Conversations with Connected Vehicles

Simon Mayer
Department of Computer Science
ETH Zurich, Switzerland
Email: simon.mayer@inf.ethz.ch

Josh Siegel
Field Intelligence Laboratory
Massachusetts Institute of Technology, USA
Email: j_siegel@mit.edu

***Abstract*—We present a system that allows drivers and fleet managers to interact with their connected vehicles both by means of direct control and indirect goal-setting. The ability to move data from vehicles to a remote server is established by the flexible and secure open vehicle telematics platform "CloudThink." Based on this platform, we present several prototypes of how people can be enabled to conveniently interact with connected vehicles: First, we demonstrate a system that allows users to select and interact with vehicles using object recognition methods and automatically generated user interfaces on smartphones or personal wearable devices. Second, we show how functional semantic metadata can be used to smooth the boundaries for interacting with vehicles in the physical and virtual worlds. Finally, we present a method for monitoring interactions between vehicles and remote services which increases safety and security by enhancing driver oversight and control over the data that leaves and enters their vehicle.**

A car whose telemetry data is projected into the "Cloud" and made accessible for third parties via standard Web technologies represents an interesting example of a Web-enabled smart thing. With vehicles being among people's most frequently interacted with objects and representing the location where they spend the most time outside home and the office, Web-enabling vehicles affords designers and developers a unique opportunity to improve quality of life. Usable vehicle data has the potential to conserve valuable natural resources, reduce aggravation, and improve safety. Integrating and connecting automobiles to the broader Web ecosystem paves the way to reducing transit cost and time through efficiency and reliability improvements, while also unlocking richer information and actuator access than is presently available. It thus provides a unique lever to improve user experience, a primary differentiator in today's vehicle market: from failure prognostics to comfort and convenience improvements, a well-connected car has the ability to change how people see and interact with transportation systems at large. Improved connectivity not only makes the vehicle user experience better, it also facilitates seamless integration of automobiles with other transit systems. By lowering barriers to vehicular application development, a whole new ecosystem of automotive applications may be developed, and OEMs, consumers, and infrastructure all stand to benefit from reduced cost and heightened quality and reliability. The key to making vehicular connectivity a widespread success lies in simplifying data acquisition and manipulation, and creating a platform that is brand-agnostic. In essence, the challenge is creating a system capable of supporting the many modes of data acquisition and actuation required by the various application builders while balancing the need for high safety and security for critical systems integrated in user's vehicles. Such a system then needs to be integrated with a means of enabling car owners as well as fleet managers to conveniently interact with their vehicles, irrespective of the make or model.

The basic infrastructure that we build upon in this paper to make cars accessible over the Web is supplied by *CloudThink*, a platform developed to make vehicle data and actuation capabilities usable for clients in near real time via an intuitive REST API. The use of *CloudThink* integrating cars into the *Web of Things* opens up many interaction methods for smart things to the automotive domain: we can recognize cars and interact with them directly via user interface devices such as smartphones or smartglasses, can visualize their interactions with other smart things and remote applications, and can semantically describe the data and services they provide to enable the automatic collaboration of vehicles with smart devices and remote Web services.

In this paper, following a brief introduction of the CloudThink platform, we discuss several applications that have been developed on top of the platform to make it easier for users and third-party applications to interact with connected vehicles. Specifically, we present a combination of a visual object recognition system with embedded model-based user interface descriptions that enables drivers to directly interact with their cars. We furthermore demonstrate the applicability of functional semantic service descriptions to automobiles – this system integrates cars with other smart devices and Web services, thus creating physical mashups that make use of data and functionality provided by the connected vehicles. Finally, we show a system that we created to help users stay in control of what data leaves their automobiles – and which commands enter them – by using an augmented-reality interface that visualizes interactions of clients with cars.

I. The CloudThink Platform

In the context of a collaboration between ETH Zurich, Massachusetts Institute of Technology (MIT), the Singapore University of Technology and Design (SUTD), the University of North Texas, and the Masdar Institute of Science and Technology in the United Arab Emirates, we have developed a vehicle-to-cloud communication platform called CloudThink [1]. The core of this system consists of a low-cost hardware platform (the *CARduino*) that connects to the on-board diagnostics port of a car using the OBD-II interface[1] and a secure back-end server that stores data uploaded from vehicles and makes it accessible to other applications via a REST API.

