GAMER AND AT-HOME OFFICE HEALTH MONITORING

TOBIAS SCHOU, 201908069
OLIVER LEMONAKIS, 201910301

GROUP: GOLF

PROJECT REPORT
December 2022
Advisor: Niels Olof Bouvin

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

GAMER AND AT-HOME OFFICE HEALTH MONITORING

LEMONAKIS & SVENNINGGAARD

Project Report

Department of Computer Science
Faculty of Natural Sciences
Aarhus University

December 2022

1 INTRODUCTION

Health monitoring systems have already been implemented in different industries in several different capacities. Health monitoring exists, of course, in the medical industry as a vital tool of treatment, but is also prevalent in the sports industry. More specifically as either accessories to the novice runner in order for them to improve their understanding of their body, or as a means for athletes to further increase their competitive performance.

Within the sports industry these products are most oftenly offered in the shape of Smartwatches or heartrate monitors that are strapped around the chest. Apple recently expanded upon their Smartwatch range with additional products providing utility for more extreme sports such as free diving and Iron-Man athletes.

However, such a health monitoring solution has never been brought to the office worker, nor the gamer. Our project seeks bridge this divide between gamers and the health monitoring industry. Our project monitors multiple relevant aspects of a gamers health. The following points detail our aims.

1. **Monitoring of the users mood via facial expressions**
2. **Monitoring of the users hydration**
3. **Monitoring of the users environment, including humidity, temperature and light exposure**

Our project aims to assist the user in their gaming/computer session, allowing them to monitor their potentially unhealthy habits. In order to do this we've created a website from which the user is able to monitor all sensory data in sessions, giving them an overview of what they behaviours they should change. This website allows the user to initiate a session, after which all sensory data will be stored, available for the user to analyze after their session has ended. This data is then stored, allowing the user to also gain a overview over how the data between sessions has changed. This data is visualized in different interactive charts, making the data more digestible for the user.

In the Architecture section we will detail how we've implemented our system using Microsoft Azure as well as Firebase. Our sensory side of the project and how we've set up our M5Stack Core 2[1] will also be detailed in this section as well as the following Implementation section. Our sensory data is run through a broker using MQTT

allowing us to receive sensory data remotely from our M5Stack Core 2.

Lets proceed with Architecture and Implementation

Gitlab (https://gitlab.au.dk/golfp2pcloud)

Youtube Demo (https://youtu.be/nIdZXpqBURQ)

Website (https://cloudproject-8112c.web.app/)

2 ARCHITECTURE & IMPLEMENTATION

2.1 OVERALL ARCHITECTURE

As a generalized overview of our architecture, the following points will detail our architecture visualization.

1. **MQTT Broker** - This broker is meant for carrying data from our IoT devices to our back-end and front-end. The broker allows us to certain topics and subsequently subscribe to these topics in order to receive our sensory data.
2. **M5Stack Core2** - This is our primary IoT device, to which we've connected all of our IoT devices. It is in charge of receiving all sensory data from the sensors and distributing it to the broker. From this device we are also able to receive data through the broker in order for it to function as a responsive device for our multi-modality purposes.
3. **Arduino(ESP32** - This is in charge of running our FSR(Force Sensitive Resistor) sensor, capable of tracking our hydration. The Arduino also publishes all FSR data to our MQTT broker.
4. **Azure Cloud** - Through docker our MQTT brokerage is done in the cloud.
5. **Firebase** - Firebase is used to host our website, as well as providing us with a cloud database, used to store all session data from the site and M5Stack Core2.

2.2 IN-DEPTH ARCHITECTURE

Our architecture works as follows.

Firstly, our we have multiple IoT devices in charge of our sensors.

1. We run a M5Stack Core2, running a ENV II(insert reference) module and publishing data from said sensor to our MQTT broker
2. We also run a Arduino(ESP32) in charge of our FSR sensor, and subsequently publishing said data to our MQTT broker

Secondly, our system's website is constructed as follows.

