Faculteit Ingenieurswetenschappen

KU LEUVEN

PROBLEEMOPLOSSEN EN ONTWERPEN, DEEL 3

Team CWA3

Lies Bollens
Ruben De Clercq
Jakob Festraets
Rugen Heidbuchel
Floris Kint
Peter Lacko

Quantified bike

PROGRESS REPORT

Supervisor
Prof. dr. ir. Erik Duval

Assistants
Sven Charleer
Jose Luis Santos
Robin de Croon
Joris Klerkx

ACADEMIC YEAR 2014-2015

Contents

List of Figures

1 Group members

The group consists of the following members.

1. Lies BOLLENS, Bachelor of Science in de ingenieurswetenschappen, 2nd year.
2. Ruben DE CLERCQ, Bachelor of Science in de ingenieurswetenschappen, 2nd year.
3. Jakob FESTRAETS, Bachelor of Science in de ingenieurswetenschappen, 2nd year.
4. Rugen HEIDBUCHEL, Bachelor of Science in de ingenieurswetenschappen, 2nd year.
5. Floris KINT, Bachelor of Science in de ingenieurswetenschappen, 2nd year.
6. Peter LACKO, Bachelor of Science in de ingenieurswetenschappen, 2nd year.

2 Brainstorm

The pictures of our Brainstorm session can be found in Attachment 10.1. A list of all the ideas created during the brainstorm.

1. Using a smartphone.
   (a) Get real-time feedback.
   (b) Play music/ put on a light.
   (c) Take pictures of yourself/ your surroundings.
   (d) Easily find where your bike is parked.
   (e) View in an instant where your friends are.
2. Using a GPS.
   (a) Get the fastest route from one point to another.
   (b) Display a card, showing all the places you have been to with your bike.
   (c) Make an altitude map.
   (d) Easily track the distances you've covered.
   (e) Display some points of interest (ex. All the pizza restaurants nearby).
3. Try to make your trip as efficient as possible.
   (a) Avoid red lights, based on your previous bike trips.
   (b) Get a map, displaying the quality of the roads.
   (c) Find out during which moment of the day you cover a certain route the fastest.
   (d) Get the average time needed to brake.

(e) View your speed, based on the altitude difference, the temperature, the hour,...

(f) Avoid bad roads.

4. Monitor your health.

   (a) How many calories have you lost?

   (b) How much water do you need?

The initial idea was to use a smartphone, since these devices contain almost all the sensors needed for the project. This way, creating an app for Android and iPhone devices would be enough for the concept to be functional. The assignment, however, was to incorporate an Arduino Nano and a Raspberry Pi. Since using a smartphone, an Arduino and a Raspberry Pi would get a bit too complicated, our team decided against using a smartphone. This meant that there would not be any screen attached to the BOSS device anymore. Real-time communication with the user was therefore no longer possible and functions such as find-your-friends and find-your-bike not doable anymore.

A lot of the ideas created during the brainstorm, focused on making an augmented bike. This is not what the BOSS device was aiming for: the purpose of this project is to create a quantified bike device, without looking at the augmented side of a bike trip. Once this goal had been set, a lot of the brainstorm ideas could be thrown overboard: get the fastest route, view your calorie deficit, see how much water you need, locate your friends, get points of interest,...

The idea of trying to avoid red lights seemed very appealing at first, but turned out to be too difficult to implement. It would require an accurate route planner that constantly refreshed, while taking all the red lights in the area into account. A second reason against this idea was the fact that, as a lot of the other ideas, it is much more on the augmented than on the quantified side of a bike trip.

3 Product

3.1 User stories

3.1.1 Comparing daily trips

Sophia uses her bike and the BOSS on a daily basis. She bikes to class every day and clips the BOSS on and off her bike so it does not get stolen. On the BOSS website, she can easily review each trip she made in the calendar overview. She can get a general impression of how she did each day, and then she can compare her stats with a ranked list of the stats of her friends. This includes the farthest traveled distance, the longest bike trips (in time), the fastest speeds, etc.

3.1.2 Sharing biking experiences with friends

Luke likes to go on bike trips for fun and exercise. He sometimes doesn't know where he will go, but he can easily find out where he's been in the BOSS's detail page on the website. When he sees a beautiful view, or just something interesting, he can easily take a picture on the built-in camera with a click of a button. He can also press a second button to mark certain points of interest.