The main goal of CloudThink is to enable drivers to share data from their vehicles for processing by a wide variety of third-party applications while emphasizing an open, secure,

[1]This is a standard interface that every car manufactured in the E.U. and U.S. after the year 1996 must support. The CARduino supports only the Controller Area Network variant of the OBD-II specification.

978-1-4673-8058-4/15/$31.00 ©2015 IEEE

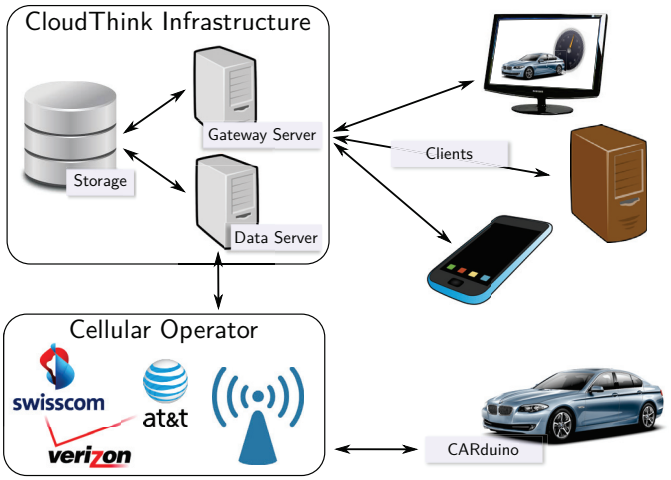


Fig. 1. Overview of the CloudThink platform. The CARduino hardware is deployed on the car and communicates with the CloudThink back end via a GSM link. In the back end, the data server parses, filters and synchronizes the data and stores it in a database. The gateway server provides a REST API for clients to conveniently access the data from a variety of devices.

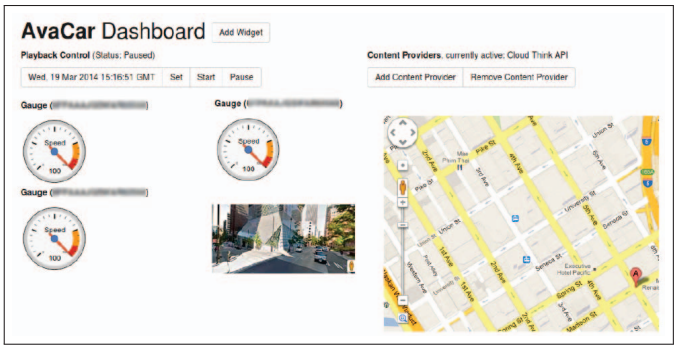


Fig. 2. A virtual dashboard that displays data from different sensors of a Web-enabled car. The current location of the vehicle is shown on a map or alternately in the form of a Google StreetView image.

and interoperable interface to these data. CloudThink thereby aims to create an "App Store" for cars that hosts applications which can access and process car telemetry, and in cases, control vehicle actuator function. The platform is distinguished from similar projects in the automotive domain in that it remains agnostic of the concrete type and make of vehicle and therefore enables data sharing across the boundaries of car manufacturers, improving data generation by unsiloing data storage and allowing for rapid, crowdsourced data collection. Use cases for the CloudThink system stretch from enhancing vehicle security (for instance, a "virtual leash" for tracking of stolen vehicles) to reducing the environmental impact of cars by helping drivers to find out how they can better economize fuel. Drivers could also use such a system to improve safety by adapting their driving style [2], and the platform could help set economic incentives to do so by enabling pay-as-you-drive (PAYD) or usage based insurance schemes [3], [4]. Potentially, drivers could also use CloudThink to monetize their driving data, for instance by sharing usage statistics and telemetry with the manufacturer of their car – or drivers could provide the same data to non-profit organizations to achieve better transparency of how specific vehicles fare in realistic situations, for instance with respect to their fuel consumption, maintenance effort, or to provide data leading up to a vehicle recall. Another highly interesting application domain is using real driving data for the improvement of vehicle characteristics, for instance determining the optimal battery size for electric cars [5]. On a larger scale, real-time feedback from vehicles could help improve road quality – and safety – by enabling road operators to quickly react to problems that range from potholes to developing traffic jams [6], [7], without the need for direct user reporting.

