. *Diabetologia,* 57**,** S381-S381.

ANDROID DEVELOPMENT COMMUNITY. 2007. *Android - System Architecture (In Words)* [Online]. http://www.anddev.org/: AndroidDevelopmentCommunity. Available: http://www.anddev.org/open-news-f1/android-system-architecture-in-words-t7.html [Accessed 11 April 2015].

AOKI, N., OHTA, S., OKADA, T., OISHI, M. & FUKUI, T. 2005. INSULOT: a cellular phone-based edutainment learning tool for children with type 1 diabetes. *Diabetes care,* 28**,** 760.

ATKINSON, F. S., FOSTER-POWELL, K. & BRAND-MILLER, J. C. 2008. International tables of glycemic index and glycemic load values: 2008. *Diabetes Care,* 31**,** 2281-3.

BACCHI, E., NEGRI, C., ZANOLIN, M. E., MILANESE, C., FACCIOLI, N., TROMBETTA, M., ZOPPINI, G., CEVESE, A., BONADONNA, R. C., SCHENA, F., BONORA, E., LANZA, M. & MOGHETTI, P. 2012. Metabolic Effects of Aerobic Training and Resistance Training in Type 2 Diabetic Subjects A randomized controlled trial (the RAED2 study). *Diabetes Care,* 35**,** 676-682.

BARANOWSKI, T., BUDAY, R., THOMPSON, D., LYONS, E. J., LU, A. S. & BARANOWSKI, J. 2013. Developing Games for Health Behavior Change: Getting Started. *Games For Health Journal,* 2**,** 183-190.

BARRETT, P. H. R., BELL, B. M., COBELLI, C., GOLDE, H., SCHUMITZKY, A., VICINI, P. & FOSTER, D. M. 1998. SAAM II: Simulation, Analysis, and Modeling Software for tracer and pharmacokinetic studies. *Metabolism-Clinical and Experimental,* 47**,** 484-492.

BASSILIOUS, E., DECHAMPLAIN, A., MCCABE, I., STEPHAN, M., KAPRALOS, B., MAHMUD, F. & DUBROWSKI, A. Power defense: A video game for improving diabetes numeracy. Games Innovation Conference (IGIC), 2011 IEEE International, 2-3 Nov. 2011 2011. 124-125.

BERG, J. M., TYMOCZKO, J. L. & STRYER, L. 2002. *Biochemistry, 5th Edition. Section 30.2. Each Organ Has a Unique Metabolic Profile.* [Online]. W. H. Freeman. Available: http://www.ncbi.nlm.nih.gov/books/NBK22436/.

BERG, K. 2010. *Exercise Often Raises Blood Glucose in Type 1 Diabetes* [Online]. Available: http://www.diabeteshealth.com/news/exercise-often-raises-blood-glucose-in-type-1-diabetes/ [Accessed 9 April 2015].

BERGMAN, R. N. 2002. Pathogenesis and prediction of diabetes mellitus: lessons from integrative physiology. *Mt Sinai J Med,* 69**,** 280-90.

BERGMAN, R. N., IDER, Y. Z., BOWDEN, C. R. & COBELLI, C. 1979. Quantitative estimation of insulin sensitivity. *Am J Physiol,* 236**,** E667-77.

BERGMAN, R. N., PHILLIPS, L. S. & COBELLI, C. 1981. Physiologic Evaluation of Factors Controlling Glucose-Tolerance in Man - Measurement of Insulin Sensitivity and Beta-Cell Glucose Sensitivity from the Response to Intravenous Glucose. *Journal of Clinical Investigation,* 68**,** 1456-1467.

BERMAN, M., SHAHN, E. & WEISS, M. F. 1962. The routine fitting of kinetic data to models: a mathematical formalism for digital computers. *Biophys J,* 2**,** 275-87.

BERMAN, M., WEISS, M. F. & NATIONAL INSTITUTE OF ARTHRITIS AND METABOLIC DISEASES (U.S.) 1967. *Users manual for SAAM (simulation, analysis, and modeling),* Bethesda, Md.,, National Institute of Arthritis and Metabolic Diseases; for sale by the Supt. of Docs., U. S. Govt. Print. Off.

BERNSTEIN, R. K. 2007. *Dr. Bernstein's diabetes solution : the complete guide to achieving normal blood sugars,* New York, Little, Brown and Co.

BIERMANN, E. & MEHNERT, H. 1990. DIABLOG: a simulation program of insulin-glucose dynamics for education of diabetics. *Computer Methods and Programs in Biomedicine,* 32**,** 311-318.

BOLIE, V. W. 1961. Coefficients of normal blood glucose regulation. *J Appl Physiol,* 16**,** 783-8.

BORNEMISZA, P. & SUCIU, I. 1980. Effect of cigarette smoking on the blood glucose level in normals and diabetics. *Med Interne,* 18**,** 353-6.

BOSTON, R. C., GREIF, P. C. & BERMAN, M. 1981. Conversational Saam - an Interactive Program for Kinetic-Analysis of Biological-Systems. *Computer Programs in Biomedicine,* 13**,** 111-119.

BOUNDLESS. 2014. *Fluid Compartments. Boundless Anatomy and Physiology.* [Online]. Boundless.com. Available: https://www.boundless.com/physiology/textbooks/boundless-anatomy-and-physiology-textbook/body-fluids-and-acid-base-balance-26/body-fluids-246/fluid-compartments-1208-1206/ [Accessed 2015 25 May].

BOUTAYEB, A. & CHETOUANI, A. 2006. A critical review of mathematical models and data used in diabetology. *Biomedical Engineering Online,* 5.

BRADY, P. 2008. Android Anatomy and Physiology. *Google I/O Conference.*

BREAKINGMUSCLE.COM. 2013. *60 MINUTES OF EXERCISE CAN IMPROVE INSULIN RESISTANCE 25%* [Online]. BreakingMuscle.com. Available: http://breakingmuscle.com/health-medicine/60-minutes-of-exercise-can-improve-insulin-resistance-25 [Accessed 6 April 2015].

