HELMO: A Smartphone App for a Smart Helmet

Mihnea Rusu (mr3313@ic.ac.uk)
Imperial College London

Abstract: With the rise of cycling in Great Britain [1], become likelier for bikers. Helmo proposes a smart helmet solution coupled to a smartphone app that improves the diagnosis of fall related injuries. It does so through algorithmic classification of falls by severity and 3D visualizations of falls, available to cyclists and their physicians.

I. Introduction

Helmo, is a smart helmet that records and observes the safety of urban cyclists. According to the latest British Social Attitudes Survey [2], 64% of the people interviewed agreed that cycling on the road in the UK is too dangerous. Furthermore, in 2015 the Department for Transport calculated that there were 18,845 pedal cyclist casualties on the road [3] in Great Britain. Despite the fear-inducing statistics, cycling has increased in popularity as a method of commuting to work amongst the British population, with 741,000 people working residents doing so in 2011, up by 90,000 over a period of 10 years [1]. Thus, there exists a scope for creating a product that assists cyclists and their healthcare providers in the case of an injury.

Helmo tracks the user's head pose in real-time using an inertial measurement unit (IMU). Impacts experienced by the cyclist upon falling are recorded using resistive pressure sensors. This is accomplished through a low-power, Bluetooth Low Energy (BLE) enabled device embedded inside a regular cycling helmet that stores motion information over a rolling temporal window. The helmet is connected to an app on the user's smartphone – whenever a fall is detected, the app receives the sensor data of 10 seconds preceding the fall, through to the end of the fall. A history of the cyclist's journeys and falls/impacts is stored in a database on the cloud and exposed inside the app in a user-friendly way.

3D animations of the falls and impacts experienced by the helmet are rendered inside the app. Falls often happen, as one may anecdotally say - 'in the blink of an eye', thus one can imagine the difficulty for a bystander to observe how the user fell, and even harder for the cyclist to pinpoint precisely the position of their head through the fall. The intention behind this feature was specifically for physicians to get a more accurate view of the fall. It is hoped that such data would enable an easier diagnosis of potential injuries on or around the cyclist's head.

This report will focus on the iOS app interface of Helmo, which will be written in Swift 3. As will be explained below, smartphone plays a central role in the full system, connecting each of the individual subsystems together and providing the user with methods of interacting with the system. Thus, a brief overall view of each of the components project will be presented, for the reader to understand how and why they fit in the app and system as a whole.

II. Related Work

Carv is a ski wearable created by the start-up company Motion Metrics Ltd [4]. The product consists of two insoles to be fitted inside a pair of ski boots. Each sole contains 48 capacitive pressure sensors; they are connected to battery powered tracker units that also contain an IMU for tracking

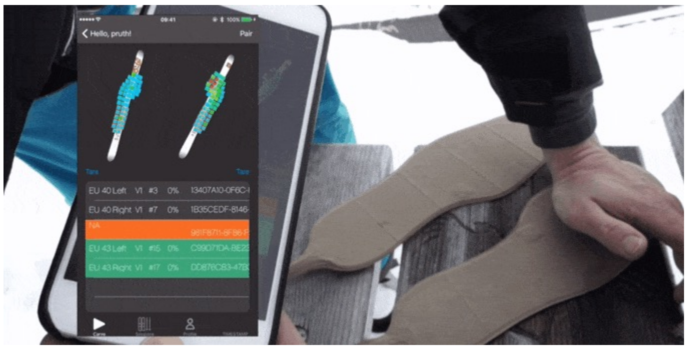

Figure 1: Carv app showing a visualization of the ski orientation and force along the surface of the sole as pressure is applied [4]

ski orientation, one located on each ski boot. As with Helmo, BLE is used to sync between the external hardware and the smartphone app. Slope descents are stored under a database, which the skier may access with an account inside the app. Ski position, as well as applied foot pressure may be viewed at any point along the path. A real-time visualization of the skis and forces experienced by each of the sensors is available inside the app, as shown in figure 1.

Compared to Carv, Helmo shares a number of similarities. Firstly, both projects measure the same two quantities namely physical force spread over an area and device orientation. Second of all, both projects link to a modern smartphone via BLE. Thirdly, information flows from the embedded tracker through the smartphone and to the cloud in a similar fashion, with the main difference being that Carv offers a subscription-based service that provides a more in-depth skiing analysis. Lastly, both apps have in-app 3D visualizations of the devices they are tracking – a feature that for Carv is an enhancement of user experience but for Helmo a potential diagnosis tool for healthcare providers.

The implementation of Carv's numerous pressure sensors is possible thanks to the relatively flat shape of a foot insole. Feet are also rather similar in shape amongst humans. Skulls, on the other hand are complex geometries that vary between people. [5]. Thus, it is far more challenging to design a densely packed grid of capacitive sensors spread over a larger, more irregularly shaped area. As such, Helmo will utilize readily available force sensitive resistors (FSRs) for this purpose.

