# A Smartwatch-based User Interface for In-Vehicle Interactions

Laura-Bianca BILIUS
MintViz Lab | MANSiD Research Center, Stefan cel Mare University of Suceava
laura.bilius@usm.ro

***Abstract*—Web-based standards and technology, such as HTTP, JavaScript, and WebSockets, can bring many benefits to designing and engineering systems for in-vehicle infotainment (IVI). In this paper, we present the technical details of a software application for in-vehicle interactions that features WebSockets-based communications between the driver's smartwatch and a tablet installed in the car that runs a web-based interface and demonstrates features representative of those found in IVI systems. Our implementation is based on a Node.js server application that processes incoming data from the smartwatch, e.g., a list of preferred songs or the driver's heart rate, and delivers that data to the in-vehicle infotainment system.**

***Keywords—WebSockets, JSON, in-vehicle interactions, user interfaces, smartwatch, Node.js, JavaScript***

## I. INTRODUCTION

The Internet-of-Things (IoT) is developing more and more because it enables the connection and communication between smart objects. With the development of web technologies applied for the automotive industry, device communications also extend to in-vehicle environments (IoV) [1]. In-vehicle environments are developing a lot to include many systems such as navigation, sensors, and features that demand increasingly more attention from drivers.

Smartwatches are used increasingly more by users because of the features they offer. Many people prefer wearing those thanks to health tracking (count heart rate, pulse rate, calories, steps, sleep), accessing messages, receiving calls, seeing social media notifications, and so on. The smartwatches are described as "someone who knows more about what you need than you do", devices which "don't just tell the time" or "there is nothing better than a smartwatch to keep you connected" [2], the reason why the smartwatch industry continue to grow.

Android Auto [3] and Car Play [4] are both connecting the smartphone to the car display making the driving safer reducing the distraction. Our implementation allows a connection with the user's smartwatch that is connected with the smartphone. The advantage is that it is displayed, besides phone data, the smartwatch data provided by the sensors, such as heart rate, steps number, hand kinematics, and data stored on the memory.

Some health problems could occur during driving, which may be unnoticeable by drivers but can affect safety. Due to the information received from the smartwatch, the driver will know all the time his health status, and in case of a negative result, it can prevent an accident.

In this paper, we describe an implementation of a smartwatch-based user interface that employs WebSockets to enable interactions between the smartwatch and an in-vehicle system. Our application consists of a Node.js server that receives data from the smartwatch and delivers it to the in-vehicle infotainment system modeled with a tablet device running a web-based user interface. To ensure safety during driving, we aim to connect the smartwatch with the in-vehicle user interface and all the desired information to be displayed and easy to read. The purpose of our system is to reduce driver's distraction, with a friendly user interface that reduces the need to check the smartwatch on the wrist and makes the driver keep their hands on the wheel and eyes on the road.

## II. RELATED WORK

The popularity of web-based systems is making a lot of progress, taking into account that users own more and more smart-devices. There is plenty of applications that combine the safety and entertainment of drivers, such as Apple Map's, Car Play, and Google Android Auto which allows the connection of user devices with the vehicle [5]. Continental's AutoLinQ System [6] is an Android-based application that offers connection with Android devices, designed to increase the drivers' user experience with their personal mobile devices while inside the vehicle.

Isenberg et. al. [7] developed an application that provides vehicle data to the user using a WebSocket connection which can be executed on multiple devices. Another paper describes an in-vehicle system able to integrate other devices and services such as navigation, vehicle sensors, advanced driver assistance systems, cameras, and so on. The menu of the system was implemented using web technologies and was provided through a web server (see [8]).

IoT is changing by integrating continuously the smartness into everyday life, squeezing more and more in-vehicle environments [9]. IoV will become indispensable for safe driving, traffic control, crash response, convenience services, and infotainment. Even though there are plenty of advantages, exists some challenges that slow down the spread such as storage of big data, security and privacy, the malfunction of cars, sensors and network hardware, mobility, and standardization [1].

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI - UEFISCDI, project no. PN-III-P1-1.2-PCCDI-2017-0917 (21PCCDI/2018), within PNCDI III.

