Non-Invasive Blood Glucose Range Estimator

Medical robotics, cyber physical engineering and virtual reality

Ioannis Papaefstathiou

Christos Karapapas - Mohammed Lubbad

Thessaloniki 2023

Abstract

With this project we explored the development of a non-invasive blood glucose estimation device using an Arduino MKR microcontroller, a MAX30100 heart rate sensor and an LM35 temperature sensor. The focus both on individual parts of the process, such as reading raw values and converting them to units of measurement, as well as on the overall process of designing and setting up the device, including the implementation of a machine learning (ML) model to predict blood glucose levels based on photoplethysmography (PPG) and temperature measurements. This work is in response to the growing number of non-invasive devices in relevant works in the literature with focus on exploring how viable this solution would be and also to explore all the challenges these solutions face from design to implementation.

The process includes three main stages, the hardware assembly and data reading, the model training using an existing dataset and model deployment on the Arduino.

A large dataset, found in the literature, was used to train the ML models that were later converted in C++ to be able to be included into the Arduino microcontroller and provide predictions for the glucose levels based on new measurements. The dataset, which includes heart rate, inter-beat interval, and skin temperature, along with glucose levels, was processed and used to train various ML models using the Pycaret library. While regression models and multi-class classifiers showed limited success, a binary classification model demonstrated promising results, achieving around 96% accuracy. The binary classifier was able to predict whether the glucose level of a subject was above or below a certain threshold that was selected to be equal to 114 mg/dL.

Deploying the model onto the Arduino posed challenges due to the limited memory of the microcontroller but after exploring various alternatives, the model was successfully deployed using the "micromlgen" library.

Overall, the project showed that it is possible to use PPG signals and temperature to predict a general class of the current glucose level of a subject, proving that non-invasive glucose monitoring could possibly have some value.

Future work could focus on ML model optimization, collection of custom data or using additional features from the existing one, improving model deployment process, conducting real-world testing and even building a better user interface.

Chapter 1. Introduction

Concept

Based on World Health Organization facts, people who suffer from diabetes rose from 108 million in 1980 to 422 million in 2014. Diabetes is causing multiple serious health problems, ranging from blindness to stroke and even creates the necessity for lower limb amputations. Only during 2019 diabetes and kidney diseases caused by diabetes were accountable for 2 million deaths worldwide. And according to later investigations there is an increase rate in these numbers.

Hyperglycemia is a condition caused by high glucose levels in blood. The human body, to mitigate this condition, creates a hormone called insulin which helps balance the glucose levels. When the system lacks the ability to produce enough insulin then it experiences a hyperglycemic episode. Having frequent episodes, meaning leaving hyperglycemia untreated, can cause damage to nerves, blood vessels, tissue and organs.

The first line of defence regarding diabetes, is to know the current state of the glucose levels in the blood. Glucose can be directly measured and thus provide an index to monitor the current state of the condition of a subject. However, measuring glucose can be somewhat uncomfortable as the procedure involves piercing the subject's finger. Given that this procedure must be repeated for each measurement and that the measurement must be repeated regularly, there is an urgent need for research into non-invasive methods of glucose measurement and this situation is the primary motivation behind our effort.

Furthermore, a lot of attempts have already been made both on the academic and on commercial level to create non-invasive glucose monitoring systems. Most of these attempts rely on the usage of spectroscopy techniques, which is a way to measure interactions between light and matter. The main principle of this technique is having at least one light source and one receiver to measure the light after it has passed through pulsatile tissue, meaning a fingertip, an earlobe or any thin section of the body with blood perfusion. The fact that there are so many existing implementations of non-invasive blood glucose estimators exists, poses the second motivation behind our effort since we would like to explore the challenges of building such a device and try to validate whether such a device is possible to be built and what could be its accuracy levels.

A more specialised variant of spectroscopy, photoplethysmography (PPG) is broadly used in devices known as oximeters that can measure the oxygenation of blood and the current heart rate. PPG devices usually use red or infrared light sources that range from about 660nm to 940nm wavelengths.

Our attempt could be described as the recreation of a non-invasive blood glucose range estimation device, of course not at production but rather at proof-of-concept level. For that purpose, we are going to use the MAX30100 sensor which includes an infrared light emitter at 940nm and a photodiode sensor that can detect light fluctuation in these wavelengths along with the LM35 temperature sensor.

The main reason for using multiple sensors is to have multiple variables and explore their correlation with glucose levels using machine learning algorithms. However, we need to consider that using an Arduino as our microcontroller would restrict us in computation power, meaning that the training of the models should happen separately from the device.

Therefore, the experiment should be separated into multiple stages: assembly, measurements, model training, new estimations, which are going to be explained in more detail in the methodology section.

Objectives

As described in the concept section. We tried to approach the issue of creating a non-invasive blood glucose measurement device for academic reasons and to explore the challenges this project includes, although a plethora of existing similar solutions already exist in the market.

Therefore, we could group our objectives into two categories, the mandatory objectives related to making the system work as intended and some optional objectives regarding the security issues and the performance of the machine learning models. In any case, we kept in mind that we should investigate the challenges from a biomedical engineering perspective. Meaning for example that in terms of the machine learning model we are highly interested in the sensitivity and specificity metrics since they are those metrics that can give us an idea of the positive to negative type of ratios. For the security we considered that whatever attempt to secure the transmission should not rely on extra hardware and that we would have a very strict amount of memory for extra cryptographic libraries. And overall, we kept in mind that such a device should be as simple as possible in anything related to the user experience.

Mandatory objectives:

- Assemble the system.
- Connect at least two sensors.
- Connect at least two actuators.
- Read from sensors.
- Perform necessary conversions from raw values to volt and units of measurement.
- Gather data and match sensor values with ground truth measurements.
- Train a classification ML model and load it on the system.
- Create a response with results on the LCD screen.

Optional and custom objectives:

- Explore the performance of multiclass classification ML models.
- Explore the performance of regression ML models.
- Inspect the data transmission during the data gathering process given that the system is connected wirelessly with a computer and investigate possible data encryption that could be applied.

Glucose Levels

The definition of Hyperglycemia, as defined by the National Library of Medicine, consists of the words hyper (high), glycaemic (sweet/sugar) + haima (blood) which are of Greek origin. Hyperglycaemia, as the term suggests, is the result of high sugar levels (glucose) in a person's blood and it is also referred to as “high blood sugar” or “high blood

glucose". This occurs when the body has too little insulin (hormone) or when the body cannot process and use insulin properly (insulin resistance).

An interesting fact about the glucose levels is that there isn't a single cut-off pair that can distinguish those with high, low and normal glucose levels and that is because the glucose concentration fluctuates frequently during the day depending on the pre-existing conditions condition of a subject and factors such as the food intake, physical activity, medication, stress, illness, menstrual cycle, sleep, alcohol and dehydration. That is a crucial part for our project since we need to decide which set of thresholds are we going to use for our projects' needs.