BRIEGEL, T. & TRESP, V. 2002. A Nonlinear State Space Model for the Blood Glucose Metabolism of a Diabetic (Ein nichtlineares Zustandsraummodell für den Blutglukosemetabolismus eines Diabetikers). *At-automatisierungstechnik,* 50.

BURNETTE, E. 2010. *Hello, Android : introducing Google's mobile development platform,* Raleigh, N.C., Pragmatic Bookshelf.

BUTLER, M. 2011. Android: Changing the Mobile Landscape. *Pervasive Computing, IEEE,* 10**,** 4-7.

CAMPBELL, P. J., BOLLI, G. B., CRYER, P. E. & GERICH, J. E. 1985a. Pathogenesis of the dawn phenomenon in patients with insulin-dependent diabetes mellitus. Accelerated glucose production and impaired glucose utilization due to nocturnal surges in growth hormone secretion. *N Engl J Med,* 312**,** 1473-9.

CAMPBELL, P. J., BOLLI, G. B., CRYER, P. E. & GERICH, J. E. 1985b. Sequence of events during development of the dawn phenomenon in insulin-dependent diabetes mellitus. *Metabolism,* 34**,** 1100-4.

CAROLYN, W. Extending the Use of Games in Health Care. *In:* SAGEEV, O., MICHAEL, S., AZZA, A., ANTHONY, C., MELANIE, K., HADI, K., FENGAN, L. & ANTHONY, O., eds., 2006. 88b-88b.

CENGIZ, E., TAMBORLANE, W. V., MARTIN-FREDERICKSEN, M., DZIURA, J. & WEINZIMER, S. A. 2010. Early pharmacokinetic and pharmacodynamic effects of mixing lispro with glargine insulin: results of glucose clamp studies in youth with type 1 diabetes. *Diabetes Care,* 33**,** 1009-12.

CHOMUTARE, T., FERNANDEZ-LUQUE, L., ÅRSAND, E. & HARTVIGSEN, G. 2011. Features of Mobile Diabetes Applications: Review of the Literature and Analysis of Current Applications Compared Against Evidence-Based Guidelines. *J Med Internet Res,* 13**,** e65.

CIVILIZED SOFTWARE. *MLAB* [Online]. CivilizedSoftware. Available: http://www.civilized.com [Accessed 22 March 2015].

CIVILIZED SOFTWARE. *MLAB: Minimal Models for Glucose and Insulin Kinetics* [Online]. CivilizedSoftware. Available: http://www.civilized.com/mlabexamples/glucose.htmld/ [Accessed 22 March 2015].

CONVERTUNITS.COM. 2015. *Glucose molecular weight* [Online]. ConvertUnits.com. Available: http://www.convertunits.com/molarmass/Glucose [Accessed 25 May 2015].

DBAZA. 2012. *dbaza's Diabetes Education for Kids (2003)* [Online]. Available: http://www.dbaza.com/dek/index.html [Accessed 23 March 2015].

DE GAETANO, A. & ARINO, O. 2000. Mathematical modelling of the intravenous glucose tolerance test. *J Math Biol,* 40**,** 136-68.

DEROUICH, M. & BOUTAYEB, A. 2002. The effect of physical exercise on the dynamics of glucose and insulin. *Journal of Biomechanics,* 35**,** 911-917.

DEVELOPER.ANDROID.COM. 2015a. *Android Developer Tools* [Online]. developer.android.com. Available: https://developer.android.com/tools/help/adt.html [Accessed 11 April 2015].

DEVELOPER.ANDROID.COM. 2015b. *Android Studio Overview* [Online]. developer.android.com. Available: http://developer.android.com/tools/studio/index.html [Accessed 21 June 2015].

DEVELOPER.ANDROID.COM. 2015c. *Notifications* [Online]. developer.android.com. Available: http://developer.android.com/guide/topics/ui/notifiers/notifications.html [Accessed 24 June 2015].

DEVELOPER.ANDROID.COM. 2015d. *Services* [Online]. developer.android.com. Available: http://developer.android.com/guide/components/services.html [Accessed 24 June 2015].

DIABETAPEDIA. 2014. *mg/dL* [Online]. Available: http://www.diabetapedia.com/mgdl/ [Accessed 5 March 2015].

DIABETES EDUCATION ONLINE & UCSF. 2015a. *Blood Sugar & Other Hormones* [Online]. Diabetes Teaching Center at the University of California, San Francisco (UCSF). Available: http://dtc.ucsf.edu/types-of-diabetes/type1/understanding-type-1-diabetes/how-the-body-processes-sugar/blood-sugar-other-hormones/ [Accessed 26 May 2015].

DIABETES EDUCATION ONLINE & UCSF. 2015b. *Types of Insulin* [Online]. Diabetes Teaching Center at the University of California, San Francisco (UCSF). Available: http://dtc.ucsf.edu/types-of-diabetes/type2/treatment-of-type-2-diabetes/medications-and-therapies/type-2-insulin-rx/types-of-insulin/ [Accessed 31 March 2015].

DIABETES HEALTH. 1992. *Captain Novolin* [Online]. Available: http://www.diabeteshealth.com/business-article-archive/captain-novolin/ [Accessed 23 March 2015].

DIABETES-SUPPORT.ORG.UK. 2015. *Insulin profiles* [Online]. Diabetes-Support.org.uk. Available: http://www.diabetes-support.org.uk/info/?page_id=408 [Accessed 7 April 2015].

DIABETES.CO.UK. 2015a. *Basal Bolus - Basal Bolus Injection Regimen* [Online]. Available: http://www.diabetes.co.uk/insulin/basal-bolus.html [Accessed 31 March 2015].

DIABETES.CO.UK. 2015b. *Blood Glucose Diaries* [Online]. Available: http://www.diabetes.co.uk/blood-glucose/blood-glucose-monitoring-diaries.html [Accessed 5 July 2015].

DIABETES.CO.UK. 2015c. *Blood Sugar Level Ranges* [Online]. Available: http://www.diabetes.co.uk/diabetes_care/blood-sugar-level-ranges.html [Accessed 31 March 2015].

