TempSense

A Project Report
Presented to
The Faculty of the Computer Engineering Department

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Bachelor of Science in Computer Engineering

By
Jiaqi Feng
Joao Carlos Frota Matos Prado
Jonathan Go-Oco
Vinayan Kathiresan
07/2021

APPROVED FOR THE COLLEGE OF ENGINEERING

_____

Kaikai Liu, Project Advisor

_____

Professor Rod Fatoohi, Instructor

_____

Dr. Xiao Su, Computer Engineering Department Chair

ABSTRACT

TempSense

By Feng, Jiaqi; Frota Matos Prado, Joao Carlos; Go-Oco, Jonathan; Kathiresan, Vinayan

In 2020, the entire world faced the biggest pandemic in the last 100 years, caused by the Sars-Covid-19 virus. The disease, commonly known as Coronavirus or Covid-19, causes respiratory problems to the victim, and although the mortality rate of this disease is low,, the contamination rate is very high. The spreading rate of the Sars-Covid-19 virus was so high, that in less than three months the entire world was having to face the virus which started in Wuhan, China.

The high spread rate of Covid-19 is due to the fact that during the first week of infection there are almost no symptoms. Furthermore, there is the concern of failure of the health system due to the large scale of people being infected such a situation that has happened in Italy, which the government of Italy to start a country-wide lockdown, prohibiting the citizens to leave their houses unless for strictly necessary needs, such as buying food and going to the hospital. Furthermore, as the virus expanded and quickly became the biggest pandemic we have ever seen, more countries faced similar situations to the one seen in Italy and country-wide lockdown became the quickest and most efficient way to mitigate the spread of the virus. With countries like the United States and Brazil reaching over 500,000 deaths by Covid-19 in 12 months.

To help the world recover and come back to normalcy, our project has been focused on finding a fast and easy way to measure if people are potentially affected by the Covid-19 virus, and can be expanded onto tracking the health of any user to check for symptoms of other possible ailments and diseases. Among the most common symptoms are fever and difficulty breathing or shortness of breath, which causes the blood oxygen rate to be lower than normal. Therefore, our group has developed an affordable device that has the ability to check temperature and blood saturation levels as well as notify if any anomaly has been detected.

Acknowledgments

We would like to express our gratitude to our professor Rod Fatoohi and our advisor Kaikai Liu, who showed great patience and affection to the students involved with this project. Especially through the struggles we faced with our lack of experience and difficulties caused due to the worldwide distance between us all, and thanks to their help this project was made possible.

Table of Contents

References

K. H. Khalid H. Almitani, “https://marz.kau.edu.sa/Files/320/Researches/70650_43625.pdf,” journal of King Abdulaziz University Engineering Sciences, vol. 28, no. 1, pp. 67–90, 2017.

S. Rost and H. Balakrishnan, "Memento: A Health Monitoring System for Wireless Sensor Networks," 2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, 2006, pp. 575-584, doi: 10.1109/SAHCN.2006.288514.

Shahriyar R., Bari M.F., Kundu G., Ahamed S.I., Akbar M.M. (2010) Intelligent Mobile Health Monitoring System (IMHMS). In: Kostkova P. (eds) Electronic Healthcare. eHealth 2009. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 27. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-11745-9_2

Valsalan, P., Baomar, T. A. B., & Baabood, A. H. O. (2020). IOT BASED HEALTH MONITORING SYSTEM. Journal of Critical Reviews, 7(04). https://doi.org/10.31838/jcr.07.04.137

Zheng YL, Ding XR, Poon CC, Lo BP, Zhang H, Zhou XL, Yang GZ, Zhao N, Zhang YT. Unobtrusive sensing and wearable devices for health informatics. IEEE Trans Biomed Eng. 2014 May;61(5):1538-54. doi: 10.1109/TBME.2014.2309951. PMID: 24759283; PMCID: PMC7176476.

Appendix

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Project Goals and Objectives

The goal of this project is to help increase awareness of health issues that can be derived from diseases such as Covid-19. We aim to do that by first providing a method for users to easily measure information about their bodies, such as temperature and blood saturation. And second, to provide the data measured in a way where the user can easily understand what it means. For example, it is known that Covid-19 can reduce blood saturation, but unfortunately blood saturation is not usually measured, therefore people affected by a respiratory disease are unaware of the low oxygen rate in their blood until other symptoms, such as fever, start showing up. Therefore, the purpose of this device will be to help users keep track of their health and log any possible changes that may indicate illness or symptoms of various medical conditions.

1.2 Problem and Motivation

Nowadays, people forget to check their own health regularly. Usually because they are so focused on their own problems, such as work and school, that there is no time left for themselves. Furthermore, in 2019, a new disease called the Covid-19, which started in China, quickly spread through the world and became the biggest pandemic of the last 100 years. Because the Covid-19 is a respiratory disease that affects the lungs, there are a few measures that can be easily tested in order to help notice any symptoms. For example, the aforementioned tracking of blood saturation, which is not normally done and could help a potential patient of Covid-19 to realise there is something wrong even before more prominent symptoms show up, such as loss of smell and taste.

Knowing this we proposed this project, which aims to help people track their health conditions in a more accessible and easier way. As people use our device and mobile app, they will consciously try to improve their health, and this will help motivate our users to make healthier choices, and go seek medical attention when signs that they can physically take note of are occuring.

1.3 Project Application and Impact

This project is meant to be applied to the general population. Anyone with access to a smartphone and internet will be able to use it and benefit from it. The idea for this project is to help any individual that wants to improve their overall wellbeing. Our aim is to create a positive impact in people's life by making them more aware of their health and to help create awareness of symptoms caused by respiratory diseases.

Since the method for data analysis will be simple and easy, we believe that there will be a constant and large stream of data. With this amount of data we will be able to provide a larger insight regarding a patient's health status, and reduce the time it takes for users to realise they have been contaminated. This will cause a positive impact in the life of our users, because these people will be able to quarantine quicker and alert the people they got in contact with about the situation. By doing so we will be reducing the chance of Covid-19 spreading and any future diseases that share similar symptoms.

1.4 Project Results and Deliverables

With the conclusion of this project, we made a device that is able to be placed anywhere with internet connectivity that can communicate with a cloud database to store the information measured. Furthermore, the user also has access to all measurements through their mobile app. Allowing for an easy way for users to measure some critical information about their health that might be a concern for respiratory diseases.

Furthermore, there is a finished prototype showing how the device is expected to run, as well as how it connects with a smartphone through a mobile app. Furthermore, there is this project report which is a collection of the research made by the team members that help make it a reality, with detailed information about the components we chose on the hardware side, the reasoning behind our choices, and how they work. On the software side there is the explanation of how we were able to develop a cross platform mobile app and what dependencies were used

1.5 Project Report Structure

This report will have 6 sections that will discuss the technical approach of developing the mobile app and the measuring device, as well as how the connection between them was established through a cloud database.

Chapter 2 will discuss the background research that was made prior to starting the project as well as courses and studies made that provided knowledge that can be applied. And lastly, a state-of-the-art summary that related to all the literature researched to aid us with this project.

Chapter 3 is about the requirements that we set for this project. We will describe the requirements for the system, interface, and software functionalities, as well as non-functional requirements. This chapter will describe how our project should work in a suitable environment.