978-1-7281-6870-8/20/$31.00 ©2020 IEEE

## III. Design Architecture and System Description

Imagine a driver that gets in his car and, before departing, connects his smartwatch to the in-vehicle user interface. With a single touch, the in-vehicle navigation system displays information about the user. All this information is updated constantly displaying real-time data received from smartwatch sensors or smartwatch memory without taking the hands off the wheel while driving.

The software architecture implies a server application, a smartwatch client, and eventually other devices for the in-vehicle environment (such as smartphone, tablet, and navigation) that can have any display resolution assuming that users have many devices with different sizes of screens.

The software architecture to access data for the in-vehicle environment is depicted in Fig. 1. The architecture approached allows providing data to a vehicle from a smartwatch through Web Applications. We approached WebSocket based on advantages offered, especially because after a connection is established, the server and the client can send messages to each other any time, making the perfect candidate for real-time applications [10]. The properties of our software application are as follows:

- *Web-based*: We approached this technology which allows to the users to interact with a server through a web browser interface and exchange data using JSON data format.
- *Open source*: We used open source technologies for public access and free usage.
- *JavaScript orientation*: We used this programming language because is used everywhere on the web, has numerous resources, and allows the use of Node.js servers.

### A. Smartwatch Interface

As Fig. 2 shows, the interface for the smartwatch exposes featured for users. The data can be send data automatically/implicitly that is fully synchronized (all functions of the actual prototype), or manually/explicitly that is partially synchronized (the desired functions of the actual prototype). For demonstration only, we choose to send information such as personal data (username, age, last connection data, heart rate value, hand kinematics value, and, steps number), music, and user task list. The user can add or remove tasks from the list from the smartwatch. The user can stop the exchange of data by choosing the stop option, so the WebSocket connection is closed. The data such as user information, task list, and music are stored in the smartwatch memory, and heart rate, steps number, hand kinematics are obtained from smartwatch sensors. As long as the smartwatch is connected to the system, all the data is sent at every half second to update the displayed information on the user interface, whether is implicit or explicit option.

Regarding the implementation of a smartwatch user interface, Tizen Studio IDE demands permissions or features within an application to give users rights. To protect the device system and user data, Tizen provides features and privileges for safe operations. To develop an application, firstly Tizen IDE checks if the privilege declared is allowed. The information from sensors of devices is acquired with the help of functions from features of Sensor API.

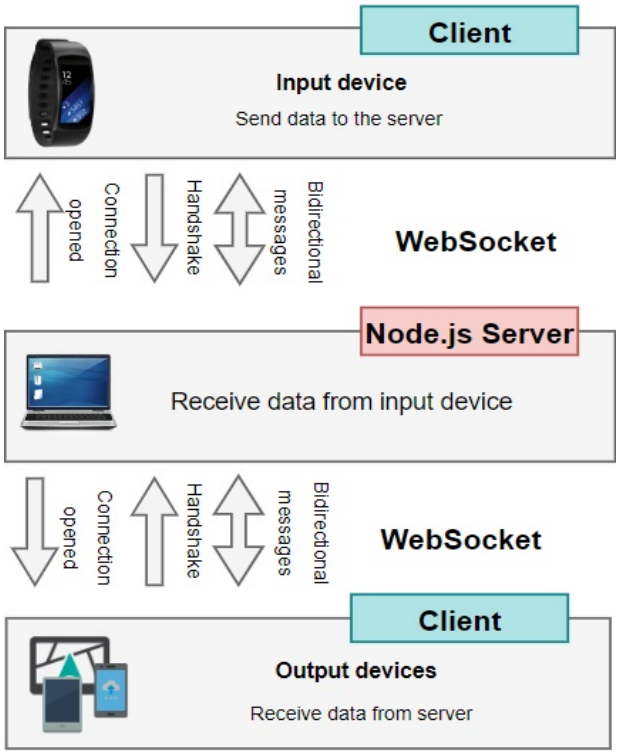


Fig. 1. System architecture based on Websockets communications.