DIABETES.CO.UK. 2015d. *Dawn Phenomenon (Liver Dump)* [Online]. Available: http://www.diabetes.co.uk/blood-glucose/dawn-phenomenon.html [Accessed 4 June 2015].

DIABETES.CO.UK. 2015e. *Glucagon* [Online]. Diabetes.co.uk. Available: http://www.diabetes.co.uk/body/glucagon.html [Accessed 26 May 2015].

DIABETES.CO.UK. 2015f. *What Affects Blood Sugar Levels* [Online]. Available: http://www.diabetes.co.uk/blood-glucose/what-affects-blood-glucose-levels.html [Accessed 01 April 2015].

DIABETESFORBUNDET. 2008. *Diabetes historisk* [Online]. Available: http://www.diabetes.no/no/Om_forbundet/Diabetes_historisk/ [Accessed 3 March 2015].

DIABETICTALK & NUTRIHAND. 2008. *Dawn Phenomenon and the Somogyi Effect* [Online]. Available: https://www.nutrihand.com/Nutrihand/showHealthCenterArticle.do?article=Medical%2F20070416.Dawn+Phenomenon+and+the+Somogyi+Effect [Accessed 4 June 2015].

DIAPEDIA. 2014. *Brain glucose metabolism* [Online]. Diapedia.org. Available: http://www.diapedia.org/metabolism/brain-glucose-metabolism [Accessed 26 May 2015].

DIEHL, L. A., LEHMANN, E., SOUZA, R. M., ALVES, J. B., ESTEVES, R. Z. & GORDAN, P. A. A serious game prototype for education of medical doctors and students on insulin management for treatment of diabetes mellitus. Serious Games and Applications for Health (SeGAH), 2011 IEEE 1st International Conference on, 16-18 Nov. 2011 2011. 1-4.

DINESEN, B. & ANDERSEN, P. E. 2006. Qualitative evaluation of a diabetes advisory system, DiasNet. *J Telemed Telecare,* 12**,** 71-4.

DINESEN, B., ANDERSEN, P. E. R. & FRANK, J. 2004. Effect evaluation of DiasNet - the Digital Hospital. Vendsyssel Hospital, Frederikshavn, Denmark. Aarhus: University of Aarhus, Aarhus School of Business, Department of Business Studies.

DONGA, E., VAN DIJK, M., HOOGMA, R. P. L. M., CORSSMIT, E. P. M. & ROMIJN, J. A. 2013. Insulin resistance in multiple tissues in patients with type 1 diabetes mellitus on long-term continuous subcutaneous insulin infusion therapy. *Diabetes/Metabolism Research and Reviews,* 29**,** 33-38.

DRUGS.COM. 2002. *Insulin lispro protamine, human* [Online]. Drugs.com. Available: http://www.drugs.com/drp/insulin-lispro-protamine-human.html [Accessed 07 April 2015].

ECLIPSE FOUNDATION. 2015. *About the Eclipse Foundation* [Online]. eclipse.org. Available: http://www.eclipse.org/org [Accessed 11 April 2015].

ERZEN, F. C., BIROL, G. & CINAR, A. 2001. Glucosim: A simulator for education on the dynamics of Diabetes mellitus. *Proceedings of the 23rd Annual International Conference of the Ieee Engineering in Medicine and Biology Society, Vols 1-4,* 23**,** 3163-3166.

EWINGS, S. M., SAHU, S. K., VALLETTA, J. J., BYRNE, C. D. & CHIPPERFIELD, A. J. 2014. A Bayesian network for modelling blood glucose concentration and exercise in type 1 diabetes. *Stat Methods Med Res*.

FARLEX, I. 2003. *The American Diabetes Association Published Efficacy Data on `dbaza's Diabetes Education for Kids' CD-ROM at the 63rd Scientific Sessions June 13-17 in New Orleans.* [Online]. TheFreeLibrary.com. Available: http://www.thefreelibrary.com/The+American+Diabetes+Association+Published+Efficacy+Data+on+%60dbaza's...-a0103299824 [Accessed 23 March 2015].

FARLEX, I. 2015a. *Extracellular Fluid* [Online]. http://medical-dictionary.thefreedictionary.com. Available: http://medical-dictionary.thefreedictionary.com/extracellular+fluid [Accessed 25 May 2015].

FARLEX, I. 2015b. *Interstitial Fluid* [Online]. http://medical-dictionary.thefreedictionary.com. Available: http://medical-dictionary.thefreedictionary.com/interstitial+fluid [Accessed 25 May 2015].

FRECKMANN, G., JOVANOVIC, L., BAUMSTARK, A., HAUG, C. & VAN DER HELM, W. 2008. The circadian study: the get-up phenomenon in type 1 diabetes. *Diabetes Care,* 31**,** e85.

FUCHSLOCHER, A., NIESENHAUS, J. & KRäMER, N. 2011. Serious games for health: An empirical study of the game " Balance" for teenagers with diabetes mellitus. *Entertainment Computing,* 2**,** 97-101.

GAMEMETRIX. 2013a. *DiabetesGame: Triviality* [Online]. Available: http://www.diabetesgame.com [Accessed 23 March 2015].

GAMEMETRIX. 2013b. *GameMetrix: About us* [Online]. Available: http://www.gamemetrixsolutions.com/about_us.html [Accessed 23 March 2015].

GENTRY, C. J. 1991. Reviews : BG Pilot (computer game), manual by Todd Ramming (1989).

GIRARD, C., ECALLE, J. & MAGNAN, A. 2013. Serious games as new educational tools: how effective are they? A meta-analysis of recent studies. *Journal of Computer Assisted Learning,* 29**,** 207-219.

GLUCOSIM. *GlucoSim WebPage* [Online]. Department of Chemical and Biological Engineering, Ilinois Institute of Technology: Department of Chemical and Biological Engineering, Ilinois Institute of Technology. Available: http://simulator.iit.edu/web/glucosim/index.html [Accessed 22 March 2015].

GLYCEMICINDEX.COM. 2014. *Frequently Asked Questions* [Online]. Available: http://www.glycemicindex.com/faqsList.php#1 [Accessed 24 May 2015].