The American Diabetes Association (ADA) has set a group of thresholds to diagnose hyperglycemia for fasting and normal fasting scenarios. After fasting (meal), glucose levels must be greater than 130 mg/dL, or postprandial levels greater than 180 mg/dL, as stated by the American Diabetes Association, 2020. Normal fasting blood glucose is expected to concentrate between 70 mg/dL (3.9 mmol/L) and 100 mg/dL (5.6 mmol/L). If a fasting person's blood glucose differs between 100 to 125 mg/dL (5.6 to 6.9 mmol/L) and in those cases it is recommended to adapt changes in lifestyle and monitoring of glycemia. When a patient's fasting plasma glucose is between 100 mg/dL to 125 mg/dL, it is detected that there is impaired glucose tolerance, or pre-diabetes. Additionally, below 70 mg/dL (3.9 mmol/L) a patient is at low fasting blood glucose concentration meaning hypoglycaemia. However, this isn't the only way to categorise blood glucose levels, in fact there are multiple tests each one with its own cut-off scores, but in general there are three main test types.

- Fasting Plasma Glucose (FPG), a measurement of blood glucose after fasting (not eating or drinking anything except water) for at least 8 hours.
    - Normal: Less than 100 mg/dL (5.6 mmol/L)
    - Prediabetes (impaired fasting glucose): 100-125 mg/dL (5.6-6.9 mmol/L)
    - Diabetes: 126 mg/dL (7.0 mmol/L) or higher on two separate tests
- Oral Glucose Tolerance Test (OGTT, a measurement done before and 2 hours after drinking a beverage containing 75 grams of glucose dissolved in water.
    - Normal: Less than 140 mg/dL (7.8 mmol/L)
    - Prediabetes (impaired glucose tolerance): 140-199 mg/dL (7.8-11.0 mmol/L)
    - Diabetes: 200 mg/dL (11.1 mmol/L) or higher on two separate tests
- Random Plasma Glucose Test, also called a casual plasma glucose test, measures blood glucose without regard to when you last ate.
    - Diabetes: 200 mg/dL (11.1 mmol/L) or higher, and you have symptoms of diabetes (e.g., increased thirst, increased urination, unexplained weight loss)

For the human organism to achieve homeostasis, the pancreas plays a key role. More specifically it regulates the macronutrient digestion and hence metabolism/energy homeostasis by releasing various digestive enzymes and pancreatic hormones. It is located behind the stomach within the left upper abdominal cavity and is partitioned into head, body and tail. When the glucose level is too high, the pancreas secretes more insulin. When the levels drop, it releases glucagon to raise them. Through its various hormones, particularly glucagon and insulin, the pancreas maintains blood glucose levels within a very narrow range of 4 - 6 mm. This preservation is accomplished by the opposing and balanced actions of glucagon and insulin, referred to as glucose homeostasis. During sleep or in between meals, when blood glucose levels are low, glucagon is released from α-cells to promote hepatic glycogenolysis. In addition, glucagon drives hepatic and renal gluconeogenesis to

increase endogenous blood glucose levels during prolonged fasting. In contrast, insulin secretion from β-cells is stimulated by elevated exogenous glucose levels, such as those occurring after a meal. [PubMed Central, NIH NLM, 2016]

Existing Solutions

Nowadays blood glucose levels can be measured through a variety of devices. A first level categorization of these measurement devices could be done by the way of sampling retrieval and type of sample.

- Invasive Methods
    - Classic Blood Glucose Monitoring: This method involves pricking the finger or another body part to obtain a blood sample, which is then analysed using glucose test strips and a glucose metre.
    - Intravenous Sampling: In certain medical settings, blood glucose levels can be measured directly from an intravenous (IV) line.
- Non-Invasive Methods
    - Infrared Spectroscopy: This method utilises infrared light to measure glucose levels by analysing the interaction between light and tissue.
    - Optical Coherence Tomography (OCT): OCT uses low-coherence light to create high-resolution cross-sectional images of tissues and can be employed to estimate glucose levels.
    - Photoacoustic Imaging: It combines laser-induced ultrasound waves and light absorption to determine glucose concentrations non-invasively.
    - Raman Spectroscopy: Raman spectroscopy analyses the scattered light to assess glucose levels.
    - Near-Infrared Spectroscopy (NIRS): NIRS measures the absorption of near-infrared light to estimate glucose levels.
- Non-Invasive Methods based on biological fluids.
    - Sweat-based sensors measure glucose levels in sweat, offering a non-invasive and continuous monitoring approach.
    - Saliva-based sensors aim to measure glucose levels in saliva samples, providing a non-invasive alternative to blood-based measurements.
- Continuous Glucose Monitoring (CGM)
    - Subcutaneous Sensors: These sensors are inserted under the skin to measure interstitial fluid glucose levels continuously. The data is transmitted wirelessly to a receiver or smartphone for real-time monitoring and trend analysis.
- Closed-Loop Systems
    - Artificial Pancreas Systems: These systems combine continuous glucose monitoring with an insulin delivery system, providing closed-loop control of glucose levels.

The device of our project falls into the category of non-invasive methods that utilise infrared spectroscopy. Theoretically, this technique is quite simple since the method works only by shining near-infrared light (NIR) onto the skin and measuring the light absorbed by the glucose in the interstitial fluid under the skin. In reality though, glucose is not the only

substance in the body that absorbs NIR light, making this task a challenge. Furthermore, this challenge becomes even harder if we consider that the spectrum range that glucose absorbs is not unique but overlaps with the spectrum of NIR that water, proteins and other substances absorb.

One of the most popular solutions that made it to the market with an FDA approval was the GlucoWatch and its upgrades developed by Cygnus Inc. But after further investigation in the literature one can easily find that the accuracy of this device was rather disappointing. More specifically, in her work, Eva Tsalikian (2004) "Accuracy of the GlucoWatch G2 Biographer and the Continuous Glucose Monitoring System During Hypoglycemia", we read that the device accuracy is about 70% for glucose levels under 70 mg/dL, that it triggers false alarms for more than half of the cases and that in absolute values the difference compared to ground truth is constantly at roughly 15 mg/dL.

Since the discontinuation of GlucoWatch, no other non-invasive glucose metre has gotten an FDA approval and we observe that instead another category of glucose metres is currently on the rise, the so-called Continuous Glucose Monitoring System (CGMS). An example of such a device is the G5 CGMS, which is implanted under the skin and monitors glucose throughout the day without the need for invasive needle-based blood samples and with a much greater accuracy at about 94.5%.