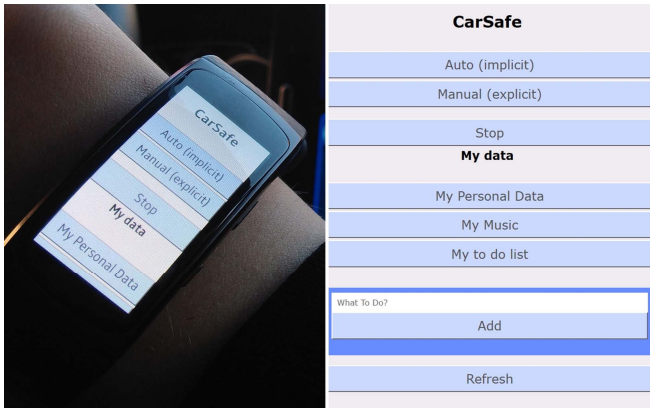


Fig. 2. Smartwatch user interface showing various options regarding data exchange between the personal device and the in-vehicle system.

### B. User Interface

The in-vehicle user interface is simple and easy to use. Once data is received, the information is synchronized in real-time on the tablet or navigation. In this way, the drivers have the opportunity to have real-time health information, as well as the music and the task list without having to touch or check the smartwatch while driving.

## IV. TECHNICAL PROTOTYPE

To implement our prototype, we used Tizen Studio 3.6 and NetBeans 8.2 IDE open-source software platforms. We designed and implemented an application for a wearable device, the Samsung Gear Fit2, using the Tizen environment approaching the native Web languages such as HTML5, CSS, and JavaScript. According to [11], "A wearable Web application is basically a Web site stored on a wearable device". Also, we designed the web interface for in-car devices or personal devices of users except smartwatch using NetBeans that is an integrated development environment (IDE) using languages such as HTML5, JavaScript, and CSS. This choice of technology represents a simple instantiation of the Euphoria software architecture [12] that enables the design of in-vehicle software applications by adopting the perspective of a smart, connected car as a smart environment [13].

Starting with the fact that the WebSocket constructor initiates a connection with the server, the client can use the *send* method to push data to it. At the same time, the client can receive data from the server using an *onmessage* handler. The goal is to provide at the same time all the smartwatch stored data and sensors values to the display through a JSON message, while there is an opened connection between smartwatch, server, and user interface. The client sends the information in JSON data format, taking into account that into JSON data it can be included strings, integers, arrays, and other different data types. In Fig. 3 is an example of a JSON message for implicit option. The JSON object has 2 important methods: *parse* and *stringify*. *JSON.parse()* takes a JSON string and transform it into JavaScript object, while *JSON.stringify()* do the opposite [14].

The WebSocket server was implemented in JavaScript with Node.js. To create the communication between server and client, we also need a library for a real-time web application such as Socket.IO. According to [15], "it has two parts: a client-side library that runs in the browser and a server-side library for Node.js". Firstly, we create a WebSocket Server instance on port 4201, which is the number for the server to listen on.

```
[
   {
       "personal_info":
           "username" : "Ana",
           "age": 29,
           "lastConnection": "November 26,2019",
       "music" : [
           "Fun – We are young",
           "Adele - Hello",
           "Coldplay – Viva la vida"],
       "to_do_list" : [
           "grocery store",
           "pick up Kevin to the airport",
           "call Maria"]
    }
]
```

Fig. 3. Example of a JSON message used in our application.

Once the connection is established, the server listens for a request from clients. Secondly, we implement the client-side JavaScript which establishes a connection to the WebSocket server. The server ran on an ASUS laptop with 16 RAM. We used four event handlers from WebSocket class such as *onopen* (will print a message for a successful connection), *onclose* (will close the connection with the server), *onerror* (WebSocket errors) and *onmessage* (data received) [16]. The most important event is *onmessage* because is called every time a message is received from the server (see Fig. 6). Regarding our application, there is an open connection and the driver can send the data from his smartwatch to the server. The JSON message collects the smartwatch sensors values and stored data, and at every half second, it sends to the server.