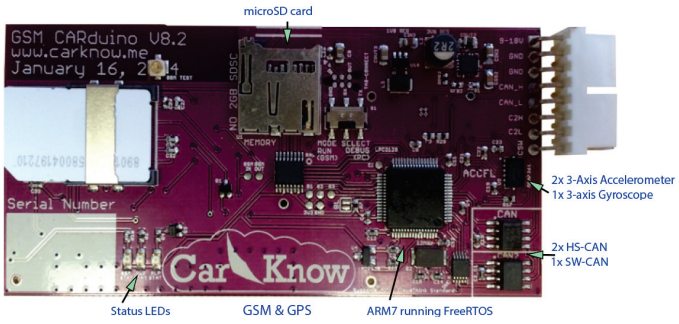
The CloudThink platform differs from contemporary OEM-backed application development systems. Interfaces proposed by auto manufacturers are limited in scope, with focus on intra-vehicle applications of data, such as within infotainment systems running locally on a car's head unit. In cases where direct, remote control of vehicle functions is allowed, these functions are controllable only through specific, walled-garden companion applications. The access of vehicle data through OEM-provided interfaces like OnStar's API or Ford's OpenXC is limited in breadth, capable of accessing standardized parameters and generic, but non-standardized pieces of information. Output modulation is limited to simple actuators, such as locks or other controls that act only when the vehicle is stationary, e.g. for car sharing applications. A system similar in nature to CloudThink is required to facilitate the best application development opportunities, by providing the richest possible vehicle data and actuation possibilities.

The main components of the CloudThink system are the *CARduino hardware*, the *data server* that stores data uploaded from individual cars in a local database, and the *gateway server* that enables third parties to access the stored data in a uniform way via a REST API (see Figure 1 on the previous page). Within the joint project, MIT is leading the hardware development (see Section I-A), while ETH's main responsibility is the implementation and maintenance of the back-end infrastructure (see Section I-B). On top of the system, the consortium has also created applications that provide better access to data stored on the platform in the form of "virtual dashboards" (see Figure 2) that could be used, for instance, by fleet managers, and has used telemetry data from CloudThink to create an application that gives drivers more insight into their own driving behavior, especially with respect to clues as to how they could improve their own fuel economy.

A. CARduino Hardware

Our CARduino hardware[2] (see Figure 3) is responsible for transmitting information from the vehicle to the CloudThink back-end infrastructure, where this data is stored in a SQL-based database. All information is sent via GSM connections and encoded in a custom message format that is tuned to minimize the volume of the transmitted data, thereby helping to keep communication cost low. Each message that is sent to the back end consists of the recording timestamp, the type of the transmitted data, the data itself, and a checksum to ensure accurate data capture. Messages are received by the CloudThink data server that parses and checks each message before persisting the contained data.

[2]The CARduino hardware is open-source. It is manufactured by CarKnow LLC., a company that has been incorporated by one of our partners in the CloudThink project.



(a)

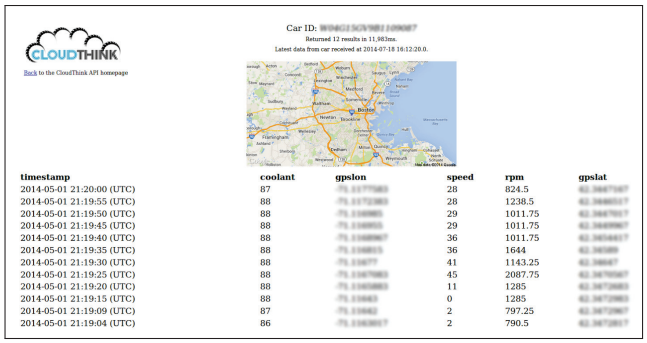
(b)

Fig. 3. (a) The CARduino hardware (ca. 9.3cm x 4.4cm x 1.5cm). One can recognize the SIM card and the SD card that is used for data buffering. The GPS and GSM antennas are attached to the bottom of the module. The "CarKnow" silkscreen refers to the name of the start-up hardware designer and distributor. (b) The CARduino hardware shown in a prototype 3D-printed casing and with an OBD-II connector cable. New variants plug directly into the diagnostic port.