Chapter 4 is a description of our system design. It will contain information about the architecture, product design, modules, interfaces, and data for our project to achieve the requirements mentioned in chapter 3.

Chapter 5 will detail the implementation of our system, including both software and hardware. Describing which dependency was used, as well as why we chose to use it and how they apply to the project. Lastly, we will show how these dependencies helped us solve problems as well as what challenges we overcame.

Chapter 6 will list the tools used in both the hardware and software side of our project. Describe how these tools were used and with what purpose, as well as standards used in this project.

Chapter 7 is the testing results we gathered throughout the experiments made during the development of both the measuring device and the mobile application. With a description of how we approach these tests and the analysis of the results we got.

Lastly, the 8th chapter will be our concluding statements and expectations for our future works.

Chapter 2. Background and Related Work

2.1 Background and Used Technologies

Because the entirety of this project has been conducted during the pandemic caused by the Covid-19, the people involved in it are separated around the world. Due to that, we decided to make this project entirely asynchronous. It is nearly impossible to keep everybody working in a synchronous way while in three different time zones, as the members of this team are spread between California, Brazil and India, especially including the many travel bans that happened in 2020, which impossibilitate the team to meet up in person.
We estimate that while unable to meet in person and work at the same time we would be able to deliver the project in a timely manner, as well as complete the entirety of our aforementioned goals. Unfortunately, it will limit the amount of people that will have access to the hardware portion of the project, and might cause some delay in response when communicating with our peers.

The technologies used in this project are Flutter, RaspberryPi, and Firebase. Flutter is used to help develop the mobile app. Flutter is an open-source tool that allows for cross-platform development, which will allow us to make the app for both Android and iOS platforms at the same time. On the hardware side we used Raspberry Pi to create a sensor that can measure the oxygen saturation and temperature of the user. There are plenty of sensors that were designed to be used with Raspberry Pi and will be later discussed in chapter 6. Finally, to connect the Raspberry Pi to the mobile app we will be using Firebased, which is a platform created by Google that helps with mobile and web applications. For this project it will be used as a live database that will be updated by the Raspberry Pi when measured and allow for the mobile app to retrieve the data in a timely manner.

Lastly, we will be using the knowledge we acquired during our college career in IoT, Embedded Systems, OOP, and Data Structures to help implement both the software side as well as the hardware. Furthermore, we had to learn how to handle technologies we have not worked with yet, such as how RaspberryPi works and how it handles the connection with its peripherals, and how to properly send the information to Firebase, as well as how to make flutter connect to the same database and present the information in a friendly way for the user.

2.2 Literature Search

Recently, with the advancement of technology, health monitoring systems have become more common among society. They are a great way to guarantee that people who have frail health stay aware of their conditions. Moreover, due to the Covid-19 pandemic people are more interested in the topic of health than ever before. Usually, when this topic comes up the first thought is what's more common to us such as Apple Watch’s heart rate sensor, or “stand-up” alarms. But there are more robust systems that can provide health monitoring that is more relatable to professional medical use. One of these examples is called Memento.

Memento was developed by S. Rost and H. Balakrishnan (2006), and it is a health monitoring network that is able to provide symptoms alert while maintaining a robust network that is fail-proof against packet loss, while still maintaining a lightweight bandwidth. Another great example is IMHMS (Shahriyar R., Bari et al, 2010), which stands for Intelligent Mobile Health

Monitoring System. A health monitoring system that was made due to the realisation of how well integrated mobile computers have become to our daily lives, and how powerful and small it was possible to make sensors that would measure information about the user. Furthermore, IMHMS allows for communication between doctors and patients. Furthermore, Valsalan et al. (2020), proposed an IOT Based Health Monitoring System that goes even further than just recording the patient's basic health status, but it also records room temperature and humidity, and shares it with a doctor, allowing for simple distant diagnosis. Lastly, Zheng et al. (2014) proposes an integration between wearable sensors to the environment. Such as simple things like a door handle with temperature sensors that can indicate if the user has a fever or not, or more complex solutions such as tattoos that have sensors and can keep monitoring 24/7 for a long term.

In conclusion, these articles helped us greatly to understand the technological advance of IoT in healthcare, and all the vast possibilities we have in the future, to achieve a position where a doctor's appointment will be just a message. Furthermore, these articles aided us to choose the best course of actions we should take when approaching the problem we are trying to solve with this project. Especially since like Valsalan et al. (2020), our inspiration for this project came with the Covid-19 pandemic and the desire to improve people's lives.

2.3 State-of-the-art

There are many innovative techniques being used to improve our everyday lives. Many of these rely on manipulating data that is gathering through sensors. For health care there are many aspects of our body that are important to measure especially during this pandemic, such as oxygen saturation, heart rate, and temperature.

There are development sensors that will measure the level of oxygen presented in the patient's breath, or infrared cameras that can follow the subjects movement and measure their temperature.Furthermore, there are sensors that could be installed in the bedsheets to monitor the user heart rate during sleep, or handles with sensors installed that can measure not only temperature but also heart rate through the palm of their hands. As well as sensors that are installed in the toilet seat that allow for weight measurement, and urine and feces weight.

But all of these data collection sensors are useless without the correct analysis. To solve this problem we have developed artificial intelligence that can cover the procedures of data analysis with big data. These AIs can receive all of the data from the sensors and manipulate it, through a process of selecting the relevant chunk of data, update it with newer measurements, and comparing with the old ones, it is possible to generate AI based reports on the patients health as well as alert for any critical conditions that are detected, as well as provide recommendations to what can be done to improve the patient's health.

According to Business Insider, the Internet of Medical Things(IoMT) is going to be valued as a $158 billion industry in 2022.

One of the primary uses is the ability to run data analytics on the vitals in the cloud to be proactive and predict health problems. This was previously done by doctors looking at charts and patient history, which is arguably not as reliable as a computer crunching numbers and making calculated predictions instead of doctors making educated guesses based on their experience, which also varies from doctor to doctor.

The well known IoMT devices are wearables like the Fitbit, Apple Watch, Samsung Galaxy watch, etc. We are not trying to compete with these devices as we are building a clinical grade device and not a wearable that uses estimates to track your health and fitness.

Currently, clinical grade IoMT is primarily used in larger devices such as smart medical beds, electrocardiograms and ultrasound machines. These machines make health information available to doctors and nurses wherever they are over the internet. They upload the vitals they have to the cloud, just like our device will.

A few IoMT startups such as Clover Health and Sensely attempt to provide or reduce the cost of health insurance using IoT, or is attempting to provide over the air access to doctors such as Babylon Health and Genoox

Others are trying to predict health based on DNA sequences such as Helix, Karius

There are a few startups that have similar goals to us:

- Neurotech which is developing a clinical grade IoMT ECG
- Pear Therapeutics which discovers, develops, and delivers clinically validated software to provide better outcomes for patients, smarter engagement and tracking tools for clinicians.
- AliveCor which is a medical device and artificial intelligence company that sells ECG hardware and software for consumer mobile devices

Chapter 3. Project Requirements

3.1 Domain and Business Requirements

Health issues have been significant since the beginning of Covid-19 pandemic. People have been realizing that we need more attention to small details about our body. We need to be alert when there's something that seems unusual as soon as possible, even though things are just a little bit off, we should proceed to investigate. There are options out there for monitoring health conditions, Fitbit and Apple Watch, but they are not cheap and easily affordable. Therefore we proposed to have this project, TempSense, so that people can be easily alerted about their health conditions. This device should be cost-efficient and easy to use.

