Industry IoT Consortium®

Technology Innovation. Business Transformation.

The IIC's Automotive and Over-The-Air Update Testbed

The Industry IoT Consortium's Automotive and Over-The-Air (OTA) Update Testbed shows how software can be managed on a fleet of vehicles remotely. Software can be deployed, updated, and replaced. A microservices architecture is used to provide the maximum functional improvement with a minimal of software change.

The OTA Testbed covers standard topics such as vehicle control and data handling, collection, filtering, and analysis. It also addresses the specific automotive safety risks in a cybersecurity realm and introduces Li-Fi as a complementary wireless communication option for such updates as well as general V2X communication. The basic challenges addressed are common among most IoT domains, whether it be medical devices, industrial machines, or vehicles.

The Go-Kart Testbed Demonstrator

The Go-Kart Testbed Demonstrator for the OTA Testbed was created to show how a system needs to be conceived for software over-the-air updates to reach its full potential. In the Demonstration, OTA is not just pushing some software over the network. This is already done to a limited extent, with limited advantages today. Rather, the overall system, with emphasis on the following points, must be considered:

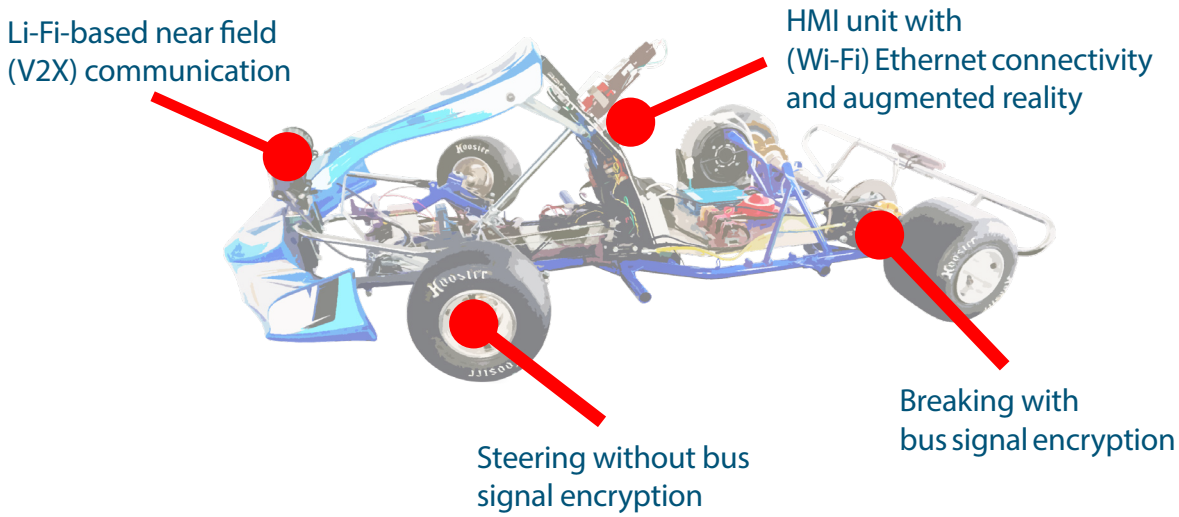
Over-the-Air Testbed Sponsors

- Limiting updates to what is required;
- Understanding what updates are needed, and what is the current software standard for each component;
- Ensuring that only the systems where the update is authorized are updated,
- Reverting any update without exception;
- Being able to update, within reason, a running system;
- Ensuring that only authorized software can be deployed;
- Deploying new software (most) anywhere, anytime; and
- Never compromising system security, safety, integrity, privacy, resilience, or robustness.
- Introducing the use Li-Fi for OTA updates and V2X communication to address the challenges associated with RF such as signal interference/jamming and limited capacity.

From these concerns, it should be clear that the entire system must be designed with OTA in mind.

One must be able to uniquely identify each relevant part of the system and know the current state of the software running on it and how the software can be changed. The safety and security of the passengers must come first.

Go-Kart Demonstrator System Features



The idea is to start with a few features and build it out over time. The closer the Go-Kart is to a fully functional auto, the better the OTA features can be demonstrated. The Testbed is far from complete; Power is not yet connected and there are not any mobile network connections such as GSM or better. The cloud connectivity is currently to an AWS server and provides the following services:

- Data connection for data collection,
- Remote control of steering and breaking, and
- Remote software management on the HMI unit.