The CARduino carries an on-board accelerometer and a GPS receiver, and can be configured to listen on a car's OBD-II interface for OBD messages that are tagged with specified OBD-II Parameter Identifiers (PIDs). This allows us to capture information about many internal processes of a vehicle, for instance basic driving data (e.g., speed, runtime, etc.) and information related to motor management (e.g., motor revolutions per minute, mass airflow, coolant temperature, etc.). In particular, we are able to gather all data required for calculating an approximation of the instantaneous fuel consumption of a car, which can be computed from the current vehicle speed and the mass airflow rate [8]. When the CARduino is unable to reach our back end, for instance due to poor network connectivity, it uses an on-board memory unit for buffering and transmits the contents of that buffer at the end of the trip. Finally, our hardware can also write to the car's OBD interface, thereby enabling it to actuate vehicle functions – we have successfully used it to lock and unlock cars, and to control wipers and power windows.

B. Back-end Infrastructure

After being stored in the CloudThink database by the data server, the vehicle information is processed by a synchronization script that aligns all received data points to common 5-second time windows using linear interpolation and, as part of this process, copies it to a different database. This script is also



timestamp	coolant	gpslon	speed	rpm	gpslat
2014-05-01 21:20:00 (UTC)	87		28	824.5	
2014-05-01 21:19:55 (UTC)	88		28	1238.5	
2014-05-01 21:19:50 (UTC)	88		29	1011.75	
2014-05-01 21:19:45 (UTC)	88		29	1011.75	
2014-05-01 21:19:40 (UTC)	88		36	1011.75	
2014-05-01 21:19:35 (UTC)	88		36	1644	
2014-05-01 21:19:30 (UTC)	88		41	1143.25	
2014-05-01 21:19:25 (UTC)	88		45	2087.75	
2014-05-01 21:19:20 (UTC)	88		11	1285	
2014-05-01 21:19:15 (UTC)	88		0	1285	
2014-05-01 21:19:09 (UTC)	87		2	797.25	
2014-05-01 21:19:04 (UTC)	86		2	790.5	

Fig. 4. Example responses of the CloudThink API's HTML interface to a request for a minute of car telemetry (GPS coordinates, coolant temperature, speed, and revolutions per minute).

responsible for calculating several metrics that are derived from the raw data – such as the instantaneous fuel consumption, and distance driven – and persisting these values together with the other synchronized data points.

It is the responsibility of the CloudThink gateway server to give client applications (which can be mobile applications, third-party servers, or Web browsers) access to the stored, synchronized data via a secure REST interface (SSL/TLS and HTTP Basic authentication). All data is provided in the JSON format for machine clients, and can be browsed and visualized using the HTML interface of the gateway server (see Figure 4). The server provides a layer of abstraction on top of the OBD PIDs that are used internally by our system and enables clients to request data about attributes such as "RPM," "consumption," and "speed" – information about which types of data a specific user is allowed to access on a specific car can also easily be loaded from the interface. Additionally, the gateway server enables clients to send actuation commands to vehicles.

Another responsibility of the gateway server is to enforce that only authorized users may access data stored in the CloudThink back end. To do this, it stores, for each registered user and application, the data fields that this user may access (e.g., GPS location, acceleration, etc.) as well as, for each vehicle, the users that are allowed to access data from this vehicle and whether they also hold actuation rights. The gateway server also keeps track of which user accesses which stored data item for billing purposes and to be able to inform each driver about who requests what kind of information about their cars. Finally, the gateway server features a public interface that enables developers of client applications to test their ideas and software prior to registering their applications as data users on the CloudThink platform. The capabilities of the gateway server are detailed in its online REST API description, along with example queries.

II. Talking to Connected Vehicles

Essentially, CloudThink aims to provide convenient access to vehicle data for third-party applications – how exactly these process the provided data to create advanced services for drivers and third parties is left to the application developers. We have developed several applications that make use of the data made available via the CloudThink platform, including a virtual dashboard and a fuel economy assistant. The primary topic of this paper, however, is how CloudThink can be used to *interact*

with Web-connected cars: to this end, we propose the automatic generation of user interfaces for vehicles and to combine this technology with an object recognition system for allowing users to select which car to interact with (see Section II-A), the semantic annotation of services that are provided by connected vehicles to enable the integration of their functionality within physical mashups (see Section II-B), and the visualization of interactions with Web-enabled cars using an augmented-reality interface (see Section II-C).

