SNOOZELES Alarm Clock

Final Project – Introduction to IoT – CS3237

Members: Linoy Erez, Eden Daniel, Noa Hausman, Koren Segal Benedek

Introduction

Morning alarms are often unwelcome, as they imply that it's time to get out of the cozy bed and start the day. There are two types of people: those who hear the alarm and immediately wake up, and those who regularly sleep through the strong and annoying noise. Some researchers suggest that deep sleepers have more sleep spindles, a form of brain activity during sleep, which "act as a noise-canceling device", making it difficult for them to wake up from the alarm sound, and sometimes even turn it off while sleeping[1]. This phenomenon is also common among people with sleep disorders, like insomnia or delayed sleep phase disorder (DSPD).

To help those people, we developd a smart alarm clock that rings continuously until the person is fully awake. It ensurs that the person has completely gotten out of bed. Our solution is composed of a physical alarm device, a web page for setting the time, two IMU sensors - connected to one belt and one wrist band - for collecting the data, and a data analysis page for the user to keep track of his progress.

System Architecture

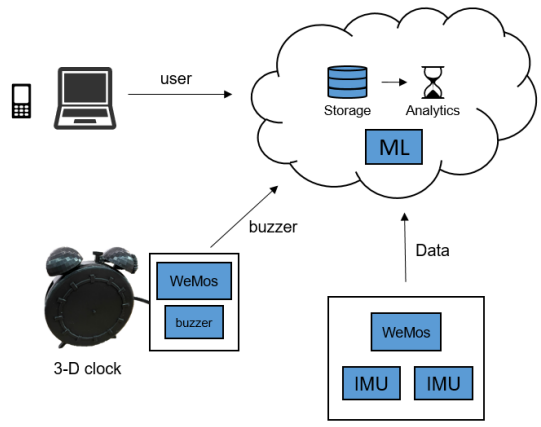

Fig 1. System Architecture

There are 4 main components in our system's infrastructure: three edge device and a main server. These devices communicate with the main server, using HTTP protocol.

1. Main Server:
   a. A Flask app which runs on the cloud.
   b. Manages the entire system, from front-end to back-end.
   c. Has a fixed IP, enables reliability and full availability.
   d. Activates the user interface.
   e. Manages the database that saves relevant multi-user information.
   f. Generates long-term analytics per user.
   g. Operates the ML model for decision making on real-time processing.
2. IMU Wemos:
   a. Uses I2C protocol to get data from two different IMU sensors simultaneously. One of them is connected to the person's chest and the other is connected his wrist.
   b. Uses mobile battery as an energy source. It is only in use during the nights so it can be charged during the day. In fact, one day of charging is enough for a few nights in a row. Therefore, power consumption was not a significant consideration in this implementation.
3. Buzzer Wemos:
   a. Connected to one buzzer sensor.
   b. Uses polling method over HTTP to update its state and act accordingly (turning the buzzer on and off).
   c. Stays in a 3-D printed clock.
4. Client Device:
   a. Any device that has access to a web browser - computer, mobile phone or even a smartwatch, thanks to the fact

[1] https://www.healthline.com/health/healthy-sleep/sleeping-through-alarm

that the user interface is implemented as a responsive website.

b. Enables the user to use the services offered to him (will be discussed later on 'application design').

Application Design

We wanted the users to have an interface in which they can manage their activity conveniently from any device. We implemented the idea with a "Flask" app, which gives us the ability to manage information between the front-end and the back-end.

We created a friendly web app with four pages as following: the first page is a landing page where a user inserts his details. After the details were submitted, the server pulls the user's details from a POST request and adds it to the database. The database stores unique values that are identified with the user and with his edge devices. Moreover, we had to take care of transferring those details between the webpages and this was handled by POST requests as well.

