Altreonic NV
Privately owned SME. BE0899997375
Legal seat: Gemeentestraat 61A bus 1, 3210 Linden, Belgium
Exploitation: Staatsbaan 4A/1, 3210 Lubbeek, Belgium
www.altreonic.com
www.kurt.mobi

VirtuosoNext for e-mobility

Why e-mobility is disruptive and is fully software driven

The automotive industry is currently confronted with major challenges: there is increasingly pressure to deliver clean, mostly defined as vehicles with electric propulsion, and more safety and convenience by introducing autonomously driven vehicles. No longer the vehicles are important, they have become part of larger systems that deliver mobility and transport as a service. At the same time this disruption holds huge opportunities. The need for mobility and transport will not diminish. On the contrary, it will continue increasing but society wants it to be sustainable.
A vehicle in such a context becomes a propulsion platform, delivering a service that can vary over time. At the same time, it creates a new type of virtual network as the vehicles increasingly communicate with the rest of the world and with each other. This evolution is similar to the revolution of digital communications when analog circuit switching was replaced with high speed packet switching. In both cases software is playing an increasingly dominant role.

Software was originally introduced as firmware in specific devices like the engine control unit. It allowed to provide more and cleaner performance. More peripheral functions became software driven (e.g. braking, gearbox, steering) resulting in a drive-by-wire vehicle. Often the devices were designed as plug-in replacements for the mechanical predecessors (also dictated by the Tier-1-2-3 supplier chain) and connected through ad-hoc relatively slow buses (LIN, CAN, FlexRay). The result today is a very high degree of complexity but also of vulnerability.

E-mobility enables a new platform

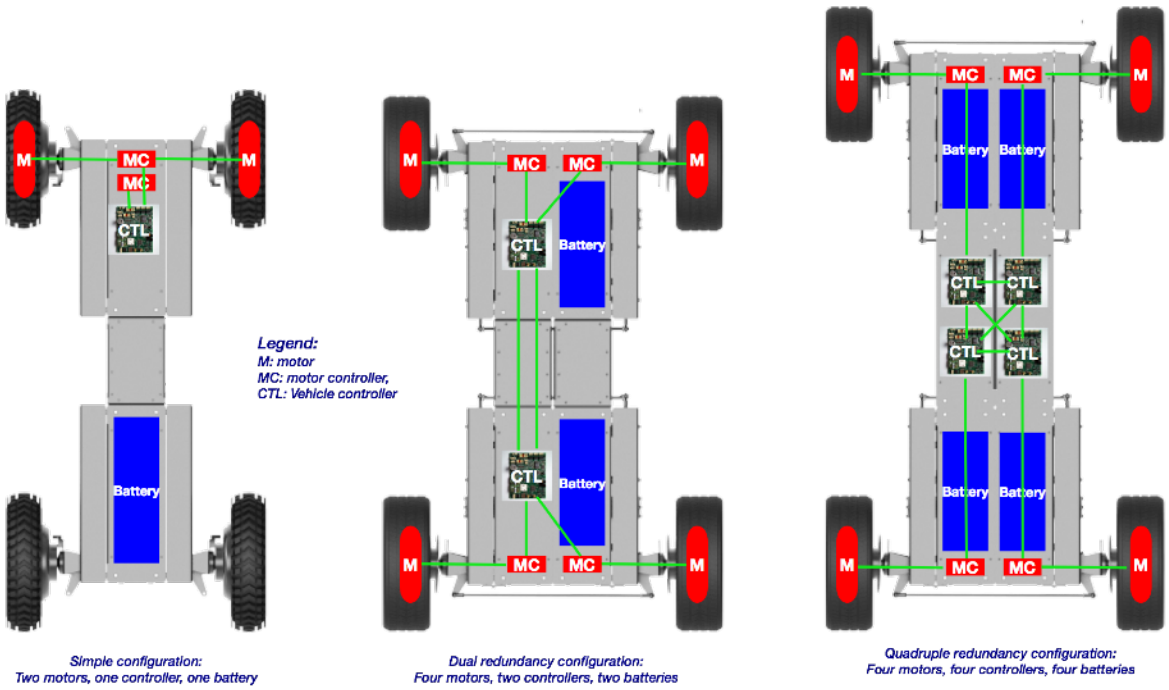
Electric propulsion and the evolution towards autonomous driving requires a new approach to avoid hitting the complexity wall. Autonomous driving will require active redundancy to meet the safety requirements. At the same time this is what electric propulsion enables as each wheel can have its own motor. Autonomous driving also requires high-speed sensors (radar, lidar, camera, IR, …) and high speed processing. The result is that a vehicle is becoming (or at least: will be) a distributed and fault tolerant embedded system driven by a dedicated super computer on wheels. In addition, it communicates with the external world (e.g. to receive instructions or to receive traffic updates), the environment and other vehicles allowing a smoother and safer traffic.