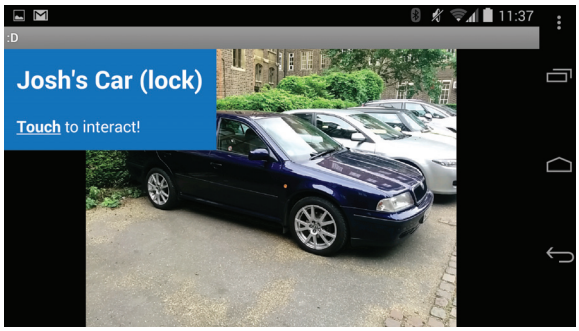
A. Direct User Interaction with a Vehicle

First, we focus on the direct interaction of a user with a connected car, meaning that we want to enable people to monitor telemetry data produced by a vehicle and to control actuators of a car using a user interface device, such as a smartphone, tablet, smartwatch, or smartglasses. Accomplishing this requires both: a means of selecting a vehicle to interact with, and rendering a suitable user interface to interact with it. We propose a system where a car is selected by means of a visual object recognition algorithm, i.e., making use of its distinctive visual features, and where the interaction takes place via automatically generated user interfaces.

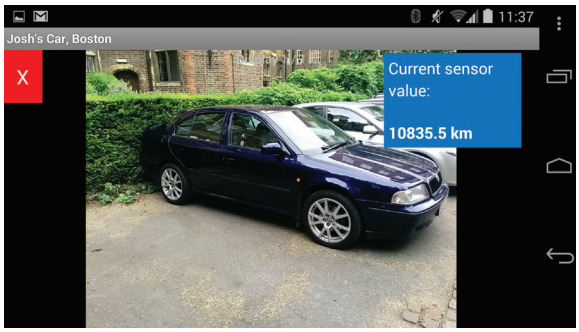
To select a vehicle to interact with, we make use of current visual object recognition methods that can be deployed on hand-held or wearable devices such as smartphones, smartwatches, or smartglasses (see Figure 5(a)). Our approach combines a Bag of visual Words (BoW) with linear SVMs and FAST/ORB feature detection/description and thus allows to recognize objects without relying on fiducial markers, meaning that our classification algorithm can be trained using only snapshots of the cars to interact with (see [9] and [10] for more information about the object recognition subsystem).

For automatically generating an interface that allows users to control the actuators of a vehicle, and monitor sensor values, we incorporate a model-based user interface generation system that relies on descriptions of the high-level semantics of interactions with Web-enabled devices (see [11] for more information). This system can be applied in the context of generating user interfaces for the sensors and actuators of Web-enabled cars without requiring any changes to the CloudThink system other than the embedding of hyperlinks to the relevant interaction descriptions.

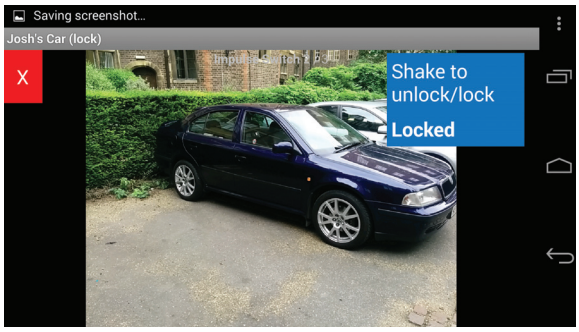
Together, the described technologies yield a system that permits users to directly interact with vehicles that are connected to the CloudThink platform and whose features are distinctive enough to enable them to be recognized as unique by our object recognition algorithm. As an example, Figure 5(b) demonstrates the interaction with the mileage "sensor" of a connected vehicle, while Figure 5(c) enables users to control a car's door locks: the embedded user interface descriptions allow rendering interfaces for multiple interaction modalities, including haptic, speech, and graphical.



(a)



(b)



(c)

(d)

Fig. 5. (a) Our smart things interaction software recognizes a car using its visual features. (b) Using the CloudThink API, a smartphone application can display values from sensors of a recognized vehicle and (c) provide user interfaces to control its actuators. (d) For cars, our visual object recognition software falls back to the few distinctive features of a vehicle, for instance the characteristics of its rims, stickers, and the number plate. Unfortunately, many of the other features are located "inside" the car, and not useful for differentiation in the most common use modalities.