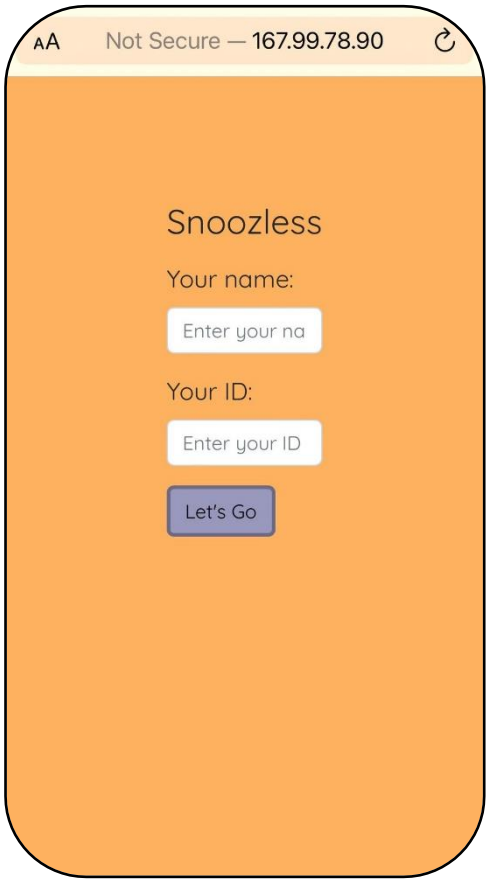

Fig 2. First Page

The second page displays the current time and allows the user to set/edit an alarm. Any action that is made by the user triggers a POST request that is sent to the server. For each request, we compare the new information to the older information held in our system and update it accordingly. Meanwhile, the alarm records in the database are constantly checked by a function that runs in the background (on a thread of its own) and decides if any of them matches the current time. This means that we are tracking the current time twice, firstly for the display in the frontend side and secondly for the alarm records in the backend side.

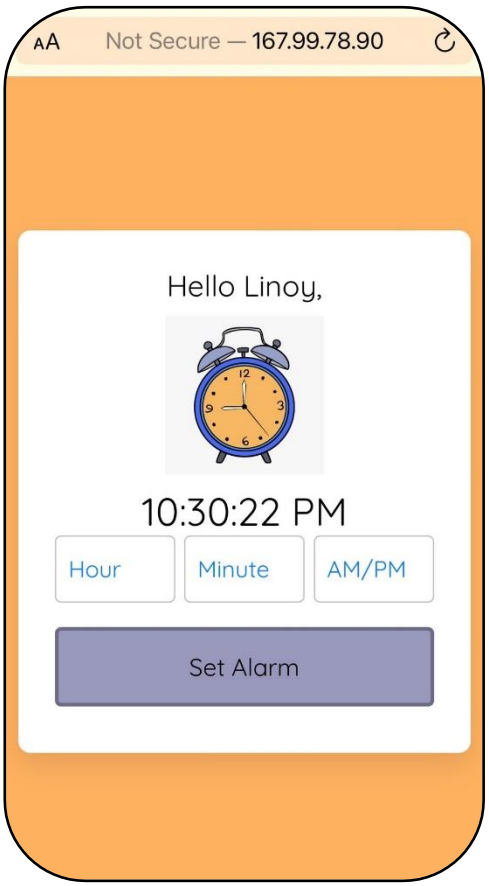

Fig 3. Second Page

The third page is uploaded only when the time for the user to wake up arrives. At this point, ML model begins to predict the current state of the user. In order to allow multiple users, we create a different thread for each prediction process. This is the point when all components are working together; buzeer rings, we accept the data coming from the IMUs (from the wemos that matches to

the person) and make predictions on real time. During this stage, we measure the total time it took the person to be fully awake, that happens when the ML predicts so. We save information regarding the above to the database for the use of future analytics.