Much more needs to be done to update other system elements and integrate the Li-Fi more tightly into the system. Of course, other cloud connectivity, such as Microsoft Azure could easily be incorporated. Messaging is used for data connection, rather than a less secure REST API.

Architecture

The base architecture is built around a Telematics Unit (TU) to manage the connection to the external network. Currently, the TU is also the HMI unit, but this can be separated easily. The TU also has connections to the internal buses: Ethernet and Controller Area Network (CAN).

We could replace this with local CAN buses talking to a domain controller with a realtime Ethernet backbone between domain controllers and the TU and HMI unit. CAN is useful for individual connects to low cost devices, but does not offer much in the way of security. Though even there, the Kart demonstrates some basic encryption over CAN. Ethernet provides much higher bandwidth and security for cameras and other data heavy connectivity.

On the TU, a realtime OSGi framework with resource enforcement is used to manage software. Each domain controller will have an OSGi instance.

Connectivity

Robust and diverse connectivity options are critical to a stable OTA and V2X communication. Granting the ability to switch between commercially available technologies such as 5G, Wi-Fi, and Li-Fi will ensure continuity, efficiency, and safety of the updates and operations.

Software Management

Proper software management is key to successful OTA. This management starts with software development. It should be possible to trace any piece of software back to its original source code and configuration. This means managing the life cycle of code, not only in development, put also into each device where it is used. This goes beyond conventional DevOps as practiced in the IT domain, since a much larger variety of target must be considered.

Security

Only authorized entities should be able to monitor devices. Again, encryption is an important aspect. Though a Rest API with encryption, such as HTTPS, are popular as a remote interface to a device, there are better ways. Message services, such as MQTT and XMPP, have the advantage of not needing to have an open port on each device. No having open ports makes the devices harder to detect and attack. The device simply does not pay any attention to incoming requests. An embedded device does not have the same resources to handle denial-of-service attacks that a server has. Of course, implementation language plays a role as well. Using a managed language can reduce security vulnerabilities considerably. *

Additionally, the use of Li-Fi will offer security measures through the intrinsic proprieties of light-based communication vs RF-based communication.

Unique Identity

One of the necessary pieces is identity management. One must be able to verify each device, each software deployment server, each software artifacts to be deployed (software bundle), and each source of such artifacts or bundles. Cryptographic methods are required here for creating credentials and identifying each player.

Remote Monitoring

Once each device and software bundle can be identified, the next step is to know what software is on each device and what is actually running on each device. This includes the version of each bundle. Version tracking is important for knowing which bundles have been verified as compatible and which bundles might need updating. Because OT systems have long lifetimes and bandwidth is limited, one does not want to update all bundles each time something is changed. Tracking what is running on each device and in what versions is a prerequisite for OTA.

* https://www.zdnet.com/article/chrome-70-of-all-security-bugs-are-memory-safety-issues/
https://www.zdnet.com/article/microsoft-70-percent-of-all-security-bugs-are-memory-safety-issues/

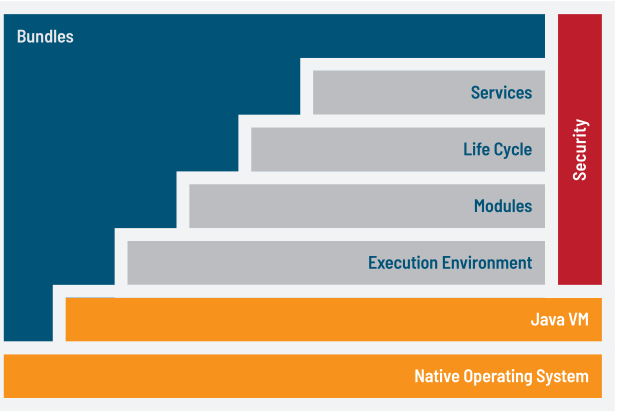
Dynamic Update

Dynamic update is not very dynamic if one must restart the system for each update. The Kart demo addresses this by using a service-oriented architecture. OSGi provides this through a service registry. Each bundle can define a service and register it in the registry. Services required by a bundle can be found through the registry. This enables services to be replaced at run time. Once a new version of a service is started, the clients of the old version can be told to roll over to the new service. Once all have done so, the old service can shut down.