Chapter 2. Specifications

Hardware

The hardware that the system is composed of, and additional devices used during the experiment:

- Microcontroller:
  - Arduino MKR1000
- Sensors:
  - MAX30100 for measuring the heart rate.
  - LM35 for measuring the temperature.
- Actuators:
  - JHD162A LCD screen to print the estimation result.
  - RGB LED to indicate when the device is powered on and give a simple visual indication about the estimation.
- Ground truth devices:
  - Fingertip Pulse Oximeter LK-87, to validate/calibrate MAX30100.
  - CK-1502 Infrared Digital Thermometer, to validate/calibrate LM35.
  - Contour Care, to compare with the final estimation of our device.

The MAX30100 sensor is a complete circuit that consists of an IR-A LED emitter that emits both infrared light at 880 nm and visible (red) light at 660 nm wavelength respectively. The circuit also bundles with a photodiode, in other words an LED receiver with capability to receive light from both wavelengths. The LM35 sensor is a temperature sensor capable of reading temperatures between -55oC to 150oC with an accuracy of 0.5oC.

The JHD162A is one of the two actuators of the system and it is an LCD display of 16x2 and its main role is to prompt the user and display the estimated result. The second actuator is RGB LEDs that emits light in the visible spectrum to be used as indicators to

inform the user about the process and the also provide a simpler representation of the estimation, for example a blue light if the estimated glucose is on the low category and a red light if the estimated glucose is on the high category.

About the ground truth devices, the Fingertip Pulse Oximeter LK-87 is used to validate the readings and calibrate the MAX30100 sensor if needed, since the primary function of the MAX30100 sensor is to function as a heartbeat rate sensor. The CK-1502 infrared digital thermometer The Contour Care will provide us the ground truth value for glucose, which also happens to be the target variable in the model building process and it will be used in the final step to validate the estimations.

Dataset

As described in the methodology section one of the main sections of the process is the gathering of data to use for the ML model training. Depending on the desired ML algorithms to use, the sample should comprise of a few hundred to several thousand samples for the algorithm to perform well. Of course, for each sample a volunteer should be found making the difficulty and time consumption of the process exceed the scope of the current project. Additionally, another issue that we would have to address if we decided to follow this path would have been the compliance with any related consent and privacy directives and regulations.

Therefore, for the aforementioned reasons, we searched and found a related dataset that has all the characteristics that our custom data would have had. The research in which we found the dataset “BIG IDEAs Lab Glycemic Variability and Wearable Device Data v1.1.1” was published in May 2023 [Cho, P., et al.]. The main features that we would have recorded, which are the heart rate and the temperature of the subjects along with the target value of blood glucose level. More specifically the dataset consists of the following fields and has in total measurements for 16 subjects.

Of course from all of the following features, we are interested mostly in those that our sensors can measure, which are the HR, the IBI and the Skin temperature. The complete list of dataset features is the following:

- Triaxial accelerometry
- Blood volume pulse
- Interstitial glucose concentration
- EDA
- HR
- IBI
- Skin temperature
- Food Log

Software

To accomplish all the mandatory objectives and have a working device as a result, a series of software programs and libraries were used. Two main platforms were used as part of the development environment, Jupyter Notebooks using Python and the Arduino IDE using C++.

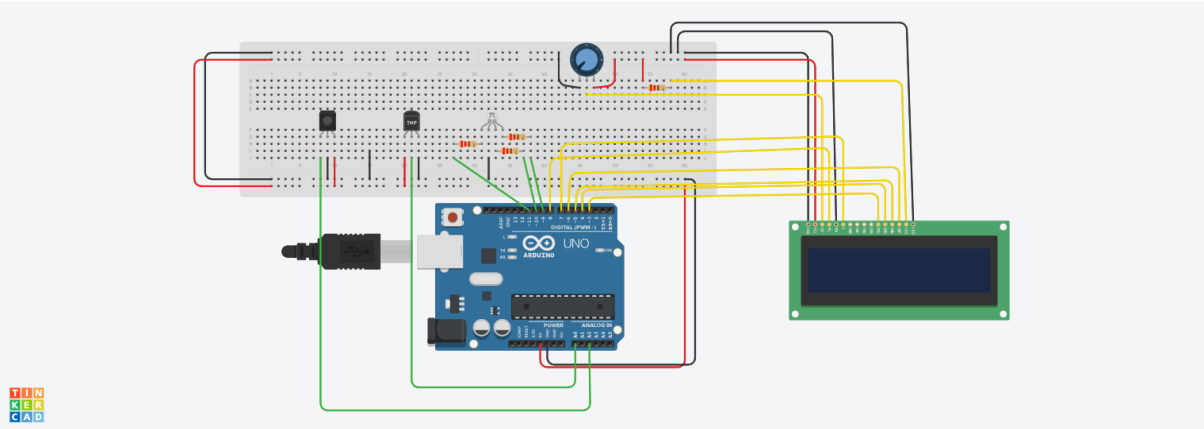
Additionally, the TinkerCAD online platform was used to design the overall circuit design of our hardware implementation, to virtually test our configuration, especially regarding the resistors, to avoid the physical damage of our equipment and to simulate some functionality while we waited for the ordered hardware to be delivered.

The DBeaver was used to import the dataset from individual files into a database scheme we created. Furthermore, we used the DBeaver to write queries with which we managed to aggregate the data from tables to views, from which we then loaded the data into Python dataframes.

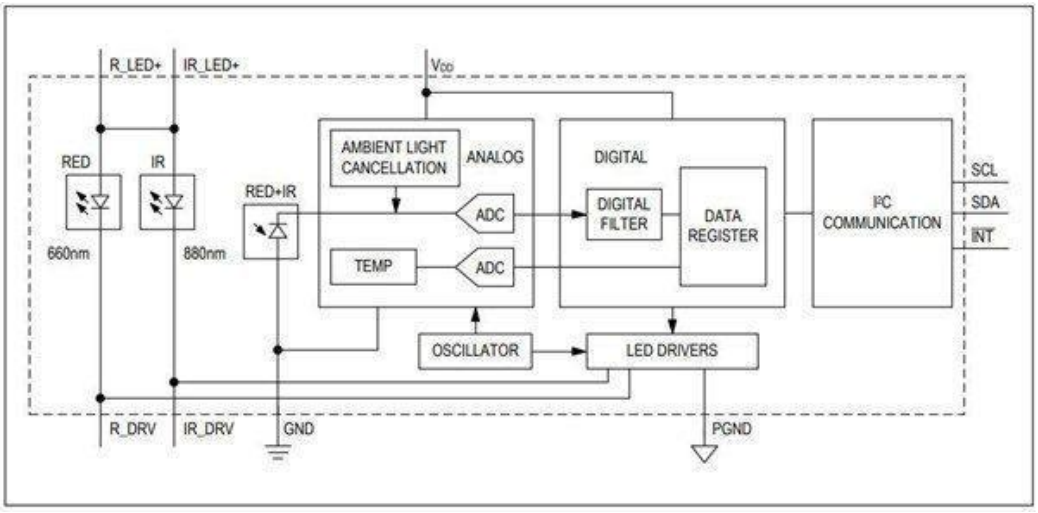
The Jupyter Notebooks were used to make an exploratory data analysis (EDA) of the dataset and to train a series of ML models and inspect their performance using various metrics that we will present in the methodology section. Since our main concern in the context of the project was not the performance optimization of the ML models but rather to achieve an overall working state of our device, we made use of the Pycaret library to speed up and simplify the model training process. The second most important library that we made use of was the “micromlgen” library which enables the easy interpreting of an existing trained ML model to C++ code.