GLYCEMICINDEX.COM & INTERNATIONAL GI DATABASE. 2014. *About Glycemic Index* [Online]. GlycemicIndex.com: University of Sydney. Available: http://www.glycemicindex.com/about.php [Accessed 7 April 2015].

GOMEZ AGUILERA, E. J., ARREDONDO, M. T., ZOREDA, J. L. & DELPOZO, F. 1989. A Simulator of Therapies for Education in Diabetes. *Images of the Twenty-First Century, Pts 1-6,* 11**,** 1965-1966.

GOMEZ, E. J., DELPOZO, F. & HERNANDO, M. E. 1996. Telemedicine for diabetes care: The DIABTel approach towards diabetes telecare. *Medical Informatics,* 21**,** 283-295.

GUILL, V. & ALESON-CARBONELL, M. 2012. Serious games and learning effectiveness: The case of It's a Deal! *Comput. Educ.*, 58**,** 435-448.

HARVARD HEALTH PUBLICATIONS. 2015. *Glycemic index and glycemic load for 100+ foods* [Online]. health.harvard.edu: Harvard University. Available: http://www.health.harvard.edu/healthy-eating/glycemic_index_and_glycemic_load_for_100_foods [Accessed 9 July 2015].

HAYES, S. 1994. Captain Novocare™ — the adventure game that teaches children about diabetes. *Patient Education and Counseling,* 23, Supplement 1**,** S133.

HEE-YEON, C., CHOON-SUNG, N. & DONG-RYEOL, S. A compariosn of open and closed mobile platforms. Electronics and Information Engineering (ICEIE), 2010 International Conference On, 1-3 Aug. 2010 2010. V2-141-V2-143.

HEISE, T., ECKERS, U., KANC, K., NIELSEN, J. N. & NOSEK, L. 2008. The Pharmacokinetic and Pharmacodynamic Properties of Different Formulations of Biphasic Insulin Aspart: A Randomized, Glucose Clamp, Crossover Study. *Diabetes Technology & Therapeutics,* 10**,** 479-485.

HEJLESEN, O. K. 1998. *DIAS - the Diabetes Advisory System: technical and physiological aspects of the system and evaluation results obtained so far,* Aalborg University. Medical Informatics Group.

HEJLESEN, O. K., ANDREASSEN, S., HOVORKA, R. & CAVAN, D. A. 1997. DIAS--the diabetes advisory system: an outline of the system and the evaluation results obtained so far. *Comput Methods Programs Biomed,* 54**,** 49-58.

HEJLESEN, O. K., PLOUGMANN, S. & CAVAN, D. A. 2000. DiasNet - an Internet tool for communication and education in diabetes. *Stud Health Technol Inform,* 77**,** 563-7.

HELSEDIREKTORATET 2009. Diabetes : forebygging, diagnostikk og behandling. Oslo: Helsedirektoratet.
HESS-FISCHL, A. 2015. *What is Insulin?* [Online]. EndocrineWeb. Available: http://www.endocrineweb.com/conditions/type-1-diabetes/what-insulin [Accessed 31 March 2015].
HIROTA, K., ISHIHARA, H., TSUBO, T. & MATSUKI, A. 1999. Estimation of the initial distribution volume of glucose by an incremental plasma glucose level at 3 min after i.v. glucose in humans. *British Journal of Clinical Pharmacology,* 47**,** 361-364.
HUCKVALE, K., ADOMAVICIUTE, S., PRIETO, J., LEOW, M.-S. & CAR, J. 2015. Smartphone apps for calculating insulin dose: a systematic assessment. *BMC Medicine,* 13**,** 1-10.
HUGHES, D. S. & NARENDRAN, P. 2014. Alpha cell function in type 1 diabetes. *Br J Diabetes Vasc Dis,* 14**,** 45.
HUME, R. & WEYERS, E. 1971. Relationship between total body water and surface area in normal and obese subjects. *Journal of Clinical Pathology,* 24**,** 234-238.
IBM DEVELOPERWORKS. 2011. *Introduction to Android development* [Online]. http://www.ibm.com/. Available: http://www.ibm.com/developerworks/library/os-android-devel/ [Accessed 11 April 2015].
ISHIHARA, H. & GIESECKE, A. H. 2007. *Fluid Volume Monitoring with Glucose Dilution,* Springer Japan.
JAVA.COM. 2015. *What is Java technology and why do I need it?* [Online]. Java.com. Available: https://www.java.com/en/download/faq/whatis_java.xml [Accessed 11 April 2015].
JAVATALK. 2014. *What is Java, history of Java, how it all began* [Online]. Javatalk.org. Available: http://www.javatalk.org/2013/11/JAVA-BRIEF-HISTORY-VERSION-HISTORY.html [Accessed 11 April 2015].
KLONOFF, D. C. 2012. The current status of bolus calculator decision-support software. *J Diabetes Sci Technol,* 6**,** 990-4.
KURODA, A., YASUDA, T., TAKAHARA, M., SAKAMOTO, F., KASAMI, R., MIYASHITA, K., YOSHIDA, S., KONDO, E., AIHARA, K., ENDO, I., MATSUOKA, T. A., KANETO, H., MATSUMOTO, T., SHIMOMURA, I. & MATSUHISA, M. 2012. Carbohydrate-to-insulin ratio is estimated from 300-400 divided by total daily insulin dose in type 1 diabetes patients who use the insulin pump. *Diabetes Technol Ther,* 14**,** 1077-80.
LANCASTER, R. J. 2014. Serious Game Simulation as a Teaching Strategy in Pharmacology. *Clinical Simulation In Nursing,* 10**,** e129-e137.
LANDT, K. W., CAMPAIGNE, B. N., JAMES, F. W. & SPERLING, M. A. 1985. Effects of Exercise Training on Insulin Sensitivity in Adolescents with Type-I Diabetes. *Diabetes Care,* 8**,** 461-465.
LEHMANN, E. D. 1997. Interactive educational simulators in diabetes care. *Med Inform (Lond),* 22**,** 47-76.
LEHMANN, E. D. 1998. AIDA - a computer-based interactive educational diabetes simulator. *Diabetes Educ,* 24**,** 341-6, 348.
LEHMANN, E. D. & DEUTSCH, T. 1991. A Physiological Model of Glucose-Insulin Interaction. *Proceedings of the Annual International Conference of the Ieee Engineering in Medicine and Biology Society, Vol 13, Pts 1-5***,** 2274-2275.
LEHMANN, E. D., DEUTSCH T FAU - CARSON, E. R., CARSON ER FAU - SONKSEN, P. H. & SONKSEN, P. H. 1994. Combining rule-based reasoning and mathematical modelling in diabetes care.