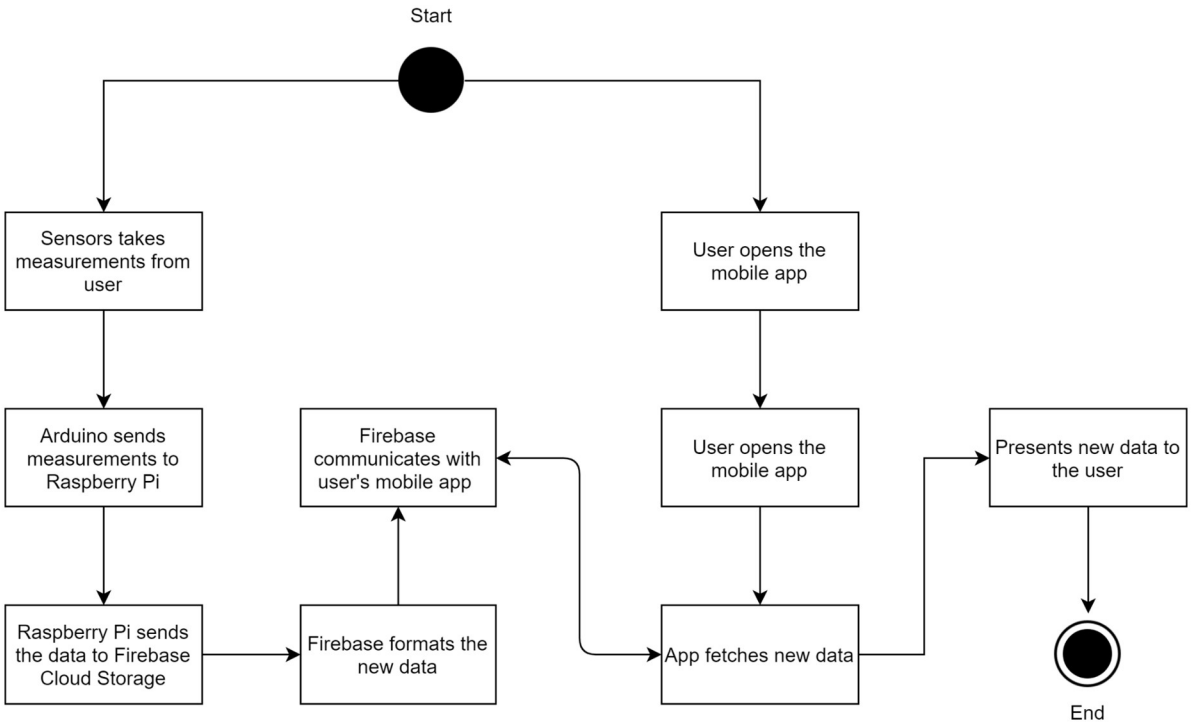
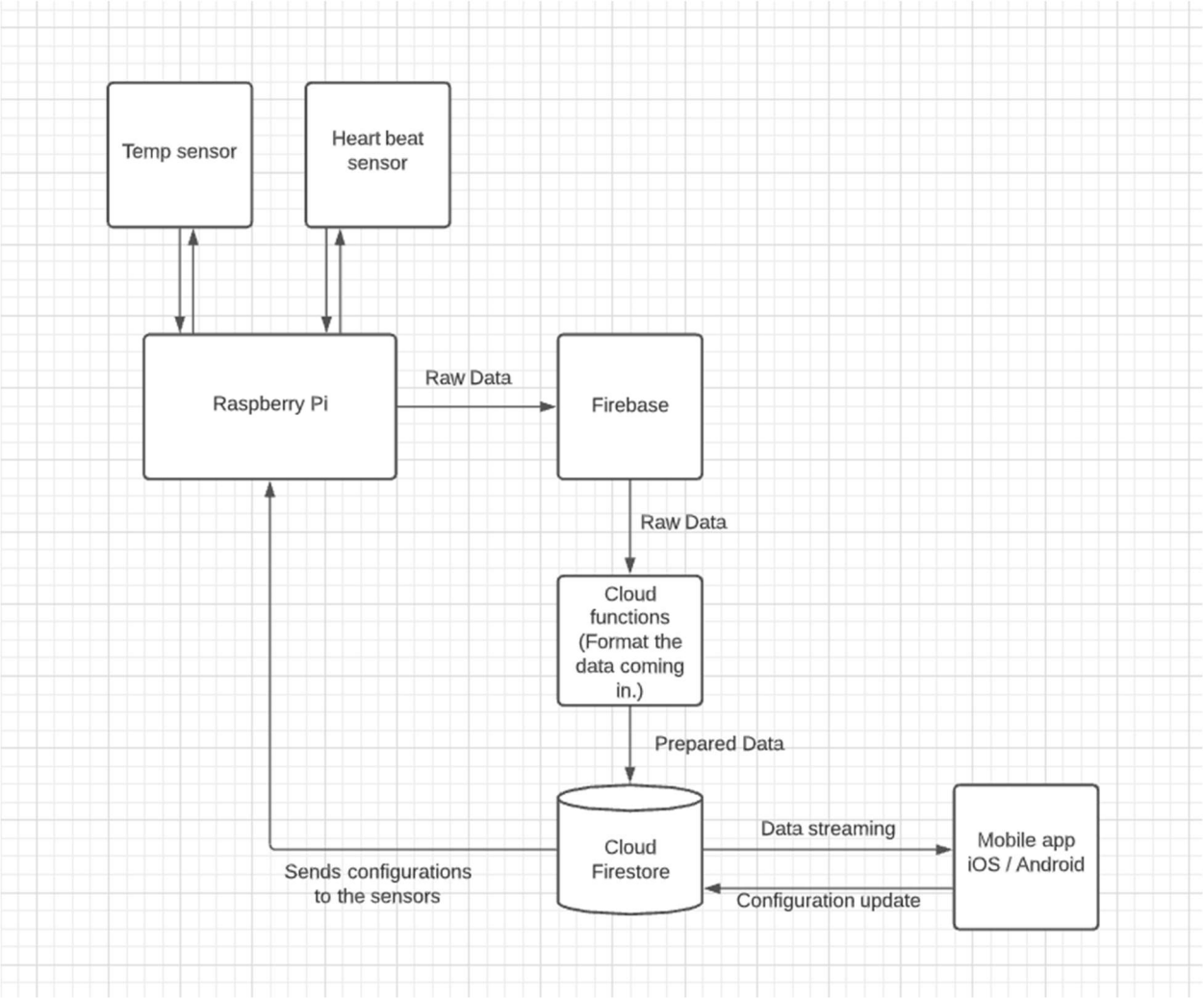


Figure 1. UML Activity Diagram


Temp sensor
Heart beat sensor
Raspberry Pi
Raw Data
Firebase
Raw Data
Cloud functions (Format the data coming in.)
Prepared Data
Cloud Firestore
Data streaming
Mobile app iOS / Android
Sends configurations to the sensors
Configuration update