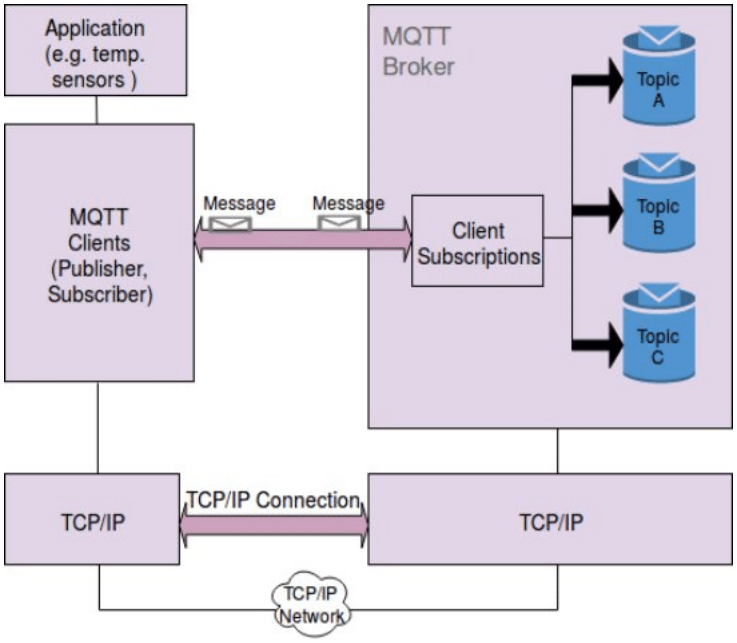
We have constructed our website using Python and ReactJS. Our website receives MQTT data by subscribing to topics corresponding to the

data we're trying to pull. This website is hosted on Firebase. Through Firebase we're also provided with cloud storage able to store all session data for the user.

Thirdly we have our MQTT broker.
We needed a method for accessing and storing our sensor data in real time, and for our purposes MQTT is optimal. Not is MQTT supported natively by our M5Stack Core2, but implementation is relatively elegant and with any unnecessary frills.

The MQTT architecture works as follows:

MQTT Architecture [3]



1. Our IoT device (M5Stack Core2 or Arduino ESP32) reads all the sensor data.
2. Through implementation of MQTT we can transmit this data in messages to a MQTT broker connected by TCP/IP.
3. These messages are able to be published to specific topics at the MQTT broker, allowing for client subscription.
4. A client with a TCP/IP connection to the broker is able to subscribe to specific topics, receiving messages published to said topic.

2.3 IMPLEMENTATION

The system consists of these core elements: sensors for data gathering, a secure MQTT broker for data receiving, website for data viewing and a database for data storing.

Sensors:
There are different sensors each with its own purpose in the system. Firstly, we have a sensor that determines the amount of water in your cup. The purpose is to make sure the user is getting the water they need and is staying hydrated. For this we are using an M5StickPlus connected via wire to an FSR. The FSR is connected to the M5 using the analog pin input. We can read a number between 1270 and 0. 1270 is max pressure and 0 is no pressure being applied at all. The software on the M5StickPlus has been made using the UIFlow[2] python-based block programming. The software works by first connecting to the internet and the MQTT broker with SSL turned on. Then we loop every 3 seconds to send data based on how much weight is being applied to the FSR. No weight is 'empty'. A little weight is 'almost empty' etc.

Then we have another M5, namely the Core2. This is also connected to a sensor using an analog pin. This sensor is the photoresistor used to determine the amount of ambient light in the environment. The purpose here being that we want to prevent the user from getting eye strain, potentially staring at a bright screen in a dimly lit environment. The software is also made using UIFlow in a similar fashion, however, the core 2 also sends humidity and temperature over MQTT that it is getting via RX and TX pins from an Arduino.

The Arduino is using a DHT11 sensor module for reading temperature and humidity and is sending via 'serial print' through the RX/TX pins to the Core2. The software is programmed in C using the Arduino IDE. Arguably the most advanced sensor is the machine learning powered camera sensor. It is a python program that connects to a webcam connected to the machine and is able to track the user's emotions. Specifically, if the user is happy, angry, or neutral. The python program uses our own AI model which has been made using Tensorflow and a lot of training data from the website Kaggle. The training data consists of images categorized after emotions. Using this data, we can train our AI to differentiate between angry, happy, or neutral. The training program is also made in python, and when it is done it outputs a model file. This file can then be used by our other python program to track emotions in real time using the webcam. This data is then being send over MQTT to our broker using the paho.mqtt package.

MQTT broker:
The broker is using NGINX running as a reverse proxy. It is responsible for getting the data from the sensors and allows other devices to listen to different topics of sensor data to get the particular data they need.