LEHMANN, E. D. & TATTI, P. 2002. Using the AIDA--www.2aida.org--diabetes simulator. Part 2: recommended training requirements for health-carers planning to teach with the software. *Diabetes Technol Ther,* 4**,** 717-32.

LEPORE, M., PAMPANELLI, S., FANELLI, C., PORCELLATI, F., BARTOCCI, L., DI VINCENZO, A., CORDONI, C., COSTA, E., BRUNETTI, P. & BOLLI, G. B. 2000. Pharmacokinetics and pharmacodynamics of subcutaneous injection of long-acting human insulin analog glargine, NPH insulin, and ultralente human insulin and continuous subcutaneous infusion of insulin lispro. *Diabetes,* 49**,** 2142-8.

LI, J., KUANG, Y. & LI, B. 2001. Analysis of IVGTT glucose-insulin interaction models with time delay. *Discrete and Continuous Dynamical Systems Series B,* 1**,** 103-124.

LIEBERMAN, D. A. 2012. Video games for diabetes self-management: examples and design strategies. *J Diabetes Sci Technol,* 6**,** 802-6.

MAGKOS, F., MOHAMMED, B. S. & MITTENDORFER, B. 2010. Enhanced insulin sensitivity after acute exercise is not associated with changes in high-molecular weight adiponectin concentration in plasma. *Eur J Endocrinol,* 162**,** 61-6.

MAGKOS, F. & SIDOSSIS, L. S. 2008. Exercise and Insulin Sensitivity-Where Do We Stand? You'd Better Run! *US Endocrinology,* 4**,** 23-26.

MAKHLYSHEVA, A. 2013. *A mobile phone-based serious game for children with type 1 diabetes.* Universitetet i Tromsø.

MAKROGLOU, A., LI, J. X. & KUANG, Y. 2005. Mathematical models and software tools for the glucose-insulin regulatory system and diabetes: an overview. *Applied Numerical Mathematics,* 56**,** 559-573.

MARKAKIS, M. G., MITSIS, G. D. & MARMARELIS, V. Z. 2008. Computational study of an augmented minimal model for glycaemia control. *Conf Proc IEEE Eng Med Biol Soc,* 2008**,** 5445-8.

MCAULEY, D. 2015. *Long-Acting Insulins* [Online]. GlobalRPh.com. Available: http://www.globalrph.com/long-acting-insulins.htm#Glargine [Accessed 07 April 2015].

MCCARREN, M. 2015a. *Checking Blood Sugar: Blood Glucose Meter Accuracy, Page 1* [Online]. diabeticlivingonline.com. Available: http://www.diabeticlivingonline.com/monitoring/blood-sugar/checking-blood-sugar-blood-glucose-meter-accuracy [Accessed 6 June 2015].

MCCARREN, M. 2015b. *Checking Blood Sugar: Blood Glucose Meter Accuracy, Page 2* [Online]. diabeticlivingonline.com. Available: http://www.diabeticlivingonline.com/monitoring/blood-sugar/checking-blood-sugar-blood-glucose-meter-accuracy?page=0%2C1 [Accessed 6 June 2015].

MEDICINENET. 2014a. *Hyperglycemia* [Online]. Available: http://www.medicinenet.com/hyperglycemia/article.htm [Accessed 31 March 2015].

MEDICINENET. 2014b. *Hypoglycemia* [Online]. Available: http://www.medicinenet.com/hypoglycemia/article.htm [Accessed 31 March 2015].

MEDICINENET, M. C. S., WILLIAM C. SHIEL JR. 2014c. *Diabetes Mellitus* [Online]. Available: http://www.medicinenet.com/diabetes_mellitus/page2.htm#what_is_diabetes [Accessed 2 March 2015].

MELLITS, E. D. & CHEEK, D. B. 1970. The assessment of body water and fatness from infancy to adulthood. *Monogr Soc Res Child Dev,* 35**,** 12-26.

MENDOSA, D. 2008. *Revised International Table of Glycemic Index (GI) and Glycemic Load (GL) Values - 2008* [Online]. mendosa.com. Available: http://www.mendosa.com/gilists.htm [Accessed 9 July 2015].

MERGENTHALER, P., LINDAUER, U., DIENEL, G. A. & MEISEL, A. 2013. Sugar for the brain: the role of glucose in physiological and pathological brain function. *Trends in neurosciences,* 36**,** 587-597.

MICHEL, H. 2013. Cognitive Maps of Serious Games: An Exploratory Approach of Learners' Representations. *In:* MA, M., OLIVEIRA, M., PETERSEN, S. & HAUGE, J. (eds.) *Serious Games Development and Applications.* Springer Berlin Heidelberg.

NADLER, S. B., HIDALGO, J. H. & BLOCH, T. 1962. Prediction of blood volume in normal human adults. *Surgery,* 51**,** 224-32.

NASJONALT SENTER FOR SAMHANDLING OG TELEMEDISIN. 2015a. *Diabetesdagboka* [Online]. Available: http://telemed.custompublish.com/better-quality-of-life-for-patients-with-diabetes.5208977-287081.html [Accessed 11 June 2015].

NASJONALT SENTER FOR SAMHANDLING OG TELEMEDISIN. 2015b. *Nasjonalt Senter For Samhandling Og Telemedisin (NST)* [Online]. Available: http://www.telemed.no [Accessed 11 June 2015].

NIH. 2015. *National Institutes of Health (NIH)* [Online]. Available: http://www.nih.gov [Accessed 22 March 2015].

NOBELPRIZE.ORG. 2009. *Facts about Diabetes and Insulin* [Online]. Available: http://www.nobelprize.org/educational/medicine/insulin/diabetes-insulin.html [Accessed 31 March 2015].