The points of interest are displayed with the photos on a map of his route that he can share with his friends on various social media sites (Facebook, Twitter, Google Plus,...) On top of the map, he can share various statistics of his trip (how bumpy the road was, how fast he went, how hot and humid it was) along with personal comments and ratings. If he wants, he can also view the statistics and comments of fellow cycling friends.

3.1.3 Fitness statistics

Johnny likes biking for sport, and loves using the BOSS to record his data. He always keeps the BOSS on his bike, and only has to flick a switch to turn it on or off. After each ride, he goes to the BOSS website to check specific details of how he did, and can compare his performance with previous rides, made possible by the detailed compare view of the BOSS website. Furthermore, he can also compare his data with his competitive friends, by use of simple numbers (for the mean speed, travel distance and travel time), or with graphs (speed during a trip, measured temperature throughout the trip, heart rate, etc.) Finally, he can challenge his friends to beat his results via various social media sites.

3.2 Product description

The BOSS records several different types of data with external sensors that are either connected to the Raspberry Pi or the Arduino Nano. If an internet server is available, the data is sent to the server. If the connection is lost, the data can be stored locally. It also has three input buttons: there is a switch to turn it on or off, a button to take a picture with a camera and a button to indicate a point of interest. The data from the last two are combined with recorded GPS data. Exterior LEDs indicate whether the connection to the server is working or not. The website starts with the user's own dashboard, showing random statistics of them and their friends. Here they can either choose to go to their personal (calendar) view, or compare and share with their friends in the social view. The calendar view gives the user an overview of their rides on each day, and a more detailed view per day after the user requests it. They can also compare multiple days by selecting the chosen days in the calendar view. The social view allows the user to compare his statistics with a friend, or find multiple rankings based on different statistics (e.g. highest speed, most distance traveled, longest bike ride). The user can also share his results and challenge friends on multiple social media sites.

4 Architecture

Our application mainly consists of 3 parts. A Raspberry Pi and an Arduino Nano are mounted on the bike to collect the data. The other one is the web interface.

4.1 Device

4.1.1 Arduino Nano

The Arduino Nano is connected to a GPS Sensor, a temperature sensor, a humidity sensor, a heart beat sensor and a few buttons and LEDs. The code on the arduino interprets the data coming from the sensors and sends it via a serial interface to the Raspberry, which is connected via a USB Cable. The interval between consecutive data transmissions depends on the kind of sensor. For instance, the state of a push button is sent more frequently than the GPS data. All data is sent in a specific format. One packet of sensor data consists of a single line. A semicolon is used as delimiter to separate values. The first value of a data packet identifies the sensor. The format of the next values depends on the kind of sensor.

4.1.2 Raspberry Pi

On the Raspberry Pi, a Python application reads data from the Arduino. When an Internet connection is available, the application can send the data to the server in realtime. Alternatively, all trip data can be stored in a local MySQL database. As soon as an Internet connection is available, that data can be sent to the server by a separate script. When that data transmission succeeds, the data is removed from the local database.

4.2 Web interface

For the web interface we use one central javascript class. This class manages all data queries from the database. It also takes care of all necessary initialisations for the main webpage. This includes the calendar initialisation and the basic actions initialisation (button actions etc.).

When data gets queried for the current month, the data is temporarily stored. The data model uses an array of day objects. These objects all have two properties.

1. ***trips*** *(Array):* contains the trips for that day.
2. ***average*** *(Object):* contains the averages for that day. Examples are average speed and total distance.

A properties array is used to identify the different average properties. This array is used for the calendar bar property and the details page. For the details page this array also holds the postfix and precision of every property. For total distance, the postfix would be ' km' for example.

5 Used technologies

5.1 Device

5.1.1 MySQL

When the application can't send data to the remote server, all data is stored on a local MySQL database. MySQL is an open source relational database engine.

5.1.2 Raspberry Pi

The Raspberry Pi is a credit card-sized single-board computer developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools.[1] Because of its low cost, the Raspberry Pi is perfectly suited to be used as computing device for the Quantified Bike. It's not a high-end device, but it is sufficiently fast to process the data for the Quantified Bike.

The application uses the following additional libraries:

1. *python-serial*: The Arduino sends sensor data to the Raspberry Pi via serial communication.
2. *SocketIO*: Websockets are used to communicate with the central database.
3. *threading*: The application on the Raspberry Pi uses threads so the application can simultaneously read data from the Serial Interface which is sent by the Arduino and send data to the local database or the server.