For the part related to the Arduino code, we used the Arduino IDE along with the library “MAX30100_PulseOximeter” which is specialised in the reading of values from the MAX30100 sensor.

Schematics



Overall setup circuit schematic



MAX30100 sensor internal circuit schematic

Chapter 3. Related Work

During our research for this project, we explored various related works from the literature to get an idea of how researchers are approaching the same problem of creating a non-invasive blood glucose measuring device. Another reason for researching in literature for related work was to validate the usage of ML techniques to achieve the prediction of blood glucose level based on PPG and other features.

One interesting work that does make use of ML techniques is the work of Enric Monte-Moreno (2010) “Non-invasive estimate of blood glucose and blood pressure from a photoplethysmography by means of machine learning techniques” where a system for non-invasive estimation of blood glucose level (BGL), systolic (SBP) and diastolic (DBP) blood pressure using a photoplethysmography (PPG) and machine learning techniques is presented. The system consists of a PPG sensor, an activity detection module, a signal processing module that extracts features from the PPG waveform, and a machine learning algorithm that estimates the SBP, DBP, and BGL values. The machine learning techniques tested were ridge linear regression, a multilayer perceptron neural network, support vector machines, and random forests. The best results were obtained with the random forest technique, with R2 values of 0.91 for SBP, 0.89 for DBP, and 0.90 for BGL estimation. The distribution of the points on a Clarke error grid placed 87.7% of points in zone A, 10.3% in zone B, and 1.9% in zone D for glucose estimation. The system complies with the grade B protocol of the British Hypertension Society for blood pressure and only in 1.9% of the cases did not detect hypoglycaemia or hyperglycaemia.

Furthermore, Sahnius Usman, Mamun Bin Ibne Reaz and Alauddin Mohd Ali in their work “Repeated measurement analysis of the area under the curve of photoplethysmography among diabetic patients” examine the possibility that features derived from PPG such as the Area Under the Curve (AUC-PPG) could potentially hold useful information that could indicate the blood glucose levels. The area under the curve (AUC-PPG) was compared between patients with HbA1c<8% (Group 1) and HbA1c>10% (Group 2). The AUC-PPG was significantly higher in diabetic subjects with HbA1c<8% than in those with HbA1c>10%. The study used a paired t-test to investigate the association between the first and repeated measurements and found no significant difference in AUC-PPG between

the first and repeated measurements for either group of diabetic patients. The study did not mention accuracy as a primary result.

As an example of more advanced ML techniques usage, Antonio Alarcón-Paredes et al. in their work “An IoT-Based Non-Invasive Glucose Level Monitoring System Using Raspberry Pi” present a complete solution that includes a visible laser beam and a Raspberry Pi Camera to take pictures of the user's fingertip and compute their histograms. The data is then processed by an artificial neural network (ANN) implemented on a Flask microservice using the TensorFlow libraries. A few interesting specifications are first the number of participants which was 514 and the morphology of the actual ANN which included an input layer of 256 neurons, two hidden layers of 1024 neurons each, a 0.2 dropout layer and finally a classification layer based on RELU. The model was trained for 100 epochs and the system results were validated against laboratory blood tests and achieved a mean absolute error of 10.37% and a Clarke grid error of 90.32% in zone A.

However, the usage of ML models is not the only approach. Even with quick research on literature one can find many examples of works that follow a simpler approach, such as the work of R.A. Buda and M. Mohd Addi, that with their work “A Portable Non-Invasive Blood Glucose Monitoring Device” they present the design of a device that detects glucose concentration in blood and determines the required insulin dose corresponding to the body mass index (BMI) of the user. The device is intended for use by both diabetic and non-diabetic patients to help maintain a normal blood glucose level for a healthy lifestyle. The paper discusses the hardware implementation and methodology used in the development of the device, as well as the results of in vitro and in vivo experiments that proved the reliability of the device. The accuracy of their implementation mentioned differs at about 4% to 16% compared to classic invasive ways of measurement. The device measures the relationship between glucose concentration and voltage using in-vitro experiments. Their work suggests that the two variables have a strong linear relationship; voltage increases as glucose concentration increases. The device then uses this relationship to calculate the glucose level based on the voltage readings obtained from the near infrared sensor. The device also considers the user's body mass index (BMI) to determine the required insulin dose. So, the suggestion is done in a conditional approach, resembling an expert decision support system and not dynamically.

Finally, there are more than five thousand papers relating to the photoplethysmography methods for non-invasive glucose estimation, which indicates that there might be some correlation between data readings from PPG sensors and blood glucose levels. But the size of literature regarding that matter and the non-widespread commercial use of such devices could also indicate that there is still room for improvement.

Chapter 4. Methodology

Process Stages

The methodology we followed, to accomplish the objectives of the project, could be divided into three main parts.

- The first stage has to do with the assembly of the device in terms of hardware, loading a basic code to get reading from sensors and validate the readings against ground truth devices to determine if any calibration is needed.
- The second stage is related to the data of the project. The initial thought was to find some volunteers, use the initial hardware setup to gather values for their heart rate and temperature and also measure their blood glucose levels with a classic invasive blood glucose monitor device to build our dataset. However, as stated in the dataset specifications chapter, this process would exceed the context of this project in many terms. Therefore, we resorted to the solution of finding an existing dataset with comparable characteristics, which we managed to do so. Furthermore, this stage is related to the steps needed to reach a trained machine learning model, from an Exploratory Data Analysis to preprocessing, training, model performance measurement and finally model optimization.
- The third stage is related to loading the ML model onto the Arduino MKR microcontroller and performing new measurements to use in predicting their relevant glucose level.

First Stage

Reading and Conversions

Regarding the temperature reading, the process from reading the raw output values of the sensor to converting them to meaningful units of measurement are the following.

- Reading data: The LM35 sensor is connected to an Arduino board via three pins: VCC (to 5V pin), GND (to a ground pin), and Out (to an analog input pin). Once connected, you'll use a basic Arduino sketch to read the sensor data. The sensor data, at this point, will be raw analog-to-digital conversion (ADC) values from 0 to 1023, corresponding to voltage values from 0V to 5V.
- Converting raw data: The raw data from the sensor needs to be converted into meaningful temperature readings. This is done in two steps: firstly, converting the ADC values to voltage (Voltage = (ADC Value / 1024) * Vref, with Vref as the reference voltage, typically 5V), and then, converting this voltage to Celsius using the scale factor of the LM35 (Temperature = Voltage / 0.01).
- Filtering data: Despite calibration, the sensor readings may still contain noise which can lead to fluctuations in the data. To smoothen out these fluctuations, data filtering techniques can be applied. Simple methods can include taking the average of multiple readings, or more complex methods such as a moving average or a low-pass filter. This step can help in providing more stable and reliable temperature readings.