The importance of safety and security

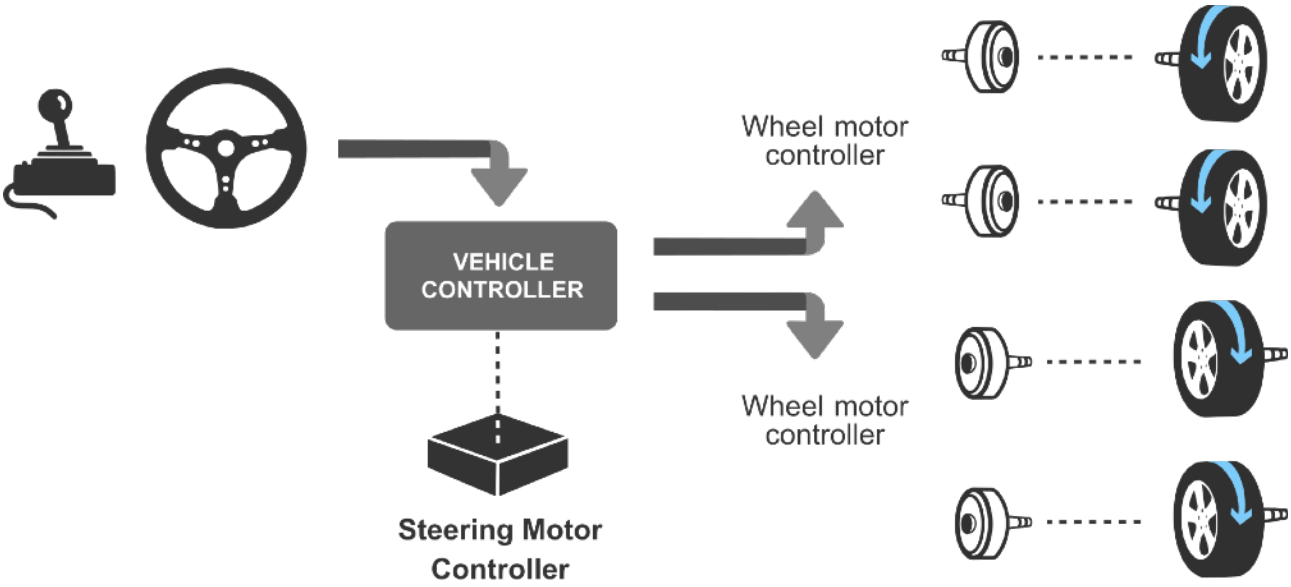
In traditional vehicles safety is provided firstly by an adequate and robust design and secondly by the experience of the driver itself. When vehicles are completely driven by software, the system can fail any moment (in nanoseconds!) without much warning signs. This is inherent to the use of digital electronics. The only way to prevent the system from catastrophic failing is an engineering process that reduces residual errors to an acceptable minimum and a defensive architecture. Ultimately redundancy is needed as well diversity to further reduce the risks due to single points of failure and common mode failures. Note that such architectures are standard practice in e.g. the aviation industry. It should be noted that in automotive recovery requirements are more stringent than in aviation. At typical highway speeds, a system failure needs to be recovered from in about 100 milliseconds.
Above defines the propulsion domain wherein absolute safety and reliability is required. No hardware or software failure should result in the vehicle abruptly coming to a halt or to execute unsafe driving commands. The autonomously driving vehicle adds a second domain whereby high speed sensors and input from the outside world are combined and processed in real-time. The resulting output are steering commands to the propulsion domain. And thirdly, such a vehicle will have a network of communication devices that interact with external nodes as well as with other vehicles. The latter is the least safety critical but dominated by security requirements. An example of a solution that communicates to the outside world is the carputer computer of Link Motion (www.link-motion.com). It uses hardware protection as well as software security protection to keep unwanted security breaches at bay.
While all three domains should never witness security breaches (as these can result in safety risks), these domains should be strictly separated and only communicate using authenticated or even encrypted messages.