NOVO NORDISK. 2015. *Levemir* [Online]. http://www.drugs.com. Available: http://www.drugs.com/pro/levemir.html [Accessed 23 May 2015].

NOVOTNY, J. A., GREIF, P. & BOSTON, R. C. 2003. WinSAAM: Application and explanation of use. *Mathematical Modeling in Nutrition and the Health Sciences,* 537**,** 343-351.

ORACLE.COM. 2011a. *The History of Java Technology* [Online]. Oracle.com. Available: http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html [Accessed 11 April 2015].

ORACLE.COM. 2011b. *Java Timeline* [Online]. Oracle.com. Available: http://oracle.com.edgesuite.net/timeline/java/ [Accessed 11 April 2015].

PAI, A. 2015. *Insulin dose calculator apps may actively contribute to inaccurate dose recommendations* [Online]. http://mobihealthnews.com: MobiHealthNews. Available: http://mobihealthnews.com/43429/insulin-dose-calculator-apps-may-actively-contribute-to-inaccurate-dose-recommendations/?utm_content=buffer62cb0&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer [Accessed 10 June 2015].

PAWLAN, M. 1999. *Essentials, Part 1, Lesson 1: Compiling & Running a Simple Program* [Online]. Oracle.com. Available: http://www.oracle.com/technetwork/java/compile-136656.html [Accessed 11 April 2015].

PEREIRA, G., BRISSON, A., PRADA, R., PAIVA, A., BELLOTTI, F., KRAVCIK, M. & KLAMMA, R. 2012. Serious Games for Personal and Social Learning & Ethics: Status and Trends. *Procedia Computer Science,* 15**,** 53-65.

PERRIELLO, G., DE FEO, P., TORLONE, E., FANELLI, C., SANTEUSANIO, F., BRUNETTI, P. & BOLLI, G. B. 1990. Nocturnal spikes of growth hormone secretion cause the dawn phenomenon in type 1 (insulin-dependent) diabetes mellitus by decreasing hepatic (and extrahepatic) sensitivity to insulin in the absence of insulin waning. *Diabetologia,* 33**,** 52-9.

PERRIELLO, G., DE FEO, P., TORLONE, E., FANELLI, C., SANTEUSANIO, F., BRUNETTI, P. & BOLLI, G. B. 1991. The dawn phenomenon in type 1 (insulin-

dependent) diabetes mellitus: magnitude, frequency, variability, and dependency on glucose counterregulation and insulin sensitivity. *Diabetologia,* 34**,** 21-8.

PERRY, J. S. 2010. *Introduction to Java programming, Part 1: Java language basics* [Online]. http://www.ibm.com/. Available: http://www.ibm.com/developerworks/java/tutorials/j-introtojava1/ [Accessed 11 April 2015].

PHARMACORAMA.COM. 2015. *Insulin, chemical structure and metabolism* [Online]. Pharmacorama.com. Available: http://www.pharmacorama.com/en/Sections/Insulin_1.php [Accessed 25 May 2015].

PLANK, J., BODENLENZ, M., SINNER, F., MAGNES, C., GORZER, E., REGITTNIG, W., ENDAHL, L. A., DRAEGER, E., ZDRAVKOVIC, M. & PIEBER, T. R. 2005. A double-blind, randomized, dose-response study investigating the pharmacodynamic and pharmacokinetic properties of the long-acting insulin analog detemir. *Diabetes Care,* 28**,** 1107-12.

PLOUGMANN, S., HEJLESEN, O. K. & CAVAN, D. A. 2001. DiasNet - a diabetes advisory system for communication and education via the internet. *Int J Med Inform,* 64**,** 319-30.

QUESADA, I., TUDURI, E., RIPOLL, C. & NADAL, A. 2008. Physiology of the pancreatic alpha-cell and glucagon secretion: role in glucose homeostasis and diabetes. *J Endocrinol,* 199**,** 5-19.

RAVE, K., HEISE, T., PFUTZNER, A., HEINEMANN, L. & SAWICKI, P. T. 2001. Impact of diabetic nephropathy on pharmacodynamic and pharmacokinetic properties of insulin in type 1 diabetic patients. *Diabetes Care,* 24**,** 886-890.

ROY, A. & PARKER, R. S. 2007. Dynamic modeling of exercise effects on plasma glucose and insulin levels. *J Diabetes Sci Technol,* 1**,** 338-47.

SANOFI. 2013. *Apidra 100 Units/ml, solution for injection in a vial* [Online]. http://www.medicines.org.uk. Available: http://www.medicines.org.uk/emc/medicine/26478/SPC/Apidra+100+Units+ml,+solution+for+injection+in+a+vial [Accessed 23 May 2015].

SANOFI. 2015. *LANTUS® Prescribing Information* [Online]. Available: http://products.sanofi.us/lantus/lantus.html#S12.2 [Accessed 2015 23 May].

SANOFI US. 2015a. *Lantus: Lantus vs Detemir* [Online]. Available: http://www.lantus.com/hcp/about-lantus/vs-detemir [Accessed 7 April 2015].

SANOFI US. 2015b. *Lantus® is Approved for Use Once a Day* [Online]. Available: http://www.lantus.com/HCP/about-lantus/vs-nph [Accessed 23 May 2015].

SCHEINER, G. 2006. Sports & fitness. The great blood glucose balancing act. *Diabetes Self Management,* 23**,** 48-52.

SCHEINER, G. 2014. *Strike the Spike II* [Online]. DiabetesSelfManagement.com: Diabetes Self-Management. Available: http://www.diabetesselfmanagement.com/managing-diabetes/blood-glucose-management/strike-the-spike-ii/ [Accessed 7 April 2015].

SCHNELL, O., ERBACH, M. & WINTERGERST, E. 2013. Higher Accuracy of Self-Monitoring of Blood Glucose in Insulin-Treated Patients in Germany: Clinical and Economical Aspects. *Journal of Diabetes Science and Technology,* 7**,** 904-912.