Website:
The website is hosted in the cloud using Google's Firebase solution. It has been made using Meta's ReactJS Javascript framework, which makes it easy to display a nice UI. As previously stated, the website's purpose is to display the data to the user. The website displays MQTT data in real time, but we also wanted to display graphs that might reveal certain patterns in the user's behaviour or environment. The way we decided to go about this is using so called 'sessions'. The user can start a session using a button and stop it when done. When a session is running the software begins to track the MQTT data coming in and display certain messages or suggestions to the user in the 'console' based on the data being tracked in real time. When the session is ended the software uploads and stores the data tracked in the session to the cloud database. On the website you are also able to view all the past sessions stored in the database. The data is presented as graphs for each sensor with sensor data on the y-axis and time on the x-axis. Except for the emotion graph which is displayed as a bar chart.

Cloud database:
The data is all stored in a cloud database using Google's Cloud Firestore database system. This way, the data is always available and easy to manage.

3 EVALUATION & CONCLUSION

3.1 EVALUATION

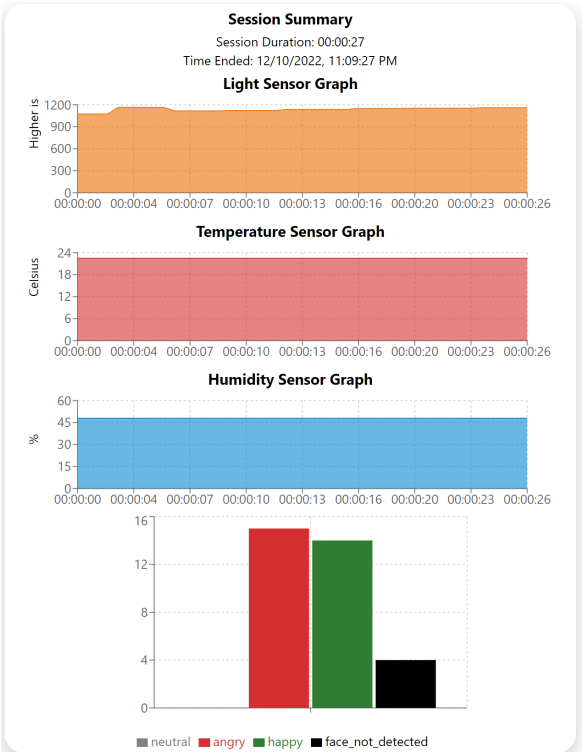
For our evaluation we ran two scenarios.

1. We evaluated the system with a developer of the system itself, capable of being critical of the systems performance, since they have had a hand in development.
2. We evaluated the system with an intended user of the system. This is in order to evaluate the merits of the system's abilities to convey the information we want them to understand, as well as gain insight into the system's design.

We will begin our findings with the system's developers results.

Our evaluation for the system's developer went as follows.

Session Data Example



The developer initiates the system, and prepares to start a session. During the session the developer checks that all processes necessary

to run the system are running. If so, the developer proceeds. Subsequently the developer starts a session, and tests the system's capabilities. As soon as the session is over, the developer checks the results of the session and checks that all necessary output was created by all input.

In our evaluation our developer tested the response time for the live data and found it to be as speedy as necessary. Genuine live tracking of the sensor data was possible. The camera AI ran as intended, with a phew hiccups. These hiccups are solely due to the fact that the camera sensor at times is slightly slow at picking up the corresponding facial expression that the user is displaying, due to a negligible inaccuracy within the AI itself.
This could potentially be due to the data-set feeding the AI not being accurate enough, or simply that while training the AI, not enough rounds of epochs were ran. Another potential problem with the AI is that, while we are feeding it less emotions in order to boost accuracy, several emotions are therefore lumped together. Initial development of the AI was ran with six emotions in total, in order to create a broader emotional spectrum. These emotions were:

1. Disgust
2. Anger
3. Fear
4. Happiness
5. Surprise
6. Neutral

During initial implementation of the AI we found that, even though we had a lot of data to feed the AI (around 1000 pictures for each emotion) we were still not able to particularly accurately predict what emotion was being expressed by the user. Disgust was often mistaken as anger, fear was often also mistaken for anger. Disgust was almost impossible to produce, and surprise would only be encountered when, almost comically, a over-expressive "O" face was displayed by the user. This was even after running very intensive training with the data-set at around twenty-thousand epochs.