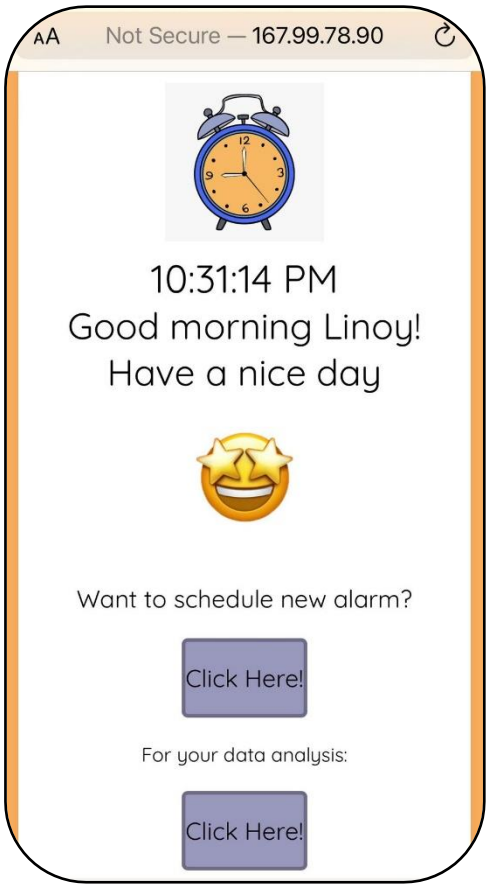

Fig 4. Third Page

The fourth and last page is the analytics that are made for the user based on his past performances on the app. Each time the user asks for his analytics, relevant data is being pulled from the database, so the analytics that are displayed will be up to date.

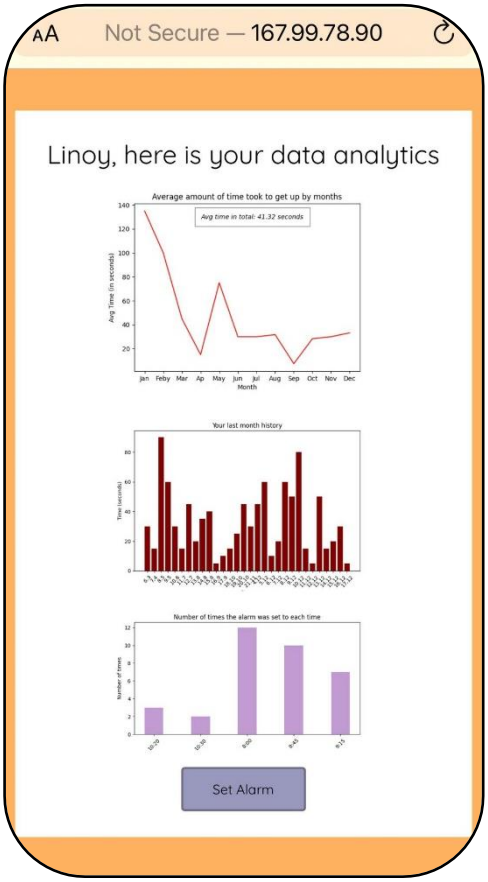

Fig 5. Fourth Page

Cloud

A good system is a one that is reliable, stable, and accessible anywhere, anytime. We want our system to be based on an infrastructure that can support multiple users as well as handle parallel actions from different users. Therefore, we use a cloud server in our system. We chose Digital Ocean's services for that purpose. One advantage of using a cloud is that it has a permanent IP. Therefore, any edge device can establish a connection with the server at any time, independently of its location. Moreover, the computing power and memory are stable and more flexible. To support multi-users effectively we had to create multiple processes with access to shared variables simultaneously. In practice, to manage global information, we are maintaining a SQL database using Sqlalchemy. Sqlalchemy handles collisions properly, hence in case of two parallel requests to the database, the information won't be lost or damaged.

Alarm	Predicting Status	Buzzer Status
User ID	User ID	User ID
Time	Predicting state	Buzzer state

Fig 6. The format of our database tables

When the alarm is set by the user, using the user's unique id, an 'Alarm' record is created, and 'Predicting Status' is initialized to false. When the alarm time arrives, the main server:

1. Updates the 'Buzzer Status', so the WeMos could activate the buzzer sensor.
2. Changes the 'Predicting Status' to true, in order to accept data sent by the IMU WeMos (until this point we ignored it) and start the prediction process.

After this process ends, the relevant records are deleted from the database.

ML

Machine learning can help simplifying hidden patterns in IoT data by analyzing massive amount of data and making decisions based on the patterns it is recognizing[2]. For our product, we created a Convolutional Neural Network (CNN) model. CNN is a type of deep learning model designed to adaptively learn spatial hierarchies of features through backpropagation. It is typically composed of three types of layers: convolutional, pooling and fully connected layers[3]. CNN is widely used for processing data with grid pattern, such as images. The data we collected is spatial and represents linear acceleration (accelerometer) and angular velocity (gyroscope) along three axes.

We have used a binary classification with the following two classes: '*awake*' and '*not awake*' (will be discussed later, on 'challenges and limitations').