Especially the application of our object recognition system

```
1  @prefix :<ex#>.
2  @prefix ex:<http://example.org/#>.
3  @prefix http:<http://www.w3.org/2011/http#>.
4  @prefix dbpedia:<http://dbpedia.org/resource/>.
5
6  {
7    ?car a dbpedia:Car; ex:hasVin ?vin.
8  }
9  =>
10 {
11   _:request http:methodName "GET";
12       http:requestURI ("https://api.cloud-think.com/things
             /" ?vin "/?parameters=latitude");
13       http:resp [ http:body ?gpslat ].
14
15   ?gpslat a dbpedia:Latitude; ex:derivedFrom ?car.
16 }.
```

Listing 1. Part of the RESTdesc description of the CloudThink API that describes how the current GPS latitude of a car can be obtained given its VIN, using an HTTP request to the CloudThink server. The server provides similar descriptions for other information about the cars it has data about.

within the context of recognizing vehicles gives rise to several challenges, predominantly due to the *visual similarity* of distinct vehicles: being based on visual object recognition technologies, our systems cannot differentiate between different cars that look alike, with the exception of cases where unique features of the car are visible in the camera frame (and related training images). While our approach is thus able to find and match visual features of vehicles, these are often located on the rims or the number plate of a vehicle, on custom stickers, inside the car, or on its reflective surfaces (see Figure 5(d)), which makes the robustness of our system heavily dependent on the current lighting conditions. Furthermore, especially in the case of "plain vanilla" cars without any distinct features, our software is not able to differentiate between cars of the same make, type, and color. Similar to humans, who often arbitrate between different cars that look alike by remembering their vehicle's parking location, we propose to partially overcome this challenge by using the GPS location of a car (which may be obtained from the CloudThink API) in conjunction with the location of the user interface device as an additional context parameter to uniquely identify a vehicle. However, while this technique can indeed help to achieve stable selection results, it introduces another challenge: the (remote or local) application that helps the user interface device arbitrate between different cars requires access to the locations of cars that are registered to the CloudThink platform or, as a minimum, an interface to send geographically bounded queries for a car (i.e., "Is car C currently located within 100 meters of the GPS coordinates `(Lat,Lon)`"). Other types of context information might be used as well for this purpose, for instance ambient beacons (e.g., Google PhysicalWeb[3]), to ensure the minimum viable amount of data are shared with unlinked parties.

B. Functional Semantic Metadata for Smart Vehicles

Apart from the direct interaction with connected automobiles, we have used functional semantic metadata in the form described in [12] to capture the functionality of the CloudThink API with respect to providing access to vehicle telemetry data of individual cars, where each car is uniquely identified using its Vehicle Identification Number (VIN). As an example, Listing 1 shows the semantic description of the API that enables clients to obtain the GPS latitude of the current location of a vehicle given its VIN: using Notation3, a superset of RDF, it specifies that, given a specific VIN (line 7 in Listing 1), the car's GPS latitude can be obtained (line 15) by sending an HTTP GET request to the specified URL (lines 11-13) – more specifically, that this latitude value is conveyed in plain text within the body of the HTTP response. Note that the correct car's URI is selected at runtime by substituting the `?vin` parameter with the supplied VIN (line 12).