5.1.3 Arduino Nano

The Arduino Nano is a single-board microcontroller, which can be used to simplify building an interactive environment. It gives us the advantage that it becomes rather easy to connect and read sensors and the like, which else would have been a difficult job. A disadvantage to the Arduino however, is the fact that it can only run one thread at a time. This makes reading multiple sensors slightly harder. For example, the GPS module has to wait for a signal, while the temperature sensor gives us a constant signal. Luckily, it still is possible to program the Arduino so that it can read all sensors on the right time and process all the data to the Raspberry Pi. To make it possible for the Arduino to read the sensors, we use some custom-made libraries.

1. DHT11 for the temperature-humiditysensor
2. Adafruit_GPS for the GPS module

5.2 Web interface

5.2.1 Javascript

Javascript is a programing language used to program the behavior of a web page. It is easy to implement in a HTML file. An advantage of Javascript is that it is executed on the client side. This way, the web server won't unnecessarily be strained. A disadvantage is that a lot of lines have to be written in order to get some basic functionality. In this project, Javascript is used to create all functionality on the client side. Without Javascript, the client side would not be able to get any data from the server and would thus be useless.

5.2.2 JQuery

Jquery is a JavaScript library, created to simplify JavaScript programming. Using JQuery, lots of commands are a lot easier and faster to use. In every JavaScript file used in this project, JQuery commands are present.

5.2.3 DataTables

For the calendar, an external Jquery library 'DataTables' was used. This library makes working with HTML tables a whole lot easier. It provides asynchronous initialisation, pagination, sorting, etc.

5.2.4 ScrollTo

When a calendar day is clicked, detailed information for that day is presented in a details section beneath the calendar. To automatically scroll down to this view, we used the external jQuery ScrollTo library. This library still presents a few bugs, so we are still looking for an alternative.

5.2.5 Chart

Chart is an external jQuery library for making charts. We have not used this library yet, but it seemed easy to use and supports nice animations.

5.2.6 JSON

JSON is a data interchange format. Although it stands for JavaScript Object Notation, it is language independent. JSON is relatively easy to understand and is faster than other similar formats. In this project, JSON is used for data interchange between the client side and the databank.