- Calibrating the sensor: Even with the conversion in place, the readings from the LM35 may not exactly match the true temperature due to various factors. To rectify this, calibration is performed. In a two-point calibration, you measure the sensor output at two different known temperatures (using a reliable reference like a precise digital thermometer) and adjust the conversion formula's scale factor and offset until the sensor's output aligns with the known temperatures.

However, doing the same process to measure the Heart Rate from the MAX30100, might include the same steps, but in fact the process differs a lot due to the fact that we cannot simply convert one single value into beat per minute, which is the corresponding unit of measurement. This is because we first need to gather a window of readings, then find the peaks that represent the QRS part of an PPG signal and then calculate the number of peaks for a minute. So the process for getting the Heart Rate is the following.

- Reading data: The MAX30100 sensor is interfaced with the Arduino through the I2C protocol, where SDA and SCL pins of the sensor are connected to the corresponding pins on the Arduino. A library like the MAX30100lib is often used to interface with the sensor and gather raw data. A basic sketch can be written to initialise the sensor and read the raw heart rate data. It's important to note that the raw data from the sensor isn't in beats per minute (BPM), but represents the amount of infrared light absorbed by blood flowing through the wrist.
- Converting raw data: Conversion of raw data into a meaningful heart rate reading in BPM, as already mentioned in the software specifications section we made use of the “MAX30100_PulseOximeter” library.
- Filtering data: Finally, the heart rate data might be noisy due to motion artefacts, changes in blood flow, or other factors. To smooth out these fluctuations and get more reliable heart rate readings, you can apply various filtering techniques. One simple approach is averaging multiple readings over a period of time. More advanced filtering techniques like a low-pass filter or a moving average filter can also be implemented to minimise noise and improve the stability of the heart rate readings.
- Calibrating the sensor: Before calibrating the sensor we validated the output with the ground truth device we already mentioned in hardware specifications, the “Fingertip Pulse Oximeter LK-87” and we found no discrepancy, in fact the readings from our sensor followed the readings of the oximeter closely, therefore we omitted the step of performing an adjustment on the peak detection threshold, which is usually the easiest way to calibrate a heart rate sensor.

Second Stage

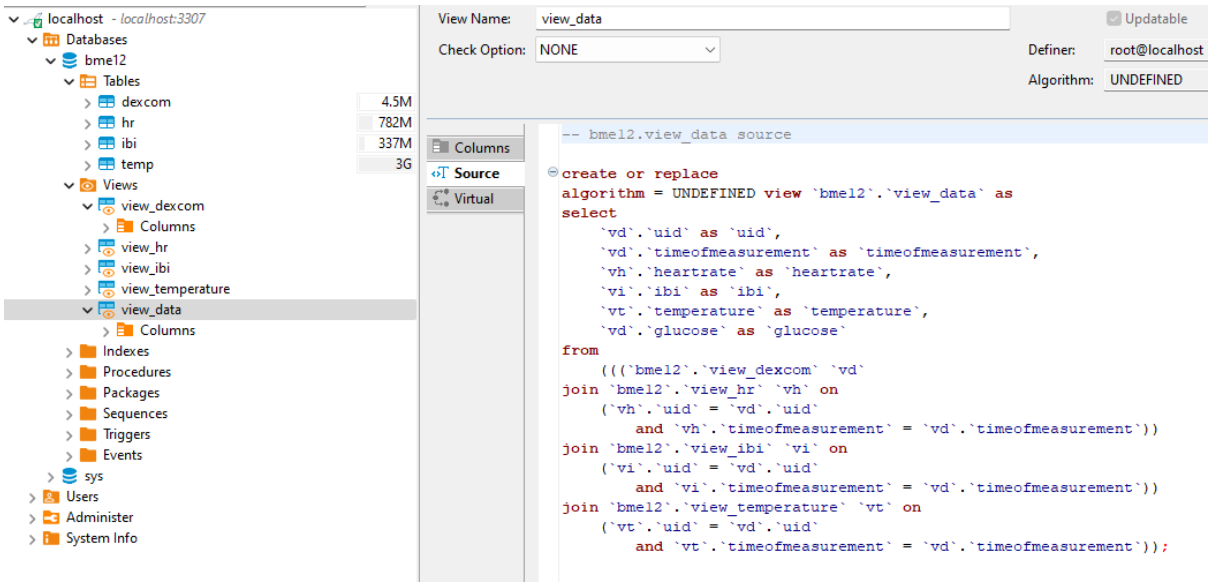
Data Preparation

Before proceeding with the exploratory data analysis we had to transform the data from the initial format to one that we could load into a Python dataframe. The data initially was splitted into 16 folders, one for each subject and inside each of these folders there was a csv file per feature, meaning that we had 128 files in total. Each file contains a timestamp and the actual measured value.

Given the fact that the measurements could suffer from fluctuating values from one sample to the next and also the fact that the glucose measurements were far less and more

scarce than the rest of the measurements, we aggregated the data by calculating the average value, for example, of the heart rate measurements for the same minute that a glucose measurement was taken. The same approach was followed also for the temperature measurements.

After the aggregation of temperature and heart rate measurements we created a view which included the subject id, the timestamp, the temperature, the heart rate and the glucose levels. Finally the data from the overall view were exported into a csv file and the Pandas library was used to load it into a dataframe.



Final view from which we exported the data we used in the Python dataframe.