The application in the KURT e-mobility platform

The KURT e-mobility concept is based on several complementary architectural views. At the level of the propulsion platform, the unit of propulsion is a single wheel. This unit contains everything to provide propulsion: a motor in the wheel, a motor controller, a steering motor and controller, a vehicle controller and a battery. A propulsion platform is composed by combining 2 or 4 of these units. In the simplest case, just two wheels with motor and a single vehicle controller can be used, the two other wheels being passive. This is sufficient for normal use when a driver is on board.



For vehicles require more safety and security we apply our own ARRL (Assured Reliability and Resilience) criterion. This criterion defines architectures that take into account the fault behavior and its consequences on the continued operation of a mission or safety critical system.
Dual redundancy is obtained by the front and rear wheels section being independently controlled. This is sufficient for urban autonomous shuttles. If one section fails, the vehicle can continue to drive until a safe point is reached. For higher speeds, quadruple redundancy can be applied. A vehicle can then continue to drive with some performance limitations. In both cases, will the software quickly reconfigure the load over the remaining wheel. The passenger might barely notice it.



All of the sections are galvanically isolated to reduce common mode failures but communicate using high speed interconnects (typically ethernet or optical fiber) configured in a redundant mesh. This provides the advantages of reduced wiring, a principle that is also applied to peripheral nodes like sensors and actuators.
In such a configuration steering commands can be of any source, be it from a steering device operated by a human driver or from an autonomous driving computer or a remote operator driving “over the air”. Of course, in the latter case, all security requirements apply.

Why VirtuosoNext meets the challenge

If next generation mobility is driven by software, why not base the whole system concept on it? The system view put forward by VirtuosoNext Designer is one of “Interacting Entities” and is visible in the KURT architecture. Modular units can interconnect and can be scaled up, reducing production cost while increasing flexibility and robustness, even achieving fault tolerance.

VirtuosoNext has its roots in 1991 when it was first demonstrated on a parallel processing network. Later on, it evolved and was most often used in demanding aerospace and defense projects, some of them having 1000’s of processors. In 2001, it was acquired by Wind River Systems (now Intel) and removed from the market a few years later. Today’s VirtuosoNext is a fifth generation RTOS developed anew with formal techniques in 2005. The result was a much cleaner and portable Real-Time Operating System, still fully distributed. The formal techniques (Leslie Lamport’s TLA+/TLC) reduced the code size with a factor 10 (now around 20 to 30 Kbytes/node). The latest implementation provides fine-grain space and time partitioning as well as the capability to recover from runtime faults in a few microseconds. The latter is available on lower end microcontrollers (like ARM-Cortex-M or -R series), high-end multi-core ARM-Cortex-A and PowerPC architectures as well as on high-end multicore DSPs like TI-6678). It is hence an ideal RTOS for the future propulsion systems of autonomous cars.