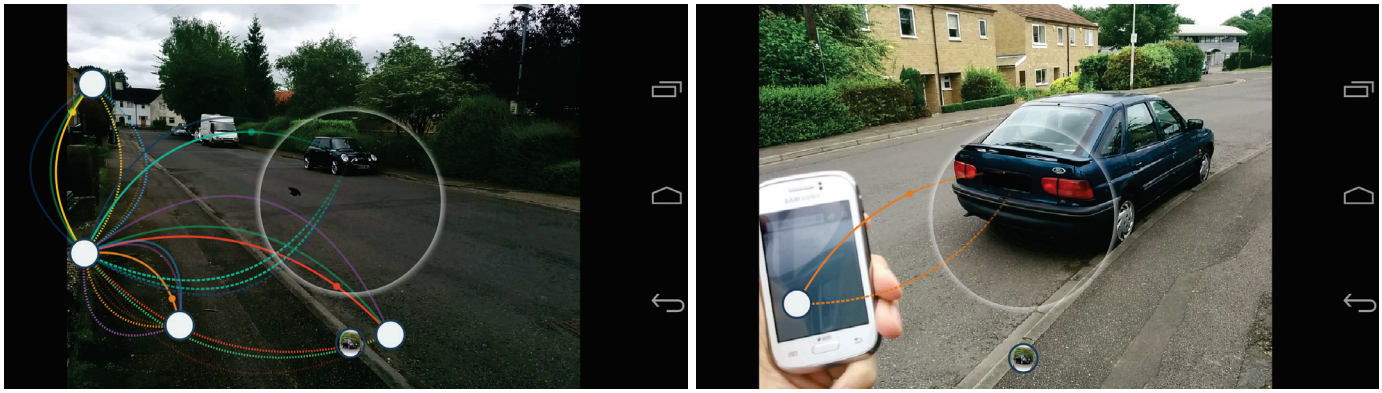
The CloudThink API contains a similar description for each parameter that can be accessed about its connected cars (e.g., their velocity, coolant temperature, and engine revolutions per minute). These descriptions can be used by semantic reasoners to integrate functionality across heterogeneous Web-enabled devices and Web services – to demonstrate that our system is capable of creating composite services on top of data produced by vehicles and published via CloudThink, we used its semantic descriptions together with those of a service able to display arbitrary images on a display, and semantically annotated two mapping applications (*Google Maps* and *OpenStreetMaps*). Given this setup, users can formulate a semantic goal that expresses their desire to obtain an *image* object that is also a *map* and is derived from the VIN of a *specific car*, and that a screen that is situated at the current location of the user should be displaying this image – this leads a semantic reasoner to propose that it could display the current location of a car on a nearby screen, by mashing up the CloudThink back end with either Google Maps or OpenStreetMaps and the screen API (see Figure 7). Our reasoning infrastructure, which is described in greater detail in [12], then sends the appropriate HTTP requests to the client which can execute them and thereby achieve the desired user goal. Because this service mashup is not hard-coded but rather derived on demand from the currently available semantic descriptions, the mashup seamlessly switches to using alternatives when services become unavailable, e.g. due to network outages – in this specific case, when we blocked access to one of the two mapping services, the mashup seamlessly switched to using the other, thus demonstrating the flexibility of our approach of using functional semantic metadata for dynamic service composition.

C. Monitoring Car Interactions

We believe that keeping users informed about what their connected smart things are doing in an increasingly connected Web of Things world – and why they are – will become increasingly important as many of our once isolated devices start interacting with each other and with remote services [13], especially if distributed services are flexibly or even automatically combined. Indeed, Web-enabled cars and platforms such as CloudThink represent an interesting domain where drivers would want to monitor which services access their vehicles: the ability to supervise and control interactions is relevant to protect drivers from attacks on their privacy such as location profiling, undesired actuator control, or the misuse of private data by authorities [14].

To support users to stay in control of their connected cars, we integrated a piece of software that logs HTTP requests and responses with the CloudThink back end and makes the collected data accessible to users in real time, using an augmented reality interface on their handheld devices [13].

[3]https://google.github.io/physical-web/



(a) (b)

Fig. 6. Our application visualizes interactions with a Web-enabled automobile that are brokered by the CloudThink platform using a visual object recognition algorithm and an augmented reality overlay on a handheld device. (a) Interactions of a client with a vehicle (and other endpoints). (b) Interaction of a handheld device with a vehicle.

Fig. 7. By publishing semantic metadata that describes its interfaces, functionality of the CloudThink API (i.e., access to sensor values and actuators of connected vehicles) can be flexibly integrated with other local and remote Web services. In this example, a user instructs an interface device (in this case, a smartphone) to display the location of his car on a map that is shown an a nearby computer screen. A reasoner then finds the necessary HTTP requests to reach that goal and executes them.

Using *Java Agents* which can modify the bytecode of Java classes and *class transformers* that can infuse any Web server based on the Grizzly NIO framework[4] with an HTTP request logger, the integration of the logging functionality with the CloudThink gateway server proved straightforward, and merely required a restart of the server with attached transformers. The combined system allows users to monitor interactions of clients with their vehicles using the Web interface of our visualization tool as well as displaying these interactions as a live overlay on handheld, or ultimately wearable, devices (see Figure 6). For the live overlay view, we again use our object recognition software to identify the vehicle whose information should be displayed: the arcs between endpoints represent HTTP connections and the dots traveling on these arcs represent actual pseudo-real time HTTP requests (for the visualization, we increase the request latencies to one second per request, however).