Once the server receives the message it sends to the user interface. For this, we create a JavaScript object to collect the information generated by smartwatch. We used the *push()* method to add items at the end of the JavaScript object (see Fig. 4 and 5). For example, in Fig. 4 it is presented the implementation code of the heart rate values acquisition generated by the smartwatch sensor. In Fig. 5, it is presented the implementation code of the personal data acquisition by driver to the JavaScript object, such as the name and age of the user, and the date of the last connection. On a click event, the collected data is converted into a JSON object and is sent to the server. The server receives from smartwatch the JSON data converted from a JavaScript object into a JSON object. The server provides the message which contains the received data to another client, in our case, a tablet (Samsung Tab A). The messages received from a server are always a string so the purpose is to convert the JSON object into a JavaScript object, using *JSON.parse()* method (see Fig. 6).

```
var data = { "userinfo": { "username":[], "age":[],
             "last_connection":[] }, "heart_rate": [],
             "macc":[], "steps":[], "music": [],
             "taks":[]
};
var hrvalue = []; // JavaScript object declaration
tizen.humanactivitymonitor.start('HRM', onHRValue);
function onHRValue(hrmInfo) {
    hrvalue = hrmInfo.heartRate;
    if (hrvalue > 55 && hrvalue < 120)
        heart_rate.push(hrvalue);
    else
        heart_rate.push("NAN");
}
```

Fig. 4. Code showing heart rate data acquisition from the smartwatch.

Other information, such as heart rate, number of steps, hand kinematics provided by smartwatch sensors and, music and tasks list are obtained using the same principle as in Fig. 4 to Fig. 6. The data is displayed on the HTML document using the *innerHTML* DOM property. When the browser loads the web page, the client sends a WebSockets handshake.

```
document.getElementById("personal_data").onclick =
function(event) {
   event.preventDefault();
   data.userinfo.username.push(username);
   data.userinfo.age.push("32");
   data.userinfo.last_connection.push(getdata());
   wsconnection.send(JSON.stringify(data));
};
```

Fig. 5. JavaScript code implementing information sent to the server.

```
var wsconnection = new
WebSocket('ws://192.168.200.15:4201');
wsconnection.onmessage = function (event) {
   console.log('Received', event.data);
   // create JavaScript Object from a JSON string
   var received_data = JSON.parse(event.data);
   // write received data on the HTML document
   document.querySelector('#username').innerHTML =
     received_data.userinfo.username[
     received_data.userinfo.username.length-1];
   document.querySelector('#age').innerHTML =
     received_data.userinfo.age[
     received_data.userinfo.age.length-1] + " years";
   document.querySelector('#time').innerHTML =
     "Last connected on " +
     received_data.userinfo.last_connection[
     received_data.userinfo.last_connection.length-2];
};
```

Fig. 6. JavaScript code implementing the WebSockets connection.

Once the clients are connected to the server, the exchange of messages can take place at any time. WebSocket protocol provides a way to change data between clients and, server with a single opened connection [17]. Communications between devices were implemented using WebSocket technology. The WebSocket protocol provides a full-duplex connection and takes place in two steps [18]. To establish the connection, the client sends a Web Socket handshake request and the server responds [10]. The web server was implemented using Node.js [19], a platform based on JavaScript for server-side and networking applications. To exchange data between browser and server, we used JSON since it is compact, easy to parse and load in JavaScript, can be generated rapidly, and is a universal data structure available in all modern browsers [20].

## V. Conclusion and Future Work

In this paper, we developed a smartwatch user interface approaching WebSocket communications using JSON objects to transmit messages for the in-vehicle environment. The application focuses on drivers by offering the possibility to obtain information while driving without the need to check the smartwatch. The data is displayed on the navigation or a tablet that has an internet connection from the in-vehicle environment. The advantages of our application imply an easy implementation, portability of Web applications which makes it suitable for any device and the data exchange in real-time.

Future work will focus on extending our application focusing by means of distributed user interfaces and by considering the findings of surveys of gesture and voice input for in-vehicle media consumption [21]. Another objective is to implement other in-car functions, such as making and taking phone calls, sending and receiving messages, setting alarms, calendar events, processing notifications as well as to include other sensors and input devices for in-vehicle interaction [22].

## Acknowledgment

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI-UEFISCDI, PN-III-P1-1.2-PCCDI-2017-0917 (21PCCDI/2018), PNCDI III.

## References

[1] Sadiku, Matthew & Tembely, Mahamadou & Musa, Sarhan. (2018). Internet of Vehicles: An Introduction. International Journal of Advanced Research in Computer Science and Software Engineering. 8(11). 10.23956/ijarcsse.v8i1.512.

[2] “9 Huge Advantages of Having a SmartWatch.” Smart Geek Wrist,2020, Available at: www.smartgeekwrist.com/advantages-of-smart-watch/.

[3] Android Auto, Available at: https://www.android.com/auto/.