6 Course Integration

7 Conclusion

8 Appendix: Workload

9 Appendix: Planning

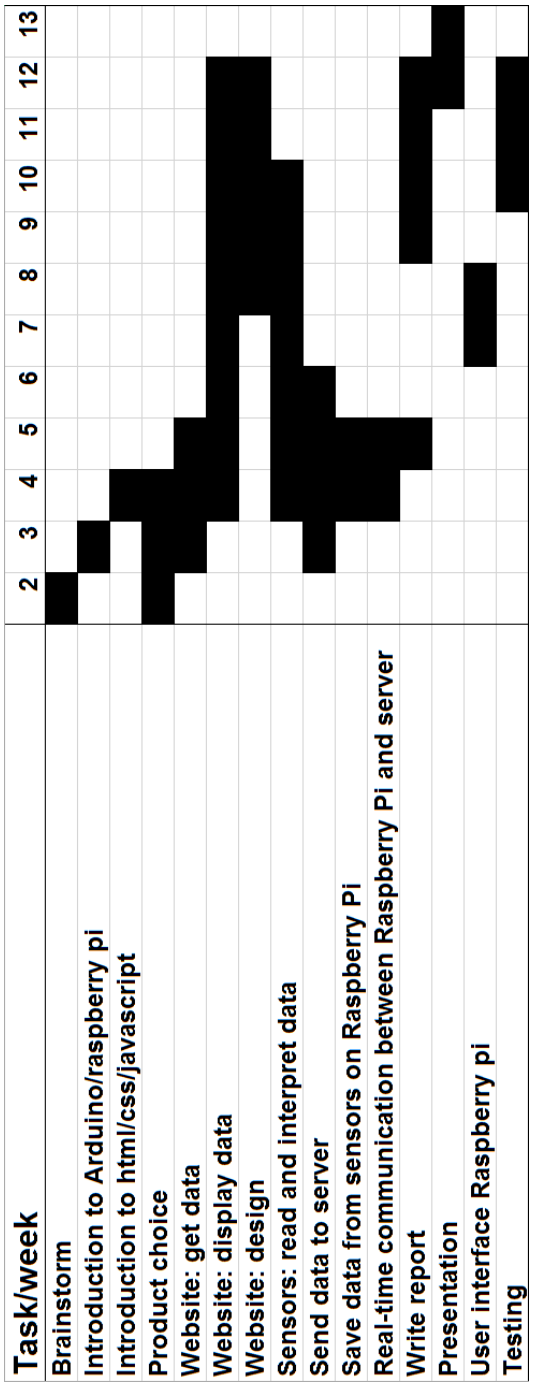

Figure 1: Gantt Chart

10 Attachments

10.1 Brainstorm pictures

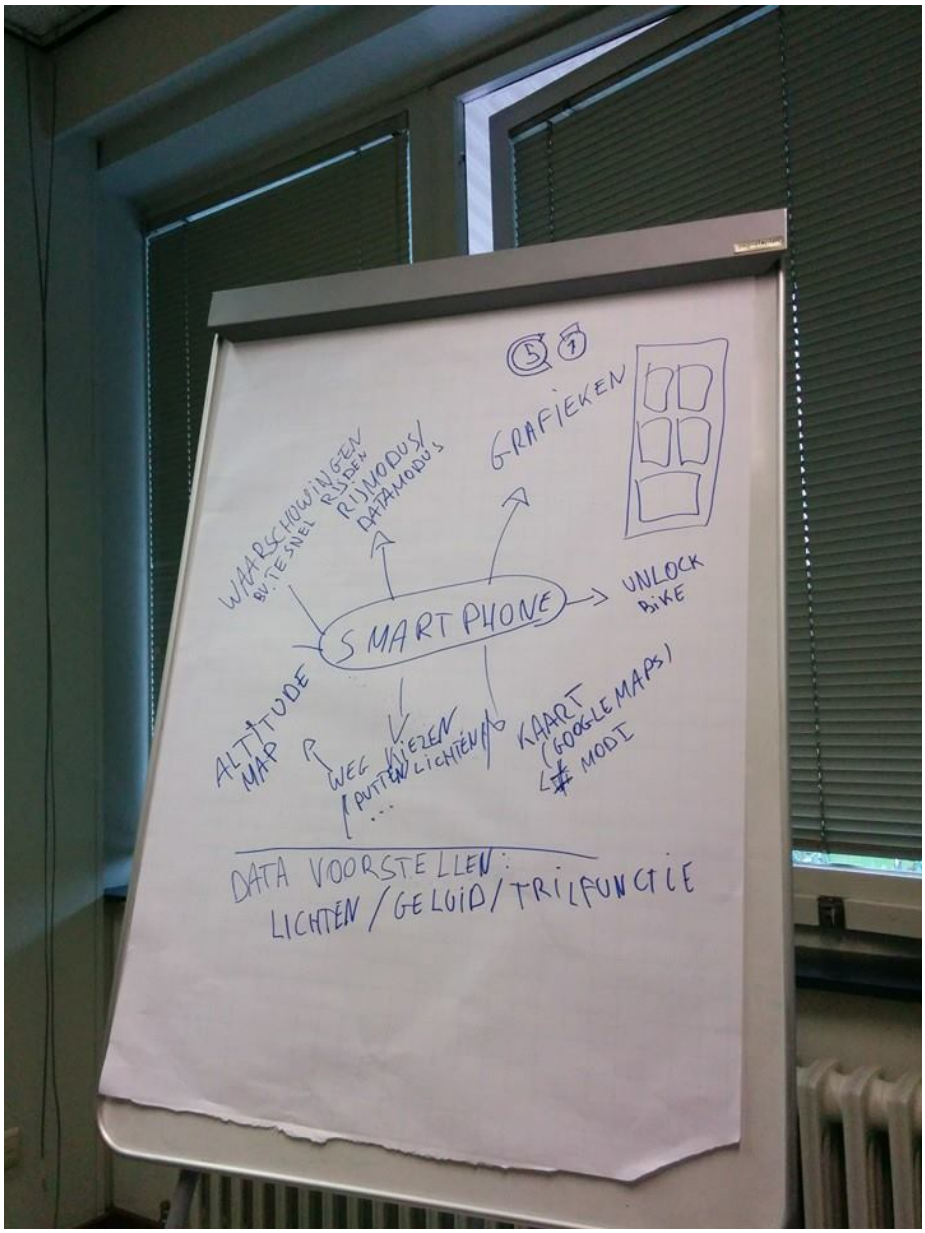