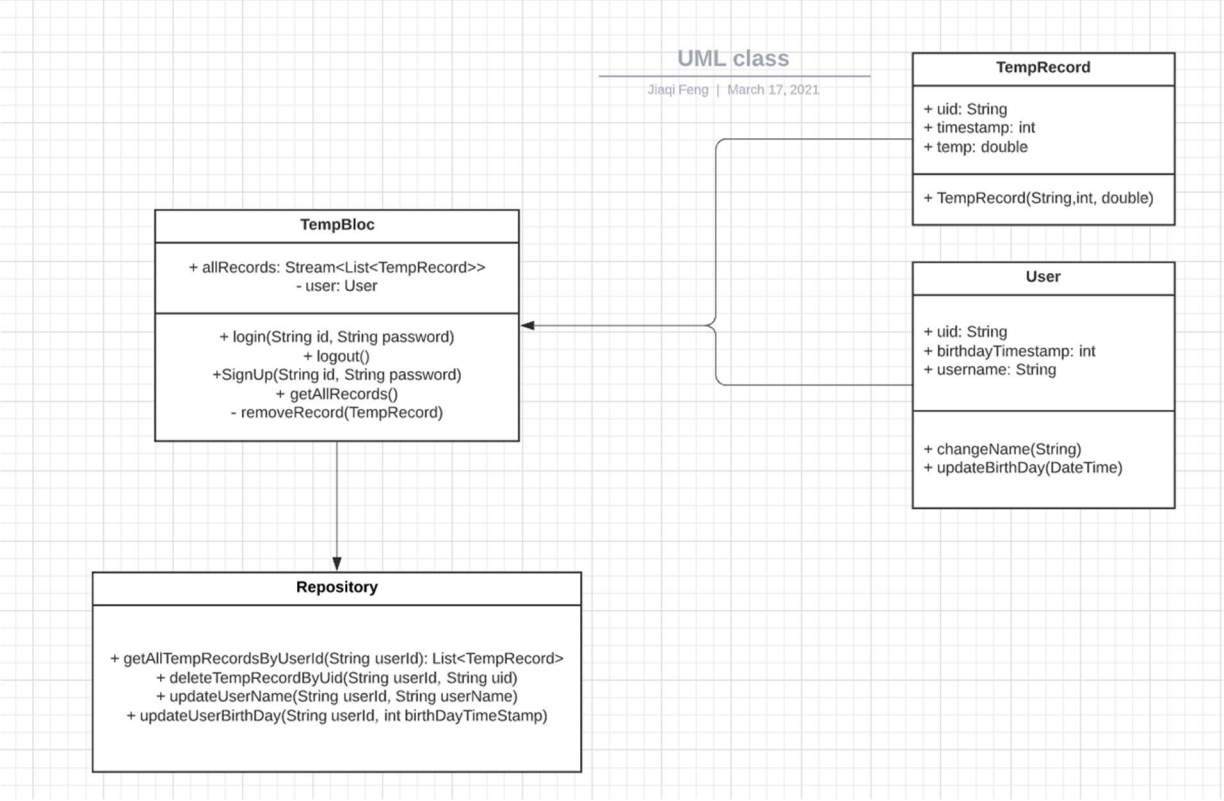


Figure 2. Overview of mobile application structure

3.2 System functional requirements

a. Devices should be able to constantly detect their body temperature using the sensors.
b. Users should be able to see their body temperature history from their mobile phone.
c. Users should be able to see their recent body history from their mobile phone.
d. Users should be able to change their personal information, in this case, their usernames.

3.3 Non-functional requirements

a. Devices should be reliable.
b. Device should be easy to set up.
c. Device should be running fine for a long time without human intervention.
d. App should be easy to understand and intuitive.

3.4 Context and interface requirements

a. App should have clear distinction between types of data.
b. App should have a list view to show the temperature history.
c. App should have a dialog to inform users in case of unusual temperature.
d. Device to measure data must be easy to use

3.5 Technology and resource requirements

a. Sensors must measure reliable information
b. Database must be able to store all information measured
c. Database must be accessible by Raspberry Pi and Mobile App
d. Measuring device and mobile app must not require a lot of power to use

Chapter 4. System Design

4.1 Architecture Design

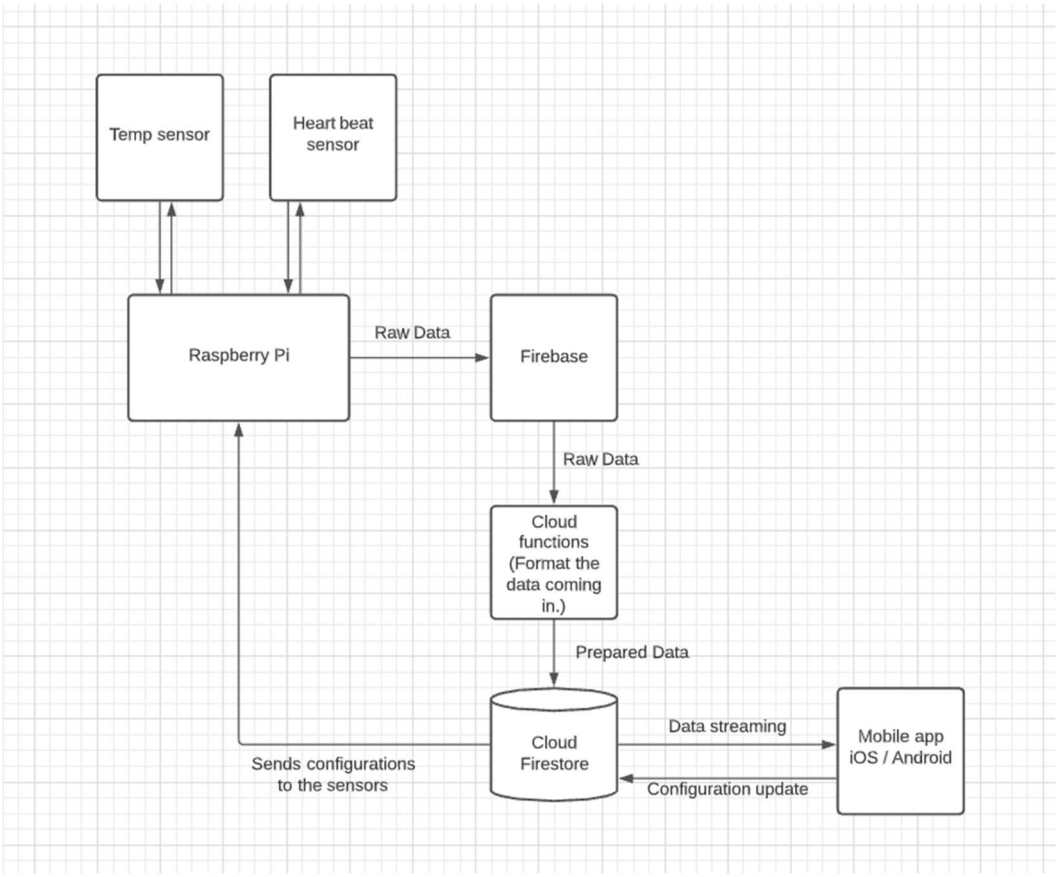


Figure 3. Overview of the TempSense structure.