We were achieving around a 60 percent accuracy, according to TensorFlow, however, this 60 percent accuracy was soon discovered to be slightly misleading, due to previous results of barely even being able to produce certain emotions. Subsequently we were forced to heavily reduce the amount of emotions detectable by the AI, and as such we achieved a much higher accuracy. The reduction did in fact

provide us with a much greater accuracy in detecting previously detected emotions, and as such we reduced the set of emotions that we trained the AI to detect. This did not necessarily "dumb-down" the AI, as we were already incapable of producing results relevant enough to merit several of the existing emotions we had trained the AI to detect.

Continuing on from this, we also saw impressive results with responsiveness of the website. Session data is updated rapidly, when concluding a session, and graphing of said data is also rapid. The UI of the website runs as intended and the developer found the UI to be fairly intuitive. This is however from the view of someone who has developed the system, and is therefore not only familiar with the structure and frame of the UI, but also someone who has imprinted upon the system itself, their own biases and preferences. UI critique will be more sensible from an outside perspective, judged by someone who has not had any connection with the project previously.

The Core2 as well as the Arduino were capable of rapidly transmitting sensor data to the broker, from which data was rapidly pulled. This provides us with a certainty of performance measures, taken during implementation, being highly successful, as previous implementations of the system were quite a bit slower.

In evaluation the feedback produced by the system acted according to expectations. The feedback from the Core2(in the form of red lights) were displayed promptly. Not only was this feedback working as intended, the console providing several prompts to the user, was also working as intended.

During testing, it became readily apparent that the cloud-based solution to storage that we had implemented, Firestore, was also working as the developer had intended. Data was rapidly being stored, and subsequently was readily available to the developer for user analysis.

Now, this is all from the perspective of a developer of the system. The plight of the creator is often to be incapable of substantial self-criticism and therefore an outside perspective is needed. When looking for a evaluation participant it was crucial to find a participant from the intended user-base. We therefore concluded that it was necessary to find a "gamer" or otherwise computer enthusiast, who was no stranger to potentially frustrating or reversely exciting experiences during a session at their PC.
We found a willing participant, who like our intended user-base, played a lot of video-games at their computer. The video-games this participant plays are usually quite competitive, and therefore more likely to provoke an emotional response from the user. This com-

bined with the fact that competitive video-games have a tendency to provoke a more brisk change of pace, as far as emotional reponse goes, made the participant of high quality to us.
Now let's proceed with the results from the evaluation of the intended user.

The user's initial impressions were occupied by the UI. The user found the UI to be fairly intuitive, altough the user found navigating session data tedious. The long necessary to read previous session data was for the user, unituitive as the user wished for a more concentrated experience of the user data. The user suggested sectioning off certain parts of the website, navigable via the UI. More spefically, the user suggested a top-bar, capable of traversing several parts of the website, and upon further development(adding session analysis), this could prove in providing greater clarity for the intended user. Although the UI left some things to be wanted from the user, they were clearly able to navigate all critical aspects of the system. They were able to independently start and stop a session, and able to find their own session results.

The user started a session, and booted up their favorite video-game. Said video-game was a competitive, first-person shooter game, and as such fairly intense. The user spent around five minutes in-game testing the system. During this time the user displayed all key emotions, namely, anger, happiness and neutral. The user kept track of what the system was telling them during their session, and found some hilarity in the suggestions made by the system. The system will promptly tell you to calm down if you display anger, and as such the user's emotional response quickly changed to "Happy". The user left the room and came back into the room, only to read that the system had asked where the user was. This was also quite humorous to the user. The had several attempts to manipulate the results of the session. This included several things.

1. The user tried to manipulate light levels in the environment to manipulate the light sensor, prompting it to suggest a brighter environment for the user to proceed in.

2. The user manipulated the humidity and temperature of the environment by breathing onto the sensor itself, resulting in output from the system telling them that the environment had changed and that action should be taken to improve said environment.

3. The user grimaced at the camera in order to test the accuracy of the camera AI, and found that the camera AI produced quite an impressively accurate result.

4. Finally the user manipulated the hydration sensor by lifting up the cup on the sensor, witnessing the real-time water response on the website.