Collecting the data: each sample is composed of 12 features, the first 6 are coming from the IMU sensor connected to the chest belt, and the other 6 are coming from the second IMU sensor connected to the person's wrist. Each IMU sends the linear acceleration measured along three axes (x, y, z) and the angular velocity along the same axes, respectively. We created our own dataset by collecting the data and writing the samples to a file. We recorded approximately 1000 samples for each class in order to make the dataset rich and balanced. We then trained the CNN model on the samples file.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Acc_x_1	Acc_y_1	Acc_z_1	Gyr_x_1	Gyr_y_1	Gyr_z_1	Acc_x_2	Acc_y_2	Acc_z_2	Gyr_x_2	Gyr_y_2	Gyr_z_2	Class
2	-0.31	-0.63	1.28	-6.11	-2.94	0.64	-0.1	1.29	1.89	-1.67	2.63	0.12	0
3	-0.31	-0.63	1.28	-6.12	-2.89	0.68	-0.1	1.29	1.89	-1.55	3.4	0.32	0
4	-0.31	-0.63	1.28	-6.21	-2.89	0.71	-0.1	1.29	1.9	-1.5	2.72	0.32	0
5	-0.31	-0.63	1.28	-6.09	-3	0.66	-0.1	1.29	1.9	-1.39	2.41	0.27	0
6	-0.31	-0.63	1.28	-6.07	-2.94	0.68	-0.1	1.29	1.9	-0.99	3.09	0.35	0
7	-0.31	-0.63	1.28	-6.1	-2.96	0.68	-0.1	1.29	1.89	-0.92	3.74	0.43	0
8	-0.31	-0.63	1.28	-6.27	-2.93	0.71	-0.11	1.29	1.89	-0.96	3.24	0.61	0
9	-0.31	-0.63	1.28	-6.09	-2.96	0.71	-0.1	1.3	1.89	-0.97	2.69	0.4	0
10	-0.31	-0.63	1.28	-6.08	-2.99	0.67	-0.11	1.3	1.89	-1.24	2.98	0.28	0

Fig 7. An example of our samples file

Prediction process: the IMU data is sent in a rate of 100 milliseconds to the server, but it's being written to a file only when the time set by the user matches to the current time. At the same time, in a continues loop, the model predicts on the last 10 samples that were added to the file, and makes its decision based on a threshold we defined in advance. If 80% of the samples (or more) were predicted as 'awake', the model decides the person is fully awake and stops the alarm. Otherwise, it will continue to another iteration of the loop. When the prediction process is over, some information is sent to the main server for future analysis use. Currently, the accuracy of the model is ~90%.

2 https://www.softwareag.com/en_corporate/resources/what-is/machine-learning.html

3 https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

Long-Term Analytics

Our system collects a huge amount of data, this data is aggregated for generating long term analysis of the user's activity. These analytics are later presented in a friendly and interactive page on the web-app, in order to teach the user of his personal behavior. After a prediction is being made, the system pulls the user's personal data from the cloud, using his unique id. It then adds the new relevant information - the current date, the alarm time the user had set, and the amount of time it took him to wake up (in seconds). Then, based on that personal data, which is always up to date, three graphs are created, and are shown as images on the screen, right after the alarm goes off.

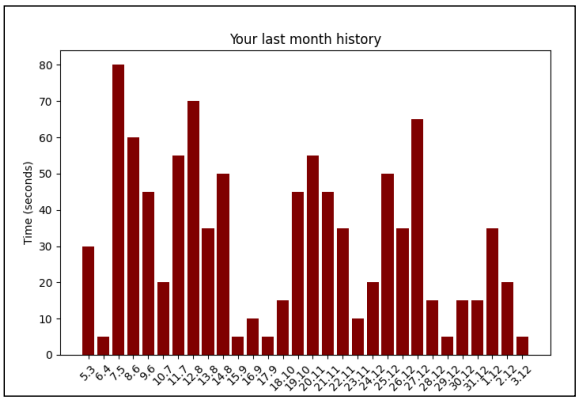

Fig 8. ***Last 30 Days History*** *- the user can observe how long it took him to wake up each day, on the last 30 days*