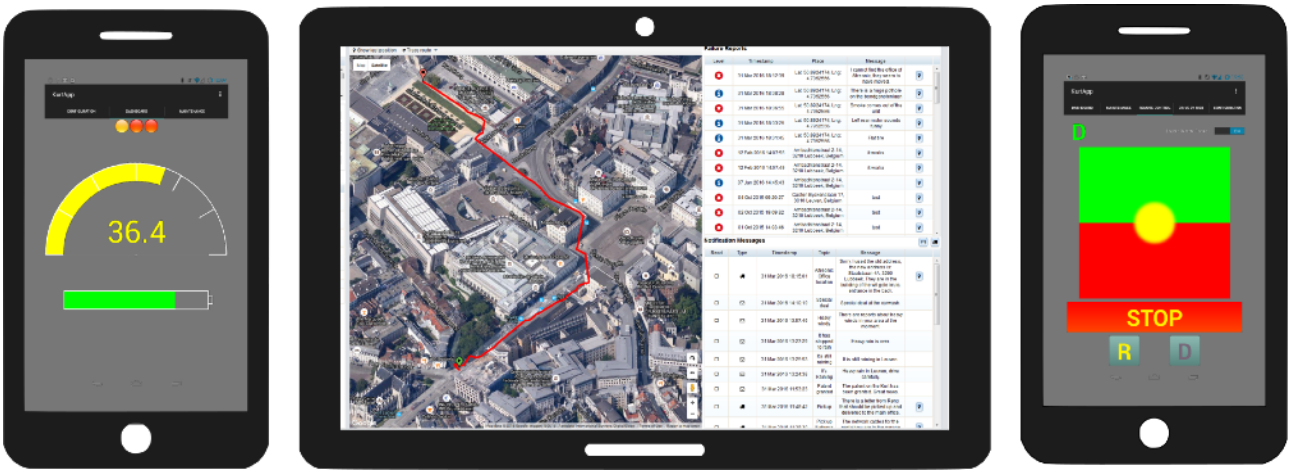
The core concept of the VirtuosoNext RTOS kernel is packet switching. Packets carry all commands (service requests) and data. This is similar as found in large digital communication

systems and the internet, except that VirtuosoNext uses a light weight implementation whereby the packets are prioritised across the network. This core concept allows transparent programming of large distributed, even when heterogenous, and independent of the communication media used. In the context of a vehicle, this means small smart sensor nodes are programmed the same way as high-end multicore processors, e.g. used for dynamic control of high speed sensor processing.

While the core component is a small RTOS kernel, most of the intelligence resides in the supporting development tools. These generate most of the code from higher level visual modelling diagrams and platform specific metamodels. This considerably reduces development times. The generated code itself is a static image, resulting in small memory requirements, low latencies and high performance. The approach also supports safety requirements as dynamic memory access is largely eliminated by design. Runtime faults are trapped and the task's state is recovered without affecting the rest of the application. The static code also largely reduces security breaches as dynamic execution of injected code is practically impossible.

The small code has also other benefits. First of all, it reduces the qualification effort. It also reduces the residual risks (as no software is ever perfect). Another benefit is performance as more code can make use of on-chip caches which then also reduces power consumption.
The application developer benefits from a clean but versatile set of services with well defined semantics. This makes it easy to present an open API (target independent) or to reuse existing code by using an adaptation layer.
VirtuosoNext is also available with support for Ada, a Safe Virtual Machine and a Qualification Package. The software itself is licensed under an Open Technology Licensing model.

A small scale demonstrator



The principle was first implemented in a small vehicle using skid steering. Next an R&D small vehicle was developed using a segmented vehicle (with a hinge between the front and rear section) using 4 motors and driven by wire. An ARM-M3 was used as controller. This vehicle was connected over bluetooth with an Android smartphone or tablet using a proprietary protocol. The smartphone can act as the vehicle console displaying e.g. speed and battery status. In a next step, the smartphone was configured to act as a virtual joystick allowing to steer and control the vehicle without a physical driver. The smartphone was also configured as a gateway to a server application in the cloud allowing to monitor a fleet of vehicles and to establish bi-directional communication with the vehicle. This set-up was expanded in a proof-of-concept demonstration whereby the vehicle was controlled from an internet browser 3000 km away (Portugal to Belgium) and whereby the smartphone camera was used as input to the remote operator. This worked well with a delay of about 100 milliseconds over 2 home-wifi connections and 3G roaming.
The set-up is also used in specific vehicle for use in hospitals, controlled by a joystick.

Annex:

VirtuosoNext Designer datasheet. Visual Designer VirtuosoNext datasheet