On to the user's session results. The user found some data useful but some data quite trivial. The hydration sensor response was especially jarring to the user, as they did not see any value in a result being either:

1. Water
2. No water

The user did however like some of the capabilities of the system. Namely, the user found real value in the emotional tracking of the system. They enjoyed manipulating it at first, however, they found that genuinely seeing how their emotional response had been during the course of their session. The accuracy of the camera AI, discovered during manipulation, led to a sense of trust for the user in the result of the session. The user had been almost equally displaying anger and happiness during their session, and were very surprised at the results. They were under the presupposition that they would almost exclusively show either anger or neutral, but they were not.

The user found that even well after their manipulation of the environment they were experiencing a change in environment quality. The user's environment had become slightly more humid and slight warmer. The user found this data to be interesting as they had not previously thought of the game that they were playing to be physically strenuous. This caused them to be able to look at their actions in a new light, and found that they had gained a new understanding of how potentially stressful something as sedentary as gaming could potentially be.

Beyond manipulating light levels initially the user did not see any change in their environment, as far as light exposure goes. The user enjoys playing in lit environments and the system therefore does not provide any insight for the user regarding the potential dangers of low-light environments while using a computer.

The user was willing to give additional feedback to us as a part of their evaluation. Namely what they thought of the system, and whether or not they could see themselves taking advantage of a system like ours.

The user first and foremost took issue with the UI. Although fairly intuitive, they could not see themselves using such a system without a severe overhaul of the UI. The session data did not work very well

for the user, since they were effectively almost incapable of searching through the session data, should there be more than a few entries. Additionally, they found that the lack of session data analysis made the system more of a novelty than a product they would employ daily. While fun for a little while, without more insightful data analysis it was of no great consequence to them. The greatest asset of the system, according to the user, was the emotional response tracking. The ability to fairly accurately track emotional expressions was to the user a great feature. They were impressed by the accuracy and wished for more emotions to be added. The user went on the explain that if more emotions, such as fear, surprise, disgust etc. were added, they would be much more interested in employing a system like ours in their daily routine. They were very interested in the possibility of using the system to track emotional response during the most stressful of times, not even necessarily things such as horror games, but horror movies as well. This, to the user, would increase the likelihood of them using a system like ours greatly, as they could find entertainment value in sharing their experience with friends. They imagined themselves using the system to try and measure, for example, who was the most scared out of a group of friends watching the same thing. If the camera AI would also be able to track more than one face at a time, this feature could be employed for groups of people, not just individuals.

Wrapping up, the user expressed what would have to be implemented for them to be interested in using the system in a individual manner. Hydration tracking would have to be improved upon. This was one of the things the user was most interested in using, as proper hydration was a problem for the user currently while playing video-games. They would often be dehydrated after a long session, and saw great value in any attempt to better the unhealthy habit they had. Subsequently they were also interested, again, in much more extensive data analysis. If they were able to track their improvement over time in several categories, they could be able to prove a either upwards or downwards trend in their habits. This would enable the user to effectively change unhealthy habits or become aware of unhealthy habits forming along the way.

3.2 CONCLUSION

In conclusion we have demonstrated the capability for our system to accurately track several behaviours of a user. We are capable of tracking hydration, user presence, user facial expression, as well as user environment. Despite this, the project's initial goals, predating this report, were not all met. The implementation of certain features, such as accurate hydration tracking, were not able to be implemented

due to a time constraint, and as such was overlooked for more critical features of the system. The system's camera AI, although not revolutionarily accurate, is still capable of accurate facial expression tracking, and as such provides real value for the system. Sensor data is as well accurate, and speedily available to the user in real time, along with several prompts for the user to change their behaviour, or make them aware of said behaviour. Although initially the system was also meant to track the user's excitement or anger level through the microphone of our Core2, this implementation was proved impossible with the M5Stack UIFlow's limited support for the built-in microphone.

More insightful analysis of session data was also under prioritized for more crucial feature implementations, however, currently session data analysis is still possible, but at the hands of the users themselves.

BIBLIOGRAPHY

[1] M5Stack. *M5Stack Core2*. URL: https://shop.m5stack.com/products/m5stack-core2-esp32-iot-development-kit.

[2] M5Stack. *UIFlow IDE*. URL: https://m5stack.com/uiflow.

[3] Dipa Soni. *MQTT Architecture Reference*. URL: https://www.researchgate.net/figure/MQTT-Architecture-2-Broker-Broker-controls-the-distribution-of-information-and-mainly_fig1_316018571.