Model Training and Performance

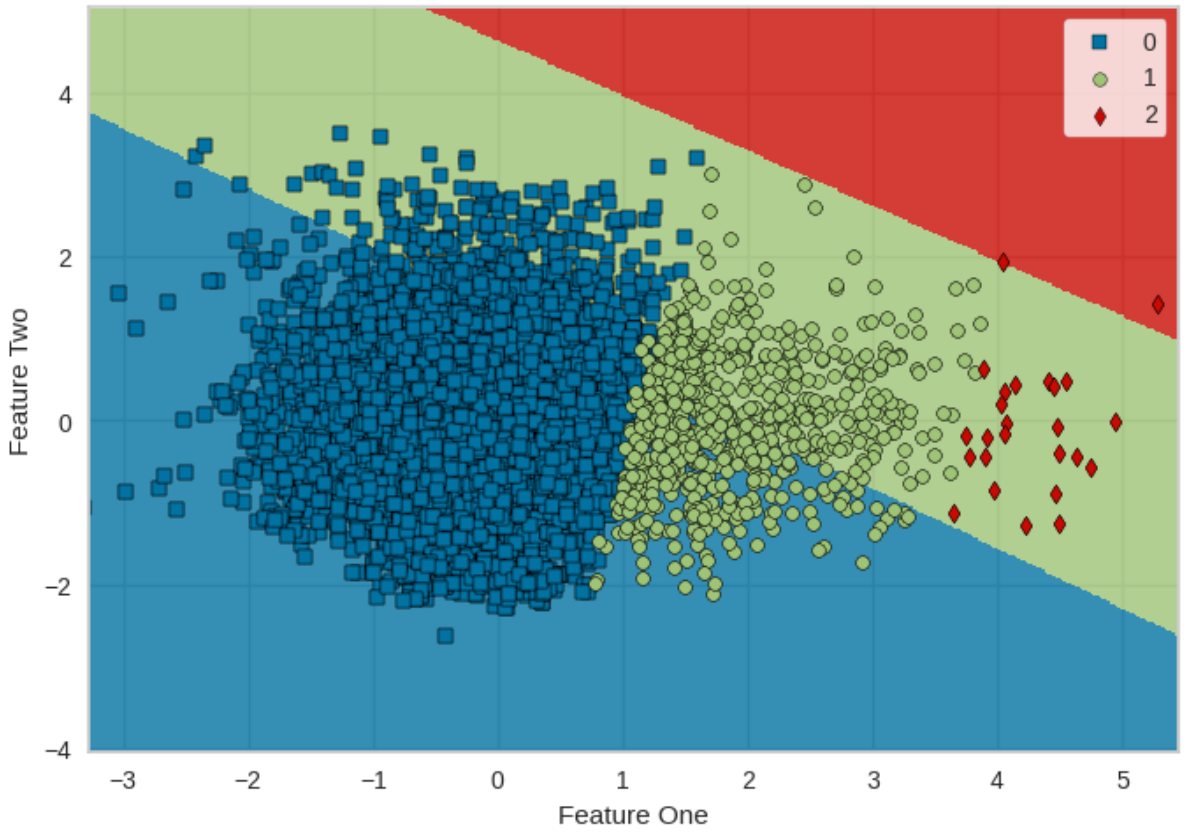
A series of attempts were made to create a model that would be meaningful in the context of this project.

The ideal model for this project would have been, of course, a regression model that would be able to predict a specific glucose value in mg/dL, however after building many different models, implemented in the “regression-pycaret” notebook, and trying various strategies such as binning of the two features, Principal Component Analysis (PCA), model optimization and various training algorithms it became clear, through the model metrics, that at least for the available data we had at the moment, it wasn’t feasible to create such a model with an acceptable performance. To be more precise the actual performance of these models had between 16 to 18 Mean Absolute Error (MAE) which is high, but it is at a scale of error that we saw also in the literature as we mentioned in the literature review section. However, the R2 (squared) metric was always near zero which indicates clearly that the models did not perform well.

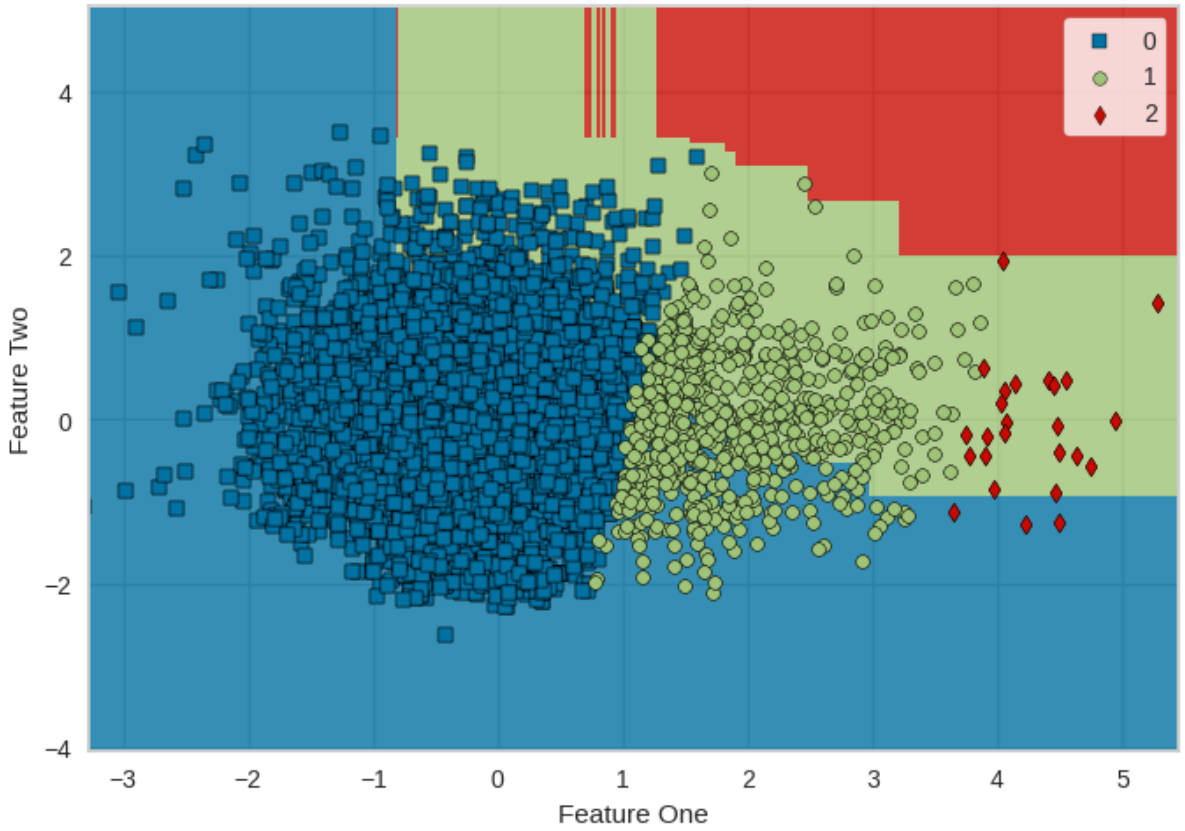
The second major attempt, implemented in the “multiclass-classifier-pycaret” notebook, was done by creating multiclass classifiers which from a metrics perspective seemed to perform great, to a point that the accuracy indicated overfitting, since it ranged from about 98.2 to 99.1 %. For that reason we proceeded with printing additional metrics such as the Area Under Curve (AUC), Confusion Matrix (CM) and decision boundaries. At this point both AUC and CM for the algorithms (lightgbm, logistic regression, xgboost and Quadratic Discriminant Analysis (QDA)), presented the same high performance. However,

the decision boundary for the same algorithms showed a bit of a different image and more particularly that the boundary of each model failed to classify correctly many of the samples, something that could be explained by either a fault in our process or an indeed bad performance. Indicatevelly, some of the important metrics to display are for example the following.