Figure 2: First brainstorm

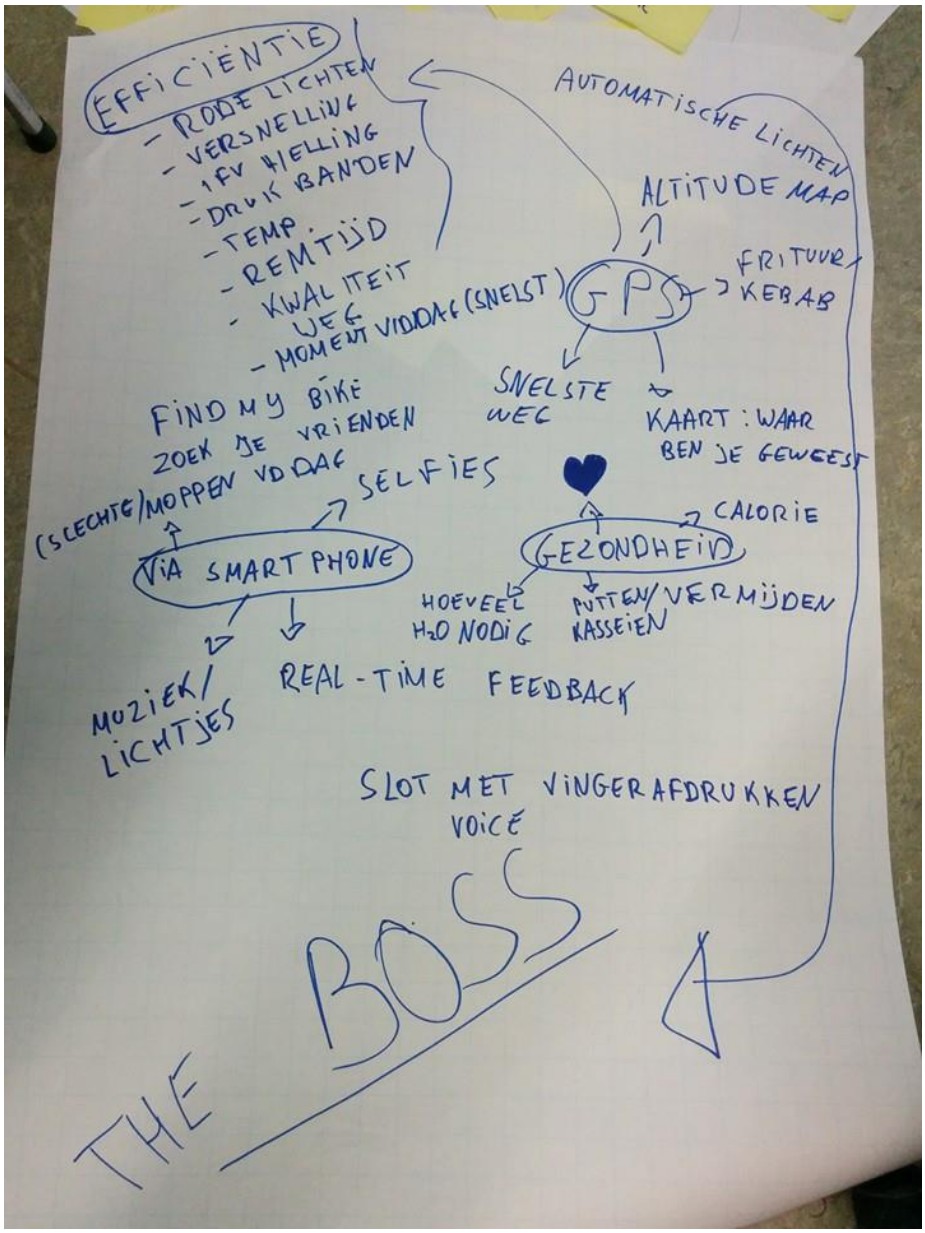

Figure 3: Second brainstorm

11 References

[1] Wikipedia. Raspbery Pi. Consultation: 2014-10-23. [Online]. Available: http://en.wikipedia.org/wiki/Raspberry_Pi