SHARAR, S. R., CARROUGHER, G. J., NAKAMURA, D., HOFFMAN, H. G., BLOUGH, D. K. & PATTERSON, D. R. 2007. Factors influencing the efficacy of virtual reality distraction analgesia during postburn physical therapy: preliminary results from 3 ongoing studies. *Arch Phys Med Rehabil,* 88**,** S43-9.

SOCIALCOMPARE.COM. 2015. *Android versions comparison* [Online]. SocialCompare.com. Available: http://socialcompare.com/en/comparison/android-versions-comparison [Accessed 11 April 2015].

SOURCE.ANDROID.COM. 2015. *Welcome to the Android Open Source Project!* [Online]. source.android.com. Available: http://source.android.com/ [Accessed 11 April 2015].

SPERO, D. & DIABETES SELF MANAGEMENT. 2013. *Controlling the Dawn Phenomenon* [Online]. www.diabetesselfmanagement.com/. Available: http://www.diabetesselfmanagement.com/blog/controlling-the-dawn-phenomenon/ [Accessed 4 June 2015].

STACKOVERFLOW.COM. 2015. *Stack Overflow is a question and answer site for professional and enthusiast programmers.* [Online]. Available: http://stackoverflow.com [Accessed 6 June 2015].

STEDMAN, T. L. 2012. *Stedman's Medical Dictionary for the Health Professions and Nursing*, Wolters Kluwer Health/Lippincott Williams & Wilkins.

STEFANOVSKI, D., MOATE, P. J. & BOSTON, R. C. 2003. WinSAAM: A windows-based compartmental modeling system. *Metabolism-Clinical and Experimental,* 52**,** 1153-1166.

STRAIGHT HEALTHCARE. 2014. *Insulin Chart* [Online]. Available: http://www.straighthealthcare.com/insulin-chart.html#humalog [Accessed 31 March 2015].

STURIS, J., POLONSKY, K. S., MOSEKILDE, E. & VAN CAUTER, E. 1991. Computer model for mechanisms underlying ultradian oscillations of insulin and glucose. *Am J Physiol,* 260**,** E801-9.

SWAN, K. L., DZIURA, J. D., STEIL, G. M., VOSKANYAN, G. R., SIKES, K. A., STEFFEN, A. T., MARTIN, M. L., TAMBORLANE, W. V. & WEINZIMER, S. A. 2009. Effect of Age of Infusion Site and Type of Rapid-Acting Analog on Pharmacodynamic Parameters of Insulin Boluses in Youth With Type 1 Diabetes Receiving Insulin Pump Therapy. *Diabetes Care,* 32**,** 240-244.

TABORSKY, G. J., JR. 2010. The physiology of glucagon. *J Diabetes Sci Technol,* 4**,** 1338-44.

TATTI, P. & LEHMANN, E. D. 2002. Using the AIDA--www.2aida.org--diabetes simulator. Part 1: recommended guidelines for health-carers planning to teach with the software. *Diabetes Technol Ther,* 4**,** 401-14.

THE AMERICAN NATIONAL RED CROSS. 2015. *Blood Components* [Online]. http://www.redcrossblood.org. Available: http://www.redcrossblood.org/learn-about-blood/blood-components [Accessed 25 May 2015].

THE DIABETES MALL. 2014a. *Carb Factor* [Online]. Available: http://www.diabetesnet.com/diabetes-control/rules-control/carb-factors [Accessed October 2014].

THE DIABETES MALL. 2014b. *Correction Factor* [Online]. Available: http://www.diabetesnet.com/diabetes-control/rules-control/correction-factor [Accessed October 2014].

THE DIABETES MALL. 2014c. *Hypoglycemia Unawareness* [Online]. Available: http://www.diabetesnet.com/diabetes-control/low-blood-sugars/hypoglycemia-unawareness [Accessed 26 May 2014].

THE DIABETES MALL. 2015. *Insulin Actions Times and Peak Times* [Online]. Available: http://www.diabetesnet.com/about-diabetes/insulin/insulin-action-time [Accessed 31 March 2015].

THE NOBEL FOUNDATION. 2015. *The Diabetes Dog Game* [Online]. Available: http://www.nobelprize.org/educational/medicine/insulin/game/insulin.html [Accessed 23 March 2015].

THOMPSON, R. 2014. *Glycemic Load Table* [Online]. lowglycemicload.com. Available: http://www.lowglycemicload.com/glycemic_table.html [Accessed 9 July 2015].

TOFFOLO, G., BERGMAN, R. N., FINEGOOD, D. T., BOWDEN, C. R. & COBELLI, C. 1980. Quantitative estimation of beta cell sensitivity to glucose in the intact organism: a minimal model of insulin kinetics in the dog. *Diabetes,* 29**,** 979-90.

TURNER, D., GRAY, B. J., DUNSEATH, G., LUZIO, S. D., BAIN, S. C., WEST, D. J., CAMPBELL, M. D. & BRACKEN, R. M. 2013. Increasing the duration of an acute resistance exercise session tempers exercise-induced hyperglycaemia in those with Type 1 diabetes. *Diabetic Medicine,* 30**,** 16-16.

TURNER, D., GRAY, B. J., WEST, D. J., CAMPBELL, M. D., HANLEY, S., LUZIO, S., DUNSEATH, G., BAIN, S. C. & BRACKEN, R. M. 2014. Similar magnitude of post-exercise hyperglycaemia following moderate and low intensity resistance exercise in individuals with Type 1 diabetes. *Diabetic Medicine,* 31**,** 70-70.

TURNER, D., LUZIO, S., GRAY, B. J., DUNSEATH, G., REES, E. D., KILDUFF, L. P., CAMPBELL, M. D., WEST, D. J., BAIN, S. C. & BRACKEN, R. M. 2015. Impact of single and multiple sets of resistance exercise in type 1 diabetes. *Scandinavian Journal of Medicine & Science in Sports,* 25**,** E99-E109.

TUTORIALSPOINT.COM. 2014. *Android Architecture* [Online]. tutorialspoint.com. Available: http://www.tutorialspoint.com/android/android_architecture.htm [Accessed 11 April 2015].