For this to work, each service user must know with which versions of a service it is compatible and each service should have a defined version. OSGi provides a standard infrastructure for managing these versions and dependencies. With careful rollover design, even relatively time critical system can be updated with this mechanism. This is a major reason for choosing OSGi for the management of on board software.

Software Reuse

Managing software in bundles—modular software elements—simplifies software reuse. Only the bundles that require change need be updated. The rest can remain the same. There are a plethora of software tasks that can be implemented as a bundle and reused in a wide variety of systems. A message client is a good example. It can even be reused in the same system, for say remote software management and data transmission. Using virtual machine technology can increase software reused dramatically, by enabling the same code to run on vastly different hardware. With the proper compilation technology, this can be done with very little performance penalty.



A Standard Framework

The Demonstration uses OSGi with Realtime Java and resource enforcement as the on-board gateway between the vehicle and the cloud. OSGi provides standard APIs to the developer and in this configuration offers all the services needed for OTA: identity management, remote monitoring, security, dynamic update, and software reuse. The term bundle for software management comes from OSGi and so does the term microservice. OSGi is the first standard based on a microservice architecture. Version tracking and compatibility tracking are an integral part of the OSGi standard. OSGi features are also used to manage software that runs outside the OSGi framework, such as FPGA, GPGPU programs, on-board SOCs external to the gateway, and containers and executables running alongside the gateway. With OSGi, code can be moved seamlessly between the cloud and the gateway.

Data Collection, Filtering and Analysis

OSGi is already being used for data collection, filtering, and analysis. The realtime extensions enable bundles to reliably capture data that is only available for a short interval. OSGi makes it easy to decide where to analyze what data and to reuse analysis code.

On-Board Filtering and Analysis

On-board filtering and analysis is important for deriving information that the drive needs to be made aware of unconditionally. Whether it be engine malfunction or an imminent collision, the drive (even a virtual one) cannot rely on cloud computation to provide such information. The transport delays are often too longer and connections are not always available. On-board analysis avoids these problems.

Data Visualization and V2X

Combining the Li-Fi-based V2X capabilities will also show how a virtual/augmented reality application could be used to obtain real-time information about the vehicle. This feature will expedite the collection of information when the vehicle goes in for service, or other situation where such information needs to be displayed and analyzed quickly. It could also be expanded to provide other V2X communication features..

Flexible Deployment

Having a flexible deployment scheme means that where analysis takes place can be determined at the application level without incurring addition costs. A highly constrained autonomous system with constant connectivity and slow rates of travel can have much more analysis in the field than a car that is designed to run on the autobahn and in the average city. Never the less, much of the same software could be used in both scenarios. Lower cost on-board systems could be used in the former case, than in the later, where much more analysis must be done locally.

Changing the Game

The purpose of the Go-Kart Demonstration is to come as near as possible to a full automotive Proof of Concept for OTA. This means covering a much of the concerns of modern electrified, autonomous autos as possible. Connectivity adds an entire new layer of requirements to OT systems. Security and updatability are traditionally IT concerns. On the other hand, realtime data collection and response with limited resources goes well beyond what most IT systems are of. Only by combining techniques from both worlds is a successful OTA possible.

About the Industry IoT Consortium® (IIC®) The Industry IoT Consortium delivers transformative business value to industry, organizations, and society by accelerating adoption of a trustworthy internet of things. The Industry IoT Consortium is a program of the Object Management Group® (OMG®). Please visit us on the web at iiconsortium.org.

Get in touch with the Over-the-Air Testbed sponsors:

aicas GmbH
Emmy-Noether-Str. 9
76131 Karlsruhe, Germany

Phone: +49 721 663 968 0
Web: https://www.aicas.com
Email: info@aicas.com

AASA Inc.
12353-Sunrise Valley Drive
Reston, VA 20191 USA

Phone: +1 703 444 6170
Web: https://www.aasainc.com
Email: info@aasainc.com