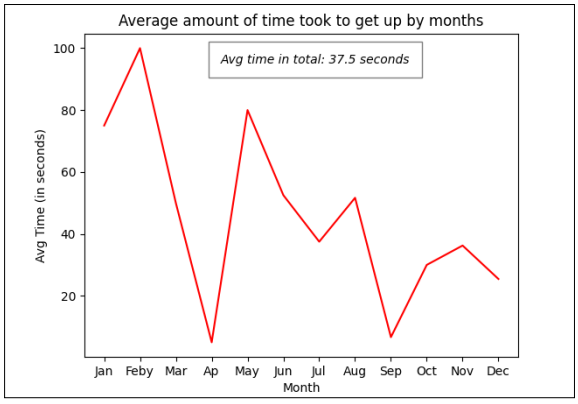

Fig 9. ***Monthly Average Time*** *- the user can observe how long it took him to wake up each month on average*

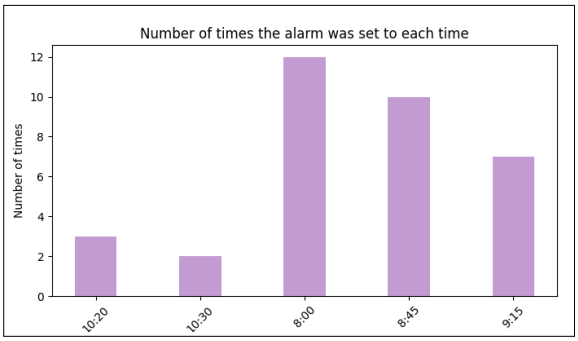

Fig 10. ***Most Used Alarms*** *- the user can observe his preferable hours to wake up*

Challenges and Limitations

Although our solution answers the main problem, we had some difficulties along the way in finding the best practice to deal with the challenges that faced us. Described below are the main challenges and limitations:

1. ***Multiple Users*** – one challenge we were facing while writing our code, was how we were going to handle multiple users. We tried thinking of the best solution for holding the unique information of each user and pull it when needed, such as the ID of the user and the time he set for the alarm. Our solution was to maintain a SQL database. The DB holds in different records the unique information of each user.
2. ***Shared Information*** – in the Flask app, each path runs independently and thus can't recognize any global variables. We had to think of a creative solution that provides an alternative for sharing information between the paths. We realized that the best solution will be using a database with fast access in order to have constant communication between those paths.
3. ***Comfortable Experience*** – our product attaches to the user's body with one chest belt and one wrist band. We think there are some improvements to be made in

order to make a comfortable product for a daily use.

4. ***Generalized Data*** – as we collected our own data for the ML model and did not use any existing data, our data is more personalized to the group members and not generalized enough. In the future, changes will have to be made in collecting more data on various people from different ages and with different physical abilities.
5. ***Machine Learning Model*** – we had to deal with some very sensitive sensors that affected the reliability of our model. For example, at first, we tried recognizing the 'waking up' process using several classes. We realized that the IMU is very sensitive to movements and therefore the data included noises. Moreover, it was sending different data, scaling-wise, every time we collected the data and thus it forced us to re-create the dataset many times. After trying various versions of the structure of our model, we decided to focus on a binary classification. This way, the ML model was able to differentiate between the classes more easily.

Future Directions

1. ***Sleep Cycles*** – the system collects data while the user is sleeping. We want to use this data to detect sleep cycles and improve the ML model by taking into consideration the person's sleep state.
2. ***Personal Recommendations*** – based on the personal long-term analysis collected, we wish to advise the user some personal recommendations.
3. ***Voice detection*** **-** using the sound sensor, we can analyze relevant sounds during a person's sleep to achieve a more accurate ML model.
4. ***User calendar synchronization*** – we wish to synchronize with the user's calendar so our system could alert the user in case the alarm time doesn't match his schedule.

Responsibilities

All group members were involved in all various fields of the project. We discussed on each topic, suggested ideas, and came up with a plan which consisted detailed tasks list. Tasks were equally shared and combined together to an integrated system. In general, Koren was in charge of the cloud services and storage, Linoy was in charge of the long term analytics and the main Flask app, Noa was in charge of the frontend side of the system including web pages and data management and Eden was in charge of the WeMos devices, communication protocols and the ML process.