Figure 3, is a visualization of the system structure. We have a Raspberry Pi as the hosting platform for the sensors, and those sensors will be collecting relevant data constantly and sending them raw to the cloud database, which is hosted using Google Firebase. The data doesn't immediately get stored to the collection, instead, it will first be passed to the cloud function hosted on Firebase to be prepared. The cloud function will filter and add relevant data such as timestamp to the raw data and convert it into json format and store it in the corresponding collection. Once the data is in the collection, the data will be streamed to the mobile client application.

In reverse, the mobile application will be able to change configurations of the device or other sorts of information. Such configurations are also stored in the Firebase. However, since it is a simple data structure, we don't need cloud functions to filter and prepare the data.

4.2 Interface and Component Design

4.2.1 User Interface

The mobile client application has an UI constructed using widgets provided by Flutter rendering framework. The client contains one screen that presents all the relevant data and information. There are several components in the main screen. On the very top, there is a simple app bar that displays very basic user information, namely, the username. Users can tap on it and make changes to it.

Below the username are a row of widgets. To the left is the recent temperature reading, it uses a bigger font because it has priority over other things. There will also be a date and time field indicating when the reading was recorded. To the right is the recent heart beat reading. Same as the temperature reading, it has a bigger font and a date and time field.

Next we have a line chart for the temperature where users can see the change of their body temperature during a period of time. Finally, we will have a listview for a more detailed temperature history presentation. All the temperature readings will be presented in this cell using the most intuitive way.

4.2.2 Components

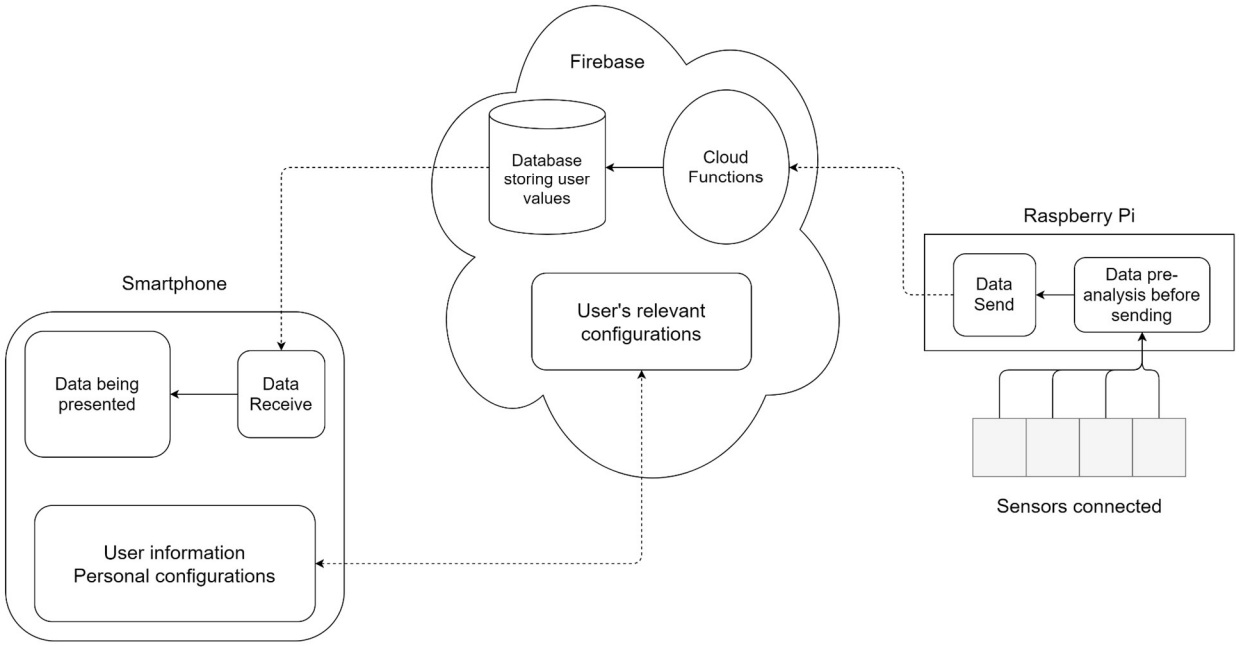


Figure 4. Diagram of components

The project is divided mainly in three pieces, the mobile app, the sensors, and the cloud database. Figure 4 is an illustration of where our project components will be connected, and is divided in the three pieces mentioned before. First, the mobile app will be designed using Dart and Flutter, because it provides a quicker way to develop both iOS and Android apps. Furthermore, the app also has a clean and nice UI where the user can analyse their own measurements, under the hood

there are modules that allow for user customization of their own relevant information, and that can communicate with the Firebase DB to receive the data measured by the sensors and send any modification that the user desires to their information.

Secondly, there is a Raspberry Pi coded in C++ and Python and connected to the many sensors that capture the user's health information. The Raspberry Pi connected to the sensors via a wire, inside the Raspberry Pi there are functions that allows for pre-analysis of the data to ensure that any outliers are excluded. For example, it would be impossible for a living human being to register a temperature of 33°C / 91.4°F, so in case of any such measures the Raspberry Pi will discard that information. After the Pre-analysis, the data measured through the sensors is sent to Firebase.

Finally, the Firebase provided with Google Cloud Computing analyses the data and embellishes it, so it is neatly stored in our database that will be accessed by the smartphone app, this data processing step is done using Javascript and Express.js. Furthermore, the Firebase also lets us store relevant user information that can be accessed and modified by the user itself.

4.3 Structure and Logic Design

[Present the detailed structure and logic design for your hardware/software components and processes. This section must include textual description accompanied with diagrams. If scientific or mathematical fundamentals are used for your project algorithm, specify what kind of formula or theory has been applied.]
For the temperature sensor, it is connected to an Arduino via 4 wires, a SDA, SCL, Power, and Ground. Ppon reading the datasheet and the example code given, we modified the code to send readings in Fahrenheit rather than the default Celsius temperatures the sensor gives. To do this, we used the formula (Reading in Celsius * 1.8) + 32 = Temperature reading in Fahrenheit. From the reading, we connect the Arduino to a USB port of the Raspberry Pi, which reads the serial output from the Arduino, and sends it over to Firebase using a HTTP Post Request.

We used a mathematical algorithm to calculate both the oxygen saturation and the heart rate from digital values(1s and 0s). For heart rate, we keep track of the increasing values to find the peak of the sine wave, and then of the decreasing values to fund the trough, Then using this we counted the number of heart beats in a given period of time and multiplied it to get the BPM. For the oxygen saturation, we used the previously found peak and trough values to find the ratio of the height of the wave to the value of the trough. This ratio is our slope(m) for a y=mx+c equation. x is -25 and c is 117. We started with 100 as our c value and then modified it to get values that were acceptable.

4.4 Design Constraints, Problems, Trade-offs, and Solutions

One of our constraints is an unreliable sensor that didn’t give consistent values. To combat this I am working on a software filter to make the reading more reliable. The sensor is light based, so I am also working on making a light proof enclosure that will also serve to immobilize the hand to get more accurate readings.

4.4.1 Design Constraints and Challenges

One of the main challenges we faced was the extent of our economic resources, because it affected our choice of peripherals to purchase, but more importantly which database to use. Currently the budget for this project had been about $50, which we had planned based on the price of peripherals. Because reliable sensors are more expensive, we had to choose cheaper versions which affect the reliability of the reading. Also, we had to factor in the price of the cloud storage we are using, which is Firebase, and we were caught surprised with the price it is required to maintain such a platform on Google Cloud and Firebase, which led us to realize that it's not only an economic price but also a scalability problem.