In terms of inertial measurement, the ski tracker is processing and storing a data stream sampled at a relatively modest rate of 220Hz [4]. Since Helmo is only meant to store short bursts of data, an IMU capable of higher data rates is desirable. Inside the helmet, the data rate of the IMU (as well as of the ADC responsible for the FSRs) will be maintained low during ambient operation and throttled up as soon as the onset of a fall is detected, in order to save power without sacrificing accuracy.

User accounts are present in both projects. In Carv, the user's descents are captured, along with sensor data that was recorded along the way. On the other hand, Helmo, records cycling commutes albeit only with basic statistics such as distance covered or time spent on journey. Raw data from the helmet would be useless to the user, outside the context of a fall. This also relaxes the storage requirements on the user database.

3D rendering is offered by both Carv and Helmo. Two main rendering libraries are available for iOS – Open Graphics Library (OpenGL and Metal). Since Carv is multi-platform (iOS and Android), there is an inclination to think that the universal OpenGL platform has been used for this purpose. While code portability is a clear advantage of OpenGL, speed and efficiency when running on a modern GPU enabled iOS device are sacrificed in comparison to the native Metal framework [6]. Since Helmo is only being built with iOS in mind, the Metal framework shines as the superior candidate between the two, for accomplishing this purpose.

The large number of similarities between the two projects will serve the Helmo app well in terms of user experience (UX) and user interface (UI) design cues, given the success of the Carv system on the Kickstarter platform.

III. System Overview

A systems level view of Helmo will reveal three major components: the embedded device, iOS app, cloud server backend and database. The helmet-mounted device contains an IMU and a number of force sensors in contact with the cyclist's head, as mentioned before. Falls detected on-board the helmet trigger an upload of sensor data to the user's smartphone that is BLE-connected to the helmet. Data is analyzed on the smartphone more thoroughly to determine fall severity and then sent through the server application programming interface (API) to the database. Each user logs in to the app at start-up with either a physician or cyclist account. Physicians can register cyclists as patients and receive alerts whenever their patients experienced a fall. They may then visualize the patient's recorded data in 3D, within the app. In the case of falls, cyclists will be prompted by the app for a small physical/audible action to avoid their physician's phone being called.

Inside the helmet, a lightweight version of the fall detection algorithm will be constantly running on sensor data that is sampled at a relatively low data rate. In order to preserve battery life, CPU clock speed will be reduced in ambient running mode and will be boosted as soon as increased data sampling rate is desired i.e. during a fall. A fall being detected results in a notification being sent to the smartphone asynchronously at the end of the fall, action that is enabled by the Bluetooth LE standard [7]. The smartphone then reads the data that was recorded on the helmet's memory into local memory, and classifies its severity with a more complex version of the device-side fall detection algorithm.

Synchronization of local user fall data and the database is performed as soon as a network connection is available. The remote cloud database, being MySQL, is unreachable directly from a smartphone through iOS's native programming languages namely Objective-C or Swift. As such, a server backend is required. Its purpose is to receive data from client-side requests in the app as well as respond to queries. Data to/from the client will originate/sink into the database.

Doctor accounts will have access to their patient's data in the measure that the cyclist allows (in order to give the cyclist a level of privacy). Push notifications will be sent to the physician's smartphone when a fall is detected.

IV. Backend Interface

The backend will expose a series of methods to the client in a standard Representational State Transfer (REST) API [8]. It will be built using the LAMP (Linux, Apache, MySQL, PHP) technology stack. Users will authenticate to the API with a token that is refreshed every 24 hours, using the open-source industry-standard authentication framework, OAuth2 [9]. The flow of backend actions to be performed by the smartphone on the cyclist side is therefore as follows:

1. *The user requests an authentication token.*
2. *Once a fall is detected, a POST request is issued to the API, with the active authentication token, to create a new fall entry under that username, along with all the sensor and other relevant data – if an Internet connection is available.*
3. *Should an internet connection not be available, the data is stored locally on the smartphone using SQLite 3 [10] [11] APIs and be uploaded as soon as the client is reconnected to the API.*
4. *The history of falls experienced by the user is re-fetched to feature the newly added database entry available for closer inspection.*
5. *Whenever the user desires a 3D inspection of any one of their falls, the fall's sensor data is fetched from the API and rendered on the iPhone GPU.*

The physician's app shares the fall visualization component with the cyclist version of the app. To make the distinction between account types (done so on the backend side by the token identity), physician accounts would internally have access to a different table of the MySQL database that would reference the patients' sensor data, as well as general user details such as name/age/etc. (which would be set by the cyclist when they first install the app). Inside the medic's table view of patient falls, such user details would be available in the table view – a feature that is unnecessary on the cyclist version of the app.

Furthermore, the physician account would automatically subscribe upon login for push notifications from the backend using Apple's APNS technology [12]. The backend side would utilize the open-source ApnsPHP framework [13] to interface to the phone. Whenever a patient experiences a fall, the physician to whom the cyclist belongs would receive a notification from the API and the newly received table entry would be available for closer inspection.