UNIVERSITY COLLEGE LONDON HOSPITAL NHS. 2013. Available: http://www.uclh.nhs.uk/PandV/PIL/Patient%20information%20leaflets/Correcting%20a%20high%20blood%20glucose%20level.pdf [Accessed 5 March 2015].

UP HEALTH SYSTEM MARQUETTE, D. L. H. 2013. *Insulin Comparison Chart* [Online]. Available: http://ww4.mgh.org/Physicians/Formulary%20Documents/Insulin%20Comparison%20Chart.pdf [Accessed 31 March 2015].

VAN CAUTER, E., POLONSKY, K. S. & SCHEEN, A. J. 1997. Roles of circadian rhythmicity and sleep in human glucose regulation. *Endocr Rev,* 18**,** 716-38.

VAN DER HEIJDEN, G.-J., TOFFOLO, G., MANESSO, E., SAUER, P. J. J. & SUNEHAG, A. L. 2009. Aerobic exercise increases peripheral and hepatic insulin sensitivity in sedentary adolescents. *The Journal of clinical endocrinology and metabolism,* 94**,** 4292-9.

VAN HERPE, T., PLUYMERS, B., ESPINOZA, M., VAN DEN BERGHE, G. & DE MOOR, B. 2006. A minimal model for glycemia control in critically ill patients. *Conf Proc IEEE Eng Med Biol Soc,* 1**,** 5432-5.

VAN RIEL, N. 2004. Minimal models for glucose and insuline kinetics - A Matlab implementation. Dept. of Electrical Engineering, BIOMIM & Control Systems, Eindhoven University of Technology.

VENES, D. & TABER, C. W. 2013. *Taber's cyclopedic medical dictionary,* Philadelphia, F.A. Davis.

WATSON, P. E., WATSON, I. D. & BATT, R. D. 1980. Total body water volumes for adult males and females estimated from simple anthropometric measurements. *Am J Clin Nutr,* 33**,** 27-39.

WATTANASOONTORN, V., BOADA, I., GARCíA, R. & SBERT, M. 2013. Serious games for health. *Entertainment Computing,* 4**,** 231-247.

WEBMD. 2011. *Nicotine and Blood Sugar a Dangerous Combo* [Online]. Available: http://www.webmd.com/diabetes/news/20110328/nicotine-and-blood-sugar-bad-combination [Accessed 1 April 2015].

WEBMD. 2015. *Types of Insulin for Diabetes Treatment* [Online]. Available: http://www.webmd.com/diabetes/guide/diabetes-types-insulin [Accessed 31 March 2015].

WEIGHT LOSS RESOURCES. 2015. *Glycaemic Index Tables* [Online]. weightlossresources.co.uk. Available: http://www.weightlossresources.co.uk/diet/gi_diet/glycaemic_index_tables.htm [Accessed 9 July 2015].

WHATHEALTH.COM. 2015a. *Glycemic Index List: Table Of Glycemic Index Values* [Online]. Available: http://www.whathealth.com/glycemicindex/list.html [Accessed 7 April 2015].

WHATHEALTH.COM. 2015b. *Glycemic Index Measurement: How To Measure The GI Of A Food Item* [Online]. Available: http://www.whathealth.com/glycemicindex/measurement.html [Accessed 7 April 2015].

WHATHEALTH.COM. 2015c. *Glycemic Index: What is the glycemic index?* [Online]. Available: http://www.whathealth.com/glycemicindex/overview.html [Accessed 7 April 2015].

WHATHEALTH.COM. 2015d. *Glycemic Load (GL)* [Online]. Available: http://www.whathealth.com/glycemicindex/glycemicload.html [Accessed 24 May 2015].

WICK, A. N., DRURY, D. R. & MACKAY, E. M. 1950. Glucose Space of the Body. *American Journal of Physiology*, 163**,** 224-228.

WILCOX, C. 2010. *Understanding Our Bodies: Insulin* [Online]. NutritionWonderland.com: Nutrition Wonderland. Available: http://nutritionwonderland.com/2010/05/understanding-our-bodies-insulin/ [Accessed 7 April 2015].

WINSAAM. 2007. *WinSAAM Website* [Online]. University of Pennsylvania, School of Veterinary Medicine. Available: http://www.winsaam.org/index.html [Accessed 22 March 2015].

WOODGER COMPUTING INC. 2001. *Java's Evolution* [Online]. Woodger.ca. Available: http://www.woodger.ca/jv_evol.htm [Accessed 11 April 2015].

WORLD HEALTH ORGANIZATION. 2015. *Diabetes, Fact sheet N°312* [Online]. Available: http://www.who.int/mediacentre/factsheets/fs312/en/ [Accessed 2 March 2015].

WRZESIEN, M. & RAYA, M. A. 2010. Learning in serious virtual worlds: Evaluation of learning effectiveness and appeal to students in the E-Junior project. *Comput. Educ.*, 55**,** 178-187.

YAZBECK, L., WATSON, M., NINNIS, N. & WASSOUF, S. 2013a. *The 0.12 Formula for the Management of Hypoglycaemia and Hyperglycaemia in Children with Type 1 Diabetes Mellitus: Does it work? (Poster)* [Online]. Available: http://www.medicalposters.co.uk/index.php/mp/article/view/34/23 [Accessed November 2014].

YAZBECK, L., WATSON, M., NINNIS, N. & WASSOUF, S. 2013b. G101(P) The 0.12 Formula For the Management of Hypoglycaemia and Hyperglycaemia in Children with Type 1 Diabetes Mellitus: Validation and Safety Data. *Archives of Disease in Childhood*, 98**,** A48-A49.

ÅRSAND, E., FROISLAND, D. H., SKROVSETH, S. O., CHOMUTARE, T., TATARA, N., HARTVIGSEN, G. & TUFANO, J. T. 2012. Mobile health applications to assist patients with diabetes: lessons learned and design implications. *J Diabetes Sci Technol*, 6**,** 1197-206.

ÅRSAND, E., TATARA, N., OSTENGEN, G. & HARTVIGSEN, G. 2010. Mobile phone-based self-management tools for type 2 diabetes: the few touch application. *J Diabetes Sci Technol*, 4**,** 328-36.