Furthermore, we face the question of privacy and information security, which is currently a very socially sensitive topic in the USA but also more cost that is added to maintain our project. Because we need to ensure that the data can be accessed by the user and the user only, and ensure that the data will not be sold to third parties for any reasons or profits. We also need to protect the user data from potential threats such towards our database. If there were no financial concerns we could learn from many messaging apps such as Telegram, WhatsApp, or Signal, which are a chat application that uses end-to-end encryption on their platform to protect their user-generated data from threats. Implementing end-to-end encryption would be very useful for our situation, as it is famous for protecting data from man-in-the-middle attacks, which is when the attacker pretends to be the legitimate user and receives the data it’s not supposed to get access to.

4.4.2 Design Solutions and Trade-offs

Using Google Cloud and Firebase, although expensive, does have its advantages. For example, storing data on Google Cloud and Firebase might make the software development process easy but might not scale well if the user base grows unexpectedly in the future. Unfortunately, as mentioned our budget is not enough to cover the amount it would cost for the amount of reading and writing per day it would require for a app like this which continuously tracks the health data of the user, and we should probably in the future transition to other data storage platform like Amazon Web Services or even Microsoft Azure.

For the user-generated content protection, we want to make sure that there are secure access rules implemented on the Firebase console so that only the admin and users can get access to the user-generated data. We might also set up user privacy agreements to inform users of our privacy policy so that users can have peace in mind while using this app.

.

Chapter 5. System Implementation

5.1 Implementation Overview

For the hardware, we are using two sensors, a Heart Rate and Oxygen sensor EC-0567, and a Adafruit TMP117 ±0.1°C High Accuracy I2C Temperature Sensor, and recording that data through an Arduino and a Raspberry Pi and sending that data to the database using HTTP Post Requests.

We have connected the Heart Rate sensor to the Raspberry Pi to receive the data and to convert it to readable values of BPM and SPO2 using mathematical algorithms.

On the software side, we have developed a functional mobile app that allows the user to see the data measured. This app was built using Flutter, which was developed by Google and allows for cross platform development from a single codebase. Flutter is easy to install and it can be used with many different editors of our choice, which allowed us to keep using the editor we were more comfortable with. While Flutter is just the platform, the language used to develop the app was Dart, which was somewhat familiar to us since it has a similar syntax to C/C++, and reduced the learning curve to understand how it works. Unlike what would have happened in case we started using Kotlin, a language designed for Android, or Swift, designed for iOS.

Next part of the software side is the Cloud Functions used on Firebase that allow us to edit the data coming in from the sensors and store them into the database. As our data is sent using HTTP it triggers the cloud function to run a JavaScript that will modify the incoming data and store it in the Firestore database in an organised way. Connected to that is the Firestore database, which is hosted on Google's Firebase as well. Firestore is a NoSQL scalable database, that unlike SQL, is non-relational, it does not need to be stored in tables, and is scalable. Furthermore, we chose to be constant and use Google's frameworks through the entire project to keep developing easier and avoid running through connectivity problems between different services.

5.2 Implementation of Developed Solutions

The temperature sensor measures the temperature in Celsius, and for American users as well as our team, which knows temperature in Fahrenheit, we then convert the readings to Fahrenheit before sending the data to Firebase. In the code, we did this by implementing the formula $(Celsius\ Reading\ *\ 1.8)\ +\ 32\ =\ Fahrenheit\ Reading$, shown in Table 1 below.

Table 1. tempsensetest.ino snippet showing formula of converting Celsius readings to Fahrenheit
f = ((reading.temperature) * 1.8) + 32;

For the temperature sensor, the data is gathered in an Arduino Mega 2560, which sends each reading over to the Raspberry Pi through Serial. Once the Raspberry Pi receives the data, we check to see if there is an internet connection. If one is established, we send the data over HTTP

Post Request, as shown in Table 2. The Heart Rate and Oxygen Saturation sensor's data readings are sent in the same way, but with different variables being sent and different URLs.

Table 2. TempSenseHTTPPost.py snippet, showing how data is sent over HTTP Post

```
if test_internet():
        reading = {'temp' : line}
        r = requests.post(url, data = reading)
        print(r.text)
```

After the data is sent to Firebase using HTTP post method, the cloud function will take the value and put it into the corresponding collection, and at the same time, the value will be paired with a timestamp indicating when the value was uploaded. Once the data is sent to Firebase, Cloud Functions do the heavier work as shown in Table 3, which shows how we managed to receive and modify the HTTP post request.

Table 3. index.js file snippet of Cloud Function modifying temperature data and storing it.

```
exports.addTemp = functions.https.onRequest(async (req, res) => {
   // Grab the text parameter.
   const original = req.body;
   functions.logger.log("Hello from info. Here's an object:", original);
   var map = req.body
     // Push the new message into Firestore using the Firebase Admin SDK, and        add date
     and time information to the table.
   const writeResult = await admin.firestore().collection('temp').add({'temp': map['temp'],
'timestamp': Date.now()});
   // Send back a message that we've successfully written the message
   res.json({result: `Temp with ID: ${writeResult.id} added.`});
 });
```

The app then fetches the temperature values and presents them nicely using a chart and list view. We used a mathematical algorithm to calculate both the oxygen saturation and the heart rate from digital values(1s and 0s). For heart rate, we keep track of the increasing values to find the peak of the sine wave, and then of the decreasing values to fund the trough. Then using this we counted the number of heart beats in a given period of time and multiplied it to get the BPM. For the oxygen saturation, we used the previously found peak and trough values to find the ratio of the height of the wave to the value of the trough. This ratio is our slope(m) for a y=mx+c equation. x is -25 and c is 117. We started with 100 as our c value and then modified it to get values that were acceptable.

The mobile application, developed using Flutter, followed a design pattern called BloC pattern which is quite popular in the Flutter community for its simplicity and great results, and is similar to the well known MVVM pattern. The difference here is that BloC combines view model and

view controller into one BloC class. The app thus has three components: bloc, model, and repository. BloC is where we store the business logic, things like initializing the data or filtering data happens here in BloC. Repository is where we connect to Firebase API and fetch data collected using the Raspberry Pi. The current version of the TempSense mobile application is shown below in Figure 5.

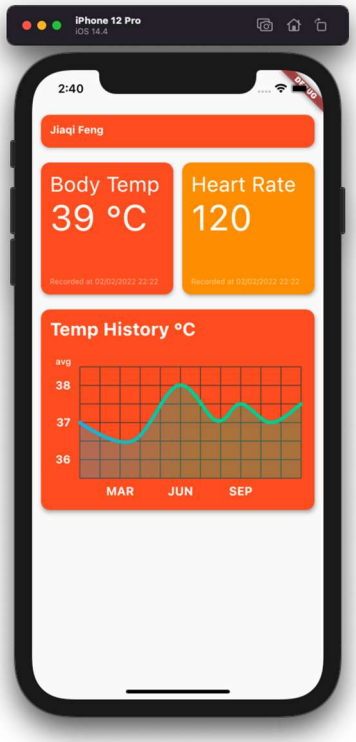


Figure 5. TempSense Mobile Application

5.3 Implementation Problems, Challenges, and Lesson Learned

The biggest problem of our set up is the inconsistent values we received from the sensor. This required a lot of trial and error debugging to filter out the values that were not correct.