V. Bluetooth Communication

A module from Cypress's PSoC 4 BLE range is utilized on the device-side for Bluetooth communication, namely the CYBLE-214009-00 [14]. There are three reasons behind choosing this specific module. Firstly, the increased on-board available memory (Flash and RAM combined) when compared to other Bluetooth SoCs [15] means that more sensor data can be stored on the chip, or the data sampling rate can be increased to higher levels without sacrificing the available storage space of a fall. Secondly, the module is of miniature dimensions, suitable for fitting inside a helmet – 11x11x1.8mm. Finally, the manufacturer provides clear, up-to-date and easy-to-follow documentation on configuring the Bluetooth stack for custom applications as well as for linking with the device from the smartphone side. Hardware debug tools for the module are available at low cost and software debug tools are free, both on the PC and the smartphone.

Bluetooth devices can be viewed architecturally at multiple layers. At the highest layer, there is communication between a master (iPhone) and a slave (helmet). High-level topology, known as Generic Access Profile (GAP) dictates whether a system is set up for broadcast (as either a broadcaster or an observer, similar to a beacon) or whether it’s designed to connect directly to another device. In the case of Helmo, the GAP will be a connecting peripheral, and the smartphone a connecting central, in BLE terminology. The central initiates the connection to the peripheral.

Every GAP profile on a BLE device has a Generic Attribute Profile (GATT). This is a mechanism that BLE utilizes to transmit information between the two connected devices. GATT servers store data (such as fall detection data), which GATT clients may then retrieve. Attributes may be created to feature subscription services, that is a notification is to be sent asynchronously from the client to the device whenever that attribute changes value. The helmet will feature a ‘fall detected’ attribute that will allow the smartphone to subscribe to it.

After the notification of a fall being detected has been transmitted to the iPhone, the sensor data from the fall will be made available under other known attributes. These attributes will be accessed by the phone, and the sensor data will be downloaded locally.

VI. Conclusion

To conclude, Helmo is a smart helmet that watches out for the safety of cyclists by detecting falls. The helmet connects to an app on the user’s iPhone, that is synced through the cyclist’s account to a database via a backend. Whenever falls are detected, they are recorded on the respective user’s account in the database, and can then be visualized by either the cyclists themselves or their physicians in 3D within the app. Processing is done on the fall data to categorize their severity

Cyclists have a limited amount of time to react post-fall, before the physician is alerted telephonically. The hope for Helmo is to help improve diagnosis of fall-related injuries that result from cycling.

VII. Bibliography

[1] Office for National Statistics, "2011 Census Analysis - Cycling to Work," Office for National Statistics, London, 2011.

[2] Department of Transport, "British Social Attitudes Survey 2015: Public attitudes towards transport," Department of Transport, London, 2015.

[3] Department for Transport, "Reported road casualties in Great Britain: main results 2015," Department for Transport, London, 2015.

[4] Motion Metrics Ltd., "Kickstarter CARV: The world's first wearable that helps you ski better," February 2016. [Online]. Available: https://www.kickstarter.com/projects/333155164/carv-the-worlds-first-wearable-that-helps-you-ski/description. [Accessed 23 February 2017].

[5] J. Blumenfeld, "Racial Identification in the Skull and Teeth," *Totem: e University of Western Ontario Journal of Anthropology,* vol. 8, no. 1, pp. 20-33, 6 June 2011.

[6] Apple Inc., "About Metal," Apple, 12 December 2016. [Online]. Available: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/MetalProgrammingGuide/Introduction/Introduction.html. [Accessed 24 February 2017].

[7] Bluetooth SIG, Inc., "Generic Attributes (GATT) and the Generic Attribute Profile," Bluetooth SIG, Inc., 2017. [Online]. Available: https://www.bluetooth.com/specifications/generic-attributes-overview. [Accessed 24 February 2017].

[8] R. T. Fielding, " Architectural Styles and the Design of Network-based Software Architectures," University of California, Irvine, Irvine, 2000.

[9] OAuth, "OAuth2," OAuth, [Online]. Available: https://oauth.net/2/. [Accessed 24 February 2017].

[10] SQLite, "SQLite Release 3.17.0," SQLite, 13 February 2017. [Online]. Available: http://www.sqlite.org/releaselog/3_17_0.html. [Accessed 24 February 2017].

[11] S. Celis, "SQLite.swift," 28 January 2017. [Online]. Available: https://github.com/stephencelis/SQLite.swift. [Accessed 24 February 2017].

[12] Apple Inc., "APNs Overview," 27 October 2016. [Online]. Available: https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1. [Accessed 24 February 2017].

[13] Immobiliare, "ApnsPHP: Apple Push Notification & Feedback Provider," Immobiliare, 27 September 2016. [Online]. Available: https://github.com/immobiliare/ApnsPHP. [Accessed 24 February 2017].

[14] Cypress Semiconductor, "CYBLE-214009-00 EZ-BLE PSoC Module Datasheet," [Online]. [Accessed 24 February 2017].

[15] Argenox, "A Guide to Selecting a Bluetooth Chipset," Argenox, 19 May 2016. [Online]. Available: http://www.argenox.com/bluetooth-low-energy-ble-v4-0-development/library/a-guide-to-selecting-a-bluetooth-chipset/. [Accessed 24 February 2017].