[4]https://grizzly.java.net/

III. Summary

The CloudThink platform, a joint effort of ETH, MIT, and several additional international partner institutions, has been created to make automobiles "first-class citizens" of the Web and enable third-party applications to easily access and process vehicle data while providing advanced services to drivers, fleet managers, and other players in the transportation industry. We and others have implemented several such applications on top of this platform, in particular in an effort to support users and applications when interacting with connected vehicles: using our systems, it is possible for users to directly interact with sensors and actuators of connected automobiles, monitor the communication of vehicles and clients in real time, and to seamlessly use functionality that is provided by cars within physical mashups.

The future developments for the CloudThink project include expanding the API to return interaction monitoring data for all available vehicles. Additionally, we plan to implement the ability for third-party application builders to run their own scripts and modify firmware behavior on the CARduino within safe and controlled boundaries, using an embedded virtual machine of sorts to limit the risk of malicious code execution and to protect sensitive vehicle networks.

Acknowledgements. This work was supported by the Swiss National Science Foundation under grant number 341627 and the SUTD-MIT International Design Center. The authors thank Yassin N. Hassan and Gábor Sörös for their help with the object recognition and interactions visualization software.

References

[1] E. Wilhelm, J. Siegel, S. Mayer, L. Sadamori, S. Dsouza, C.-K. Chau, and S. Sarma, "CloudThink: A Scalable Secure Platform for Mirroring Transportation Systems in the Cloud," *Transport*, vol. 30, no. 3, 2015 (in press).

[2] T. Lotan and T. Toledo, "An In-Vehicle Data Recorder for Evaluation of Driving Behavior and Safety," *Transportation Research Record*, vol. 1953, pp. 112–119, 2006.

[3] J. Paefgen, T. Staake, and F. Thiesse, "Resolving the Misalignment between Consumer Privacy Concerns and Ubiquitous IS Design: The Case of Usage-based Insurance," in *Proceedings of the 33rd International Conference on Information Systems (ICIS)*, 2012.

[4] J. Zantema, D. H. van Amelsfort, M. C. J. Bliemer, and P. H. L. Bovy, "Pay-as-You-Drive Strategies: Case Study of Safety and Accessibility Effects," *Transportation Research Record*, vol. 2078, pp. 8–16, 2008.

[5] R. C. Smith, S. Shahidinejad, D. Blair, and E. L. Bibeau, "Characterization of urban commuter driving profiles to optimize battery size in light-duty plug-in electric vehicles," *Transportation Research Part D: Transport and Environment*, vol. 16, no. 3, pp. 218–224, 2011.

[6] S. Rogers, P. Langley, and C. Wilson, "Mining GPS Data to Augment Road Models," in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 1999, pp. 104–113.

[7] W. Shi and Y. E. Liu, "Real-time urban traffic monitoring with global positioning system-equipped vehicles," *Intelligent Transport Systems*, vol. 4, no. 2, p. 113, 2010.

[8] Rick Walter, "Fuel Economy Data," 2014. [Online]. Available: http://www.cert.ucr.edu/events/pems2014/liveagenda/24walter.pdf

[9] S. Mayer, M. Schalch, M. George, and G. Sörös, "Device Recognition for Intuitive Interaction with the Web of Things (Poster Abstract)," in *Adjunct Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, ACM, 2013, pp. 239–242.

[10] S. Mayer and G. Sörös, "User Interface Beaming – Seamless Interaction with Smart Things using Wearables," in *Proceedings of the Glass and Eyewear Computers Workshop*, 2014.

[11] S. Mayer, A. Tschofen, A. K. Dey, and F. Mattern, "User Interfaces for Smart Things – A Generative Approach with Semantic Interaction Descriptions," *ACM Transactions on Computer-Human Interaction*, vol. 21, no. 2, 2014.

[12] S. Mayer, N. Inhelder, R. Verborgh, R. V. de Walle, and F. Mattern, "Configuration of Smart Environments Made Simple – Combining Visual Modeling with Semantic Metadata and Reasoning," in *Proceedings of the 4th International Conference on the Internet of Things (IoT)*, 2014.

[13] S. Mayer, Y. N. Hassan, and G. Sörös, "A Magic Lens for Revealing Device Interactions in Smart Environments," in *Proceedings of the SIGGRAPH Asia 2014 Symposium on Mobile Graphics and Interactive Applications (MGIA)*, 2014.

[14] F. Dötzer, "Privacy Issues in Vehicular Ad Hoc Networks," in *Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, G. Danezis and D. Martin, Eds., Springer, 2006, vol. 3856, pp. 197–209.