The algorithm to find the heart rate and oxygen saturation was also a challenge to perfect as the equation varies sensor by sensor, and the lower quality sensor that we used didn't give consistent results.

We are working on an enclosure to house the sensor. It is light based so we think that if we keep it in a dark enclosure we will get more consistent results. The enclosure will also serve as a consistent way to immobilize the user's finger while the readings are taken.

To combat a lot of these issues, we found that putting less pressure on the sensor yielded more accurate results, which had us conclude that the sensor was sensitive to the pressure on it.

For the temperature sensor, we originally planned to use a thermal camera, but we found the data not accurate enough to measure the body temperature of a person, so we switched to a temperature sensor that requires one to touch to measure the temperature instead.

For the software part, namely the mobile application, we have one team member who has experience in Flutter so there wasn't any problems or challenges there, however, for the backend applications, none of us have enough experience therefore we went on and decided to use Firebase instead of setting up a database on AWS or Google Cloud Platform which saved us a lot of time and efforts. We also had trouble building the connection between the server and the hardware sensors. We ended up going with Firebase cloud function which provides us a way to enable sensors to send data using HTTP calls.

Chapter 6. Tools and Standards

6.1. Tools Used

For the hardware side of this project, we are using two sensors, an Arduino, and a Raspberry Pi to send the recorded data to Google Firebase. The sensors used are the Heart Rate and Oxygen Sensor EC-0567, shown connected to the Raspberry Pi in Figure 6 and a Adafruit TMP117 ±0.1°C High Accuracy I2C Temperature Sensor, shown connected to the Arduino Mega 2560 in Figure 7.

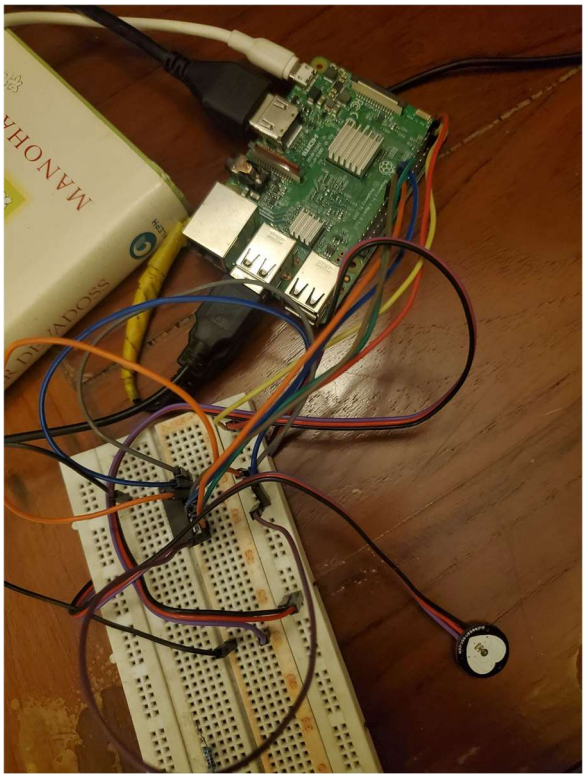


Figure 6. Heart Rate and Oxygen Sensor EC-0567

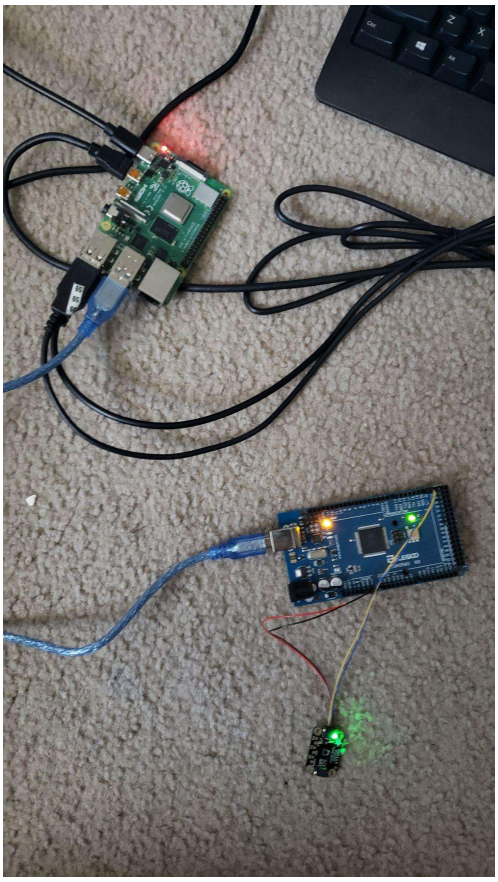


Figure 7. Adafruit TMP117 ±0.1°C High Accuracy I2C Temperature Sensor

With the Temperature sensor, we connected the device to an Arduino, as we found the Raspberry Pi implementation to be difficult due to the lack of experience and the easier implementation directions as shown by the manufacturer of the sensor. Once we get the data from the sensor, we send it over through an HTTP Post Request using code made in Arduino, which sends it over to our Google Firebase where our app team can use it.

With the Pulse and Oxygen Saturation Sensor, we connected it directly to the Raspberry Pi through an Analog to Digital Converter. The data is filtered and sent to Google Firebase from the Pi using HTTP Post Request as well.

For implementing the sensors, we communicated regularly with the app team to discuss the best course of action on how to send the data through Canvas and Discord, where we concluded on which sensors would be best, and which database would be best to send.

For the software on the mobile, we are using Flutter which is a framework for developing cross platform mobile applications provided by Google to develop the mobile client application for presenting the data detected by the sensors. We are using Google Firebase for data streaming and storage, and Cloud function for receiving the data from the hardwares and storing them into the corresponding collections. For the functions hosted on Firebase for processing data, we are using Javascript and express.js framework because we don’t have any options.

Packages used in Flutter:

- fl_chart: We are using FL charts to create beautiful and intuitive visualizations of the temperature readings.
- firebase_core: We are using firebase core as a dependency for the cloud firestore.
- cloud_firestore: We are using cloud firestore package to access data stored in the Firebase.
- flutter_bloc: We are using flutter bloc packages to help make implementing bloc patterns a breeze.
- equatable: We are using equatable to make objects easily comparable.
- intl: We are using intl for utility functions such as date formatter.
- shared_preferences: We are using shared preferences to store user preferences on the device.

6.2. Standards

The standard we used for the Adafruit TMP117 ±0.1°C High Accuracy I2C Temperature Sensor was that we needed a sensor to measure body temperature accurately enough so that users would be able to log the temperature knowing that the temperature recorded was correct. For the Heart Rate and Oxygen sensor EC-0567, we wanted the sensor to accurately measure both the heart rate and Oxygen Saturation levels without major variance in values.

Flutter allows us to easily create an application to display the data recorded in the sensors to our users. Flutter's simplicity and open-source nature gives us the freedom to create an app to our customization standards, and make it easier for the user to understand that data they record from our sensors and from their own input.

Chapter 7. Testing and Experiment

7.1 Testing and Experiment Scope

The scope of the tests conducted by the team are to ensure that reliable information can be measured and communicated between the user's smartphone, measuring sensors, and Firebase. First thing we did was to test the sensors on the arduino, because our goal was to quickly ensure that the sensors would work and be reliable enough that outlier measurements could be solved on the software side.