Model	Accuracy	AUC
Extreme Gradient Boosting	0.9916	0.9995
Logistic Regression	0.9912	0.9996
Light Gradient Boosting Machine	0.9911	0.9994
Quadratic Discriminant Analysis	0.9824	0.9988



Decision Boundary for Logistic Regression for multiclass classification



Decision Boundary for Extreme Gradient Boosting for multiclass classification

The same phenomenon was observed also for the rest of the algorithms in the multiclass classification series of experiments and they are omitted for brevity.

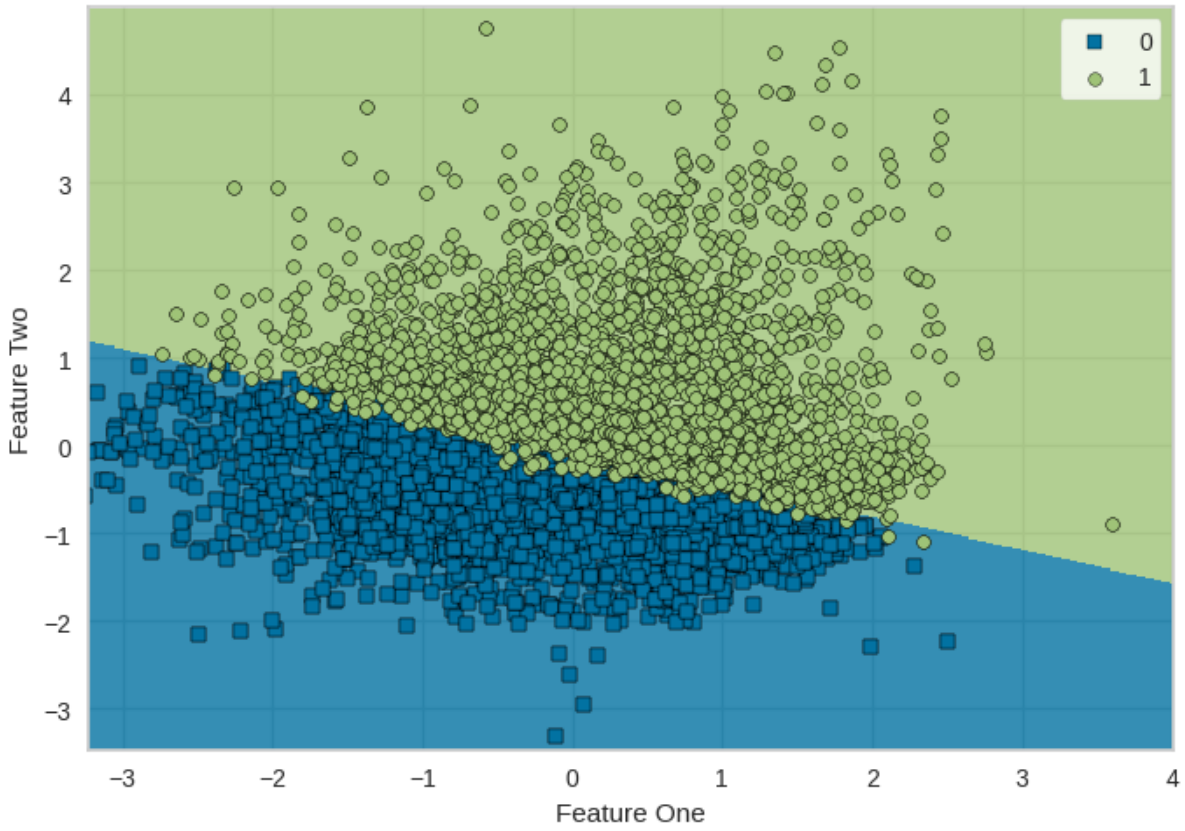
Another significant issue to mention at this point is that the classes were highly imbalanced with the cut-off values we had selected, which were 140 and 200 mg/dL to create the three classes described in the Oral Glucose Tolerance Test.

In the third attempt we tried to address the imbalance of target classes. We had two options, either use an over/under-sampling algorithm, for example the Synthetic Minority Over-Sampling Technique (SMOTE) that generates synthetic data or use different cut-off value-s that would create more balanced classes. We chose the second technique because there is a cut-off that splits the dataset in two quite equal parts and also happens to be the middle point of the middle class (Pre-Diabetes) of the Fasting Plasma Glucose (FPG) measurement test. Therefore, we proceeded with using the 114 mg/dL as cut-off value, which creates two classes with 10503 and 9715 samples respectively. We then proceeded with setting up our model to be trained, including the robust normalisation technique, 10-fold cross-validation, PCA with 2 components and binarization of the heart rate and temperature features. The result this time was a performance that seemed much more realistic and it is described by an accuracy between 95.7 to 96.1 % for the same algorithms from the previous attempts.

Model	Accuracy	AUC

Extreme Gradient Boosting	0.9574	0.993
Logistic Regression	0.9662	0.9952
Light Gradient Boosting Machine	0.9535	0.9918
Quadratic Discriminant Analysis	0.9616	0.9948

Also in this scenario, not only the accuracy and the AUC were indicating a good performance without overfitting but also the decision boundary justifies these accuracy and AUC numbers.



Decision Boundary for Logistic Regression for binary classification (114 mg/dL cut-off)

Decision Boundary for Extreme Gradient Boosting for binary classification (114 mg/dL cut-off)

Third Stage

Model Deployment and Size Limitation

Deploying the trained model on an Arduino microcontroller poses a great challenge from multiple perspectives.

First of all not any model can be deployed due to the memory size on Arduino which limits us to about 256 KB minus the bootloader and the rest of the custom code size. On the other hand our best performing model is about 966 KB, which leaves us with a couple of options.

The first option would ideally be the transformation of the model to a Tensorflow Lite model which is optimised to produce ML models of small size for usage in microcontrollers.

And the second option would be to change the implementation design and include a service, probably on the cloud or on a local machine that would host a process to make predictions with our model as is and our newly collected data transferred from the microcontroller through the network.

The second solution is very interesting and if we had attempted it we would also have had the chance to explore the challenges of safely transmitting the data to the network, however, due to the time limit of this project we had to follow the first path which is transforming the model to one of smaller size.

The first and most obvious way to accomplish that would be the application of various conversions to reach the format of a Tensorflow Lite model. And we mention conversions

and not a single conversion because there is no library offering a direct conversion between our source and target type. Therefore we need to convert the Pycaret model, which is literally a Scikit-Learn model, into a ONNX model, then to a Tensorflow model and finally into Tensorflow Lite. And although this seems relatively feasible, in reality it is a complex process with many impediments such as that from one conversion to the next not every model type is supported.

Therefore the solution that we gave was using the library “micromlgen” which can import an Scikit-Learn model and translate the model directly to C code.