[4] Apple CarPlay, The ultimate copilot, Available at: https://www.apple.com/ios/carplay/.

[5] Fangchun, Yang & Wang, Shangguang & Jinglin, Li & Liu, Zhihan & Sun, Qibo. (2014). An Overview of Internet of Vehicles. Communications, China. 11. 1-15. 10.1109/CC.2014.6969789.

[6] Amend, James M. “Continental's AutoLinQ System 'Smarting Up' Vehicle Infotainment.” WardsAuto, 4 Dec. 2011,Available at: www.wardsauto.com/news-analysis/continental-s-autolinq-system-smarting-vehicle-infotainment.

[7] Simon Isenberg, Wolfgang Haberl, Matthias Goebl, Maximilian Michel and Uwe Baumgarten, “Enabling Rich Web Applications for In-Vehicle Infotainment”, Web and Automotive, W3C, 2012 (11).

[8] Michael Eichhorn, Martin Pfannenstein, and Eckehard Steinbach. 2010. A flexible in-vehicle HMI architecture based on web technologies. In Proceedings of the 2nd international workshop on Multimodal interfaces for automotive applications (MIAA ’10). Association for Computing Machinery, USA, 9–12. DOI:https://doi.org/10.1145/2002368.2002374

[9] O. Kaiwartya et al., "Internet of Vehicles: Motivation, Layered Architecture, Network Model, Challenges, and Future Aspects," in IEEE Access, vol. 4, 5356-5373, 2016. doi: 10.1109/ACCESS.2016.2603219.

[10] V. Pimentel and B. G. Nickerson, "Communicating and Displaying Real-Time Data with WebSocket," in IEEE Internet Computing, vol. 16, no. 4, pp. 45-53, July-Aug. 2012.doi: 10.1109/MIC.2012.64

[11] Watch – Build, Samsung Developers. Available at: https://developer.samsung.com/galaxy-watch-develop/creating-your-first-app/web.html.

[12] Ovidiu-Andrei Schipor, Radu-Daniel Vatavu, Jean Vanderdonckt. (2019). Euphoria: A Scalable, Event-Driven Architecture for Designing Interactions Across Heterogeneous Devices in Smart Environments. Information and Software Technology 109. Elsevier, 43-59

[13] Ovidiu-Andrei Schipor, Radu-Daniel Vatavu. (2019). Towards Interactions with Augmented Reality Systems in Hyper-Connected Cars. Proceedings of HCI Engineering 2019, the 2nd Workshop on Charting the Way Towards Methods and Tools for Advanced Interactive Systems

[14] Ben Smith, "Beginning JSON 1st ed. Edition", Publisher: Apress; 1st ed. edition (February 22, 2015), ISBN-10: 9781484202036.

[15] “Getting Started with Socket.io.” Scaleway, [online] Available at: www.scaleway.com/en/docs/how-to-install-and-configure-socket-io.

[16] Andrew Lombardi, “WebSocket”, Editors: Simon St. Laurent and Brian MacDonald, First edition September 2015, ISBN: 978-1-449-36927-9.

[17] Fette I., Melnikov A.“The WebSocket Protocol” RFC 6455 (2011):1-71.

[18] Qigang, Liu & Sun, Xiangyang. (2012). Research of Web Real-Time Communication Based on Web Socket. International Journal of Communications, Network and System Sciences. 05. 797-801. 10.4236/ijcns.2012.512083.

[19] Node.js, Available at: https://nodejs.org/en/.

[20] Daniel J LeBlanc, “Learn to Create JavaScript Object Literals and JSON Data”, 2018, Available at: https://developing-an-organic-web-channel.ghost.io/2018/08/16/getting-started-with-json-javascript-object-notation/

[21] L. B. Bilius, R.-D. Vatavu, “A Synopsis of Input Modalities for In-Vehicle Infotainment and Consumption of Interactive Media,” Proc. of IMX ’20, the Int. Conference on Interactive Media Experiences, 2020.

[22] B.F. Gheran, R.-D. Vatavu, “From Controls on the Steering Wheel to Controls on the Finger: Using Smart Rings for In-Vehicle Interactions,” Proc. of DIS '20 Companion, the ACM Conference on Designing Interactive Systems, 2020. https://doi.org/10.1145/3393914.3395851