We bought a Pulse Oximeter from a Pharmacy to check the accuracy of the heart rate and oxygen saturation. We compared the values from the Pulse Oximeter, treating these as the expected values for both the heart rate and oxygen saturation, and compared them to the readings taken from our sensor.

Next, we tested them on the Raspberry Pi and we also checked if there were any major changes between measurements done using arduino or Raspberry Pi. The Raspberry Pi was used to communicate with Firebase, and knowing that the sensors were working fine helped in debugging or catching any failed attempts where weird data could be sent. This will also test the reliability of using HTTP posts requests to send data from Raspberry Pi to Firebase.

7.2 Testing and Experiment Approach

First step was to test the sensors. Once we were able to ensure that they were working properly we started measuring the values they would gather during the day, which allowed us to calibrate them properly. For the temperature sensor, to test the accuracy of the measurements, we conducted a comparison of the expected value from a thermometer, and compared it to that of the sensor. We randomly tested the temperature against the thermometer to see if any variable change was significant enough to be an accurate tool to be used in measuring one's body temperature.

We compared the values we got for BPM and SPO2 to those from a Pulse Oximeter we bought from a Pharmacy. We got our values to be within the range of error by comparing these values and changing the mathematical equations used. We used a trial and error method to fix these equations.

Then we tested the communication with Firebase by sending the data measured and checking the DB tables to see if there were any modifications to them when we sent a HTTP Post request. In the Python scripts, we used the line in Figure "", which printed back a confirmation that data was sent to Firebase. At the same time, we tested the reliability of the mobile app to connect with our DB. Whenever new data was uploaded to Firebase, the app was opened to see if there was any update information. With this we were able to ensure that communication between the Raspberry Pi, Firebase and the mobile app was reliable.

7.3 Testing and Experiment Results and Analysis

In testing, we found that the Adafruit AMG8833 8x8 Thermal Camera Sensor, the original sensor we planned on using to capture body temperature, had difficulties in capturing the temperature of just the target, as it would gather each individual pixel and measure the average

temperature rather than a single point. We also found that when the Thermal Camera was closer to the target, while the temperature reading would be more accurate, the temperature could range 4 degrees higher or lower than the predicted temperature. For this reason, we switched to the Adafruit TMP117 ±0.1°C High Accuracy I2C Temperature Sensor, as we found it's accuracy to the ±0.1°C fell in standard to what we wanted out of this project.

To test the accuracy of the readings, we conducted 20 random readings comparing the value of a thermometer readings as our expected values, and the temperature sensor readings as our experimental values. The reported difference between the actual value and the recorded value for the temperature readings according to the temperature sensor's manufacturer was ±0.15°C. To test this, we found the mean of the differences in the values from and calculated the p value. The measured comparison between measurements is shown in table 2 below:

Table 2. Testing results of temperature sensor

Expected Temperature Reading from Thermometer (°C)	Recorded Temperature from Temperature Sensor (°C)	Difference
36.67	36.72	0.05
37.17	37.09	0.08
36.28	36.17	0.11
36.94	36.95	0.01
36.38	36.51	0.13
36.83	36.90	0.07
37.00	36.97	0.03
36.52	36.40	0.12
37.15	37.08	0.07
37.02	37.04	0.02
36.91	36.99	0.08
36.48	36.44	0.04
36.87	37.00	0.13
36.55	36.61	0.06
36.72	36.57	0.15

36.69	36.70	0.01
36.42	36.39	0.03
37.11	37.11	0.00
37.02	36.92	0.10
36.87	36.78	0.09

We found the mean of the difference between the expected temperature values and the recorded values to be 0.069 with a standard deviation of 0.045. In observing our values, we found the majority of our measured values to be within the 0.1°C range for error that the manufacturer. And while some values exceeded that range value, we believe this may be more to do with human error, with possible reasons being the readings not being taken at the exact moment of each other, with each being taken right after, as well as both the thermometer and the temperature sensor possibly being affected to the temperature outside of the body. To test the accuracy of the heart rate sensor and oxygen saturation we took 10 readings and compared them to a Pulse Oximeter we bought at a Pharmacy. The measured values are below in table 3 and 4 respectively.

Table 3. Testing results of heart rate sensor		
Expected Heart Rate(BPM)	Read Heart Rate(BPM)	Difference(BPM)
67	77	3
67	77	3
68	70	2
70	70	0
72	71	1
75	82	7
73	75	2
65	68	3
66	67	1

Table 4. Testing result of oximeter sensor		
Expected Oxygen Saturation(SPO2)	Read Oxygen Saturation(SPO2)	Difference(SPO2)
97	96	1
98	97	1
98	95	3
98	96	2
98	97	1
98	90	8
98	97	1
97	97	0
98	97	1

Apart from one off value in both measurements, our readings were fairly consistent and within the margin of error. For the heart rate sensor, we calculated the mean difference to be 3.7, and the standard deviation was 4.6. Meanwhile, the mean difference when measuring the oximeter was 1.9 and the standard deviation was 1.1. We assume that the larger difference is caused by the fact that these are cheaper sensors than the one used to measure body temperature. Although the measurements are larger, they are still within an acceptable range.

Chapter 8. Conclusion and Future Work

When planning ahead, we intend to use a better heart rate sensor to avoid bigger disparities between expected and measured values, as well as to increase the reliability of our project. Also, we will be using a digital sensor that processes data on it's own chip and has a more reliable connection such as I2C, unlike the one we have now that works with analog signals.

Furthermore, we plan that in the future our device could run data analytics based on the data measured and provide a more reliable monitoring of the user's health over periods of time to alert of any large deviation value that could potentially be a health issue or a serious threat to the user's health. Furthermore, it would be a great addition to implement machine learning to do this task, as it would be able to adapt to the user's habit.

We plan to have the temperature sensor fully integrated into the Raspberry Pi, rather than connected to an Arduino that sends the data over serial. Along with that, since during this project we were only able to do one sensor each, the next goal would be to connect the sensors such that there is only 1 Raspberry Pi needed for the final product.

While we were able to send the data over HTTP, we may switch over to the MQTT protocol, as MQTT may offer different advantages that HTTP may not have. Also, regarding the internet, while we have found a way to check if there is an internet connection before sending the data over, we have not yet implemented a way to save the data remotely when there is no internet, and then sync the data to Firebase once a connection is reestablished.

Moreover on the software side, we understand our app's limitations in the matter of UI customization and security, so we would increase the changes to the interface in order to make so the user has more control to what he would like to see in the main screen, and implement security features to provide more comfort to the user, such as encrypted log in features and end-to-end encryption of data, therefore providing more privacy and avoiding data leakage. We also may implement changes in which the TempSense mobile app itself will control the sensors and Raspberry Pi to record data, rather than just grabbing the data straight from Firebase.

In conclusion, this project was a great learning experience, we had some difficulties in getting the team in sync as we were all in different continents due to the Covid-19 pandemic, and we went through some troubles learning how to use technologies we had never experienced before, such as Raspberry Pi, Python, and Firebase. But we also believe that it was important that we had made the mistakes we made as they were caused by our lack of knowledge of how the industry works, and thanks to that we will be more prepared for the future.