Preprocessing Replication

The second major difficulty we faced on the process of model deployment was the fact that after recording new measurements and before passing the values to our model to get a prediction of the class, we need to apply the same preprocessing techniques we applied during the training, otherwise, it would be risky to rely on the values we convert from raw to units of measurement. To accomplish this, the steps from which we ought to pass any newly gathered data are, according to our best performing model are the following:

- Normalisation. And more specifically with an algorithm the results of which would resemble the robust normalisation.
- PCA. This part is not only challenging from the view of implementation but it is also computationally expensive.
- Binarization. This is probably the easiest part of this process, however we would need to first inspect the bins used by the Pycaret setup before applying it on the Arduino sketch.

Overall this process seems complex and it probably deserves another chapter on its own, but in any way, it is unavoidable, regardless of the solution we would give to the first issue of model transformation to a smaller one or hosting the model outside of the microcontroller.

Chapter 5. Results

Results

The overall results regarding the outcome of the project are recorded in the next chapter of conclusions, however if we would like to observe some solid results we could inspect the performance metrics of the binary classifier which, as previously stated, present promising results.

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
Logistic Regression	0.9674	0.9954	0.9601	0.9716	0.9658	0.9346	0.9347	0.2220
SVM - Linear Kernel	0.9659	0.0000	0.9640	0.9651	0.9645	0.9316	0.9318	0.2610
Quadratic Discriminant Analysis	0.9635	0.9951	0.9784	0.9475	0.9627	0.9271	0.9275	0.2680
Random Forest Classifier	0.9612	0.9914	0.9515	0.9673	0.9593	0.9222	0.9224	0.8720
K Neighbors Classifier	0.9600	0.9819	0.9518	0.9646	0.9581	0.9198	0.9200	0.5850
Gradient Boosting Classifier	0.9589	0.9937	0.9516	0.9626	0.9570	0.9177	0.9178	0.9230
Extra Trees Classifier	0.9589	0.9933	0.9472	0.9667	0.9568	0.9177	0.9179	0.6540
Ada Boost Classifier	0.9565	0.9918	0.9444	0.9644	0.9542	0.9127	0.9130	0.5030
Light Gradient Boosting Machine	0.9521	0.9923	0.9350	0.9642	0.9494	0.9039	0.9043	0.5090
Decision Tree Classifier	0.9504	0.9498	0.9357	0.9601	0.9477	0.9005	0.9009	0.2980
Ridge Classifier	0.9385	0.0000	0.8821	0.9888	0.9323	0.8764	0.8812	0.2420
Linear Discriminant Analysis	0.9385	0.9939	0.8821	0.9888	0.9323	0.8764	0.8812	0.2520
Naive Bayes	0.8857	0.9641	0.8656	0.8932	0.8792	0.7707	0.7711	0.3020
Dummy Classifier	0.5195	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2600

Conclusions

To conclude, our initial purpose was to investigate the hypothesis based on which one could use PPG signals and other features like temperature to predict the blood glucose levels, from the perspective of designing a medical device end-to-end and most importantly to explore the challenges along the process.

As a result of this exploration one thing that became obvious from the start was the fact that the dataset we used, at least to the extent we used it, was not enough to allow us to perform regression in order to get specific values. Furthermore, the implementation of multiclass classification, to use cut-off values from the literature, appeared to have some questionable results that resembled overfitting, but we also keep open the possibility of some misconfiguration from our side. And most importantly we managed to train a binary classification model with an arbitrary cut-off value, the logic of which for its selection is described in the relevant chapter of methodology, which performed sufficiently enough to

say that it probably is possible to use the heart rate from PPG signals and the temperature of a subject to classify its blood glucose level as over or under 114 mg/dL.

Along the path to setup the project from conception to actually using a ML model on the microcontroller we tackled a lot of challenges such as reading from multiple sensors of different kind simultaneously, using actuator parts such as an LCD screen to display results and of course finding a way to include the ML model on the Arduino MKR having the memory limitation.

Of course not everything was resolved and the question still remains on what is the best practice when it comes to applying the preprocessing steps on new data before applying them on the prediction function of the model.

Challenges

One of the most important challenges we faced was regarding the heart rate sensor of the initial hardware configuration we attempted, the KY-039. The sensor proved to give inconsistent readings very often and lacked to give repeatable measurements when the main microcontroller was swapped from Arduino MKR to Arduino UNO, which could be a result of wrong circuit configuration but in any case, it was giving inconsistent readings even for a specific microcontroller from one attempt to another. That was the main reason for changing this heart rate sensor with the MAX30100, which appears to resolve these issues.

Another important challenge was the fact that for the phase of gathering data with our device and with the glucose metre to then match the results and create a dataset would have been extremely time consuming. The viable solution we came up with was to find an existing dataset that would cover the specifications of our project.

And of course as stated in the model deployment chapter, a major challenge had to do with finding a way to incorporate the ML model into the microcontroller, mainly due to size limitations.

Apart from these resolved issues we also faced the problem of porting and applying the used preprocessing methods from the Jupyter Notebook to the C++.

Supplements

The Arduino code, the SQL transformation scripts, the dataset after ending in the final view, the Jupyter Notebooks and the trained models, are all in our repository https://github.com/karapapas/bme12arduino/

References

- National Center for Biotechnology Information (US). (2011). PubChem Compound Summary for CID 5793, Nitric oxide. In PubChem Compound Database. https://www.ncbi.nlm.nih.gov/books/NBK430900/
- American Diabetes Association. (2020). Standards of Medical Care in Diabetes—2020 Abridged for Primary Care Providers. Clinical Diabetes, 38(1), 10-38. https://clinical.diabetesjournals.org/content/38/1/10
- World Health Organization. (n.d.). Global Health Observatory (GHO) data: Blood glucose (expected values). https://www.who.int/data/gho/indicator-metadata-registry/imr-details/2380
- Ghiasi, S. M., Dahllof, M. S., Osmai, Y., Osmai, M., Jakobsen, K. K., Aivazidis, A., & Pociot, F. (2016). Regulation of the β-cell inflammasome and contribution to stress-induced cellular dysfunction and apoptosis. Molecular and Cellular Endocrinology, 434, 57-65. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4892884/
- aTrain Education. (n.d.). The Basics of Diabetes: A Knowledge Primer. Retrieved from https://www.atrainceu.com/content/4-regulation-blood-glucose
- Eva Tsalikian(2004). Accuracy of the GlucoWatch G2 Biographer and the Continuous Glucose Monitoring System During Hypoglycemia. Experience of the Diabetes Research in Children Network (DirecNet)
- Cho, P., Kim, J., Bent, B., & Dunn, J. (2023). BIG IDEAs Lab Glycemic Variability and Wearable Device Data (version 1.1.1). PhysioNet. https://doi.org/10.13026/73s9-cw03.