Advantages of Utilizing the OMNEST Simulation Environment in Automotive Research, Testing and Verification

AMAA Berlin July 23 2014

Andras Ferencz (Test automation and real-time testing consultant)



From Ideas to Solutions With Confidence

The following challenges can be addressed using OMNEST

- Rapidly increasing system complexity
- Heterogeneous networks and a variety of coding environments in the development
- Verify Control stability, robustness, functionality, determininsm.
- Reduce time to market and costs.



Challenges for the platform/system integrator (OEM)

- OEMs need a process where requirements can be specified and verified on the system level.
- Requirements must be defined at the component level, which can be assigned to individual suppliers.
- Cars need to be integrate with cities, people, other vehicles etc.



Challenges for the component or module developer (supplier)

- Representative stimulus signals should be developed for the component testing
- A good understanding is required about other components affecting the behavior of the component under development
- Messaging stimulus is hard to develop and maintain with parameter and design changes
- Using more and more wireless sensors



What is OMNEST?

A generic simulation framework:

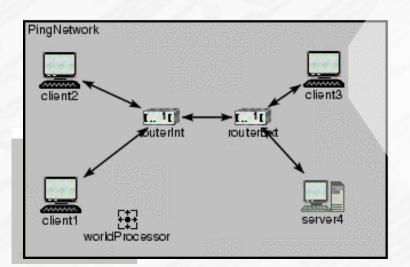
- For the simulation of **complex distributed systems**:
 - distributed hardware and software architectures,
 - communication networks,
 - queuing networks,...
- Technically: a C++-based simulation kernel plus a set of libraries and tools (GUI and command-line)
- An open environment
 - in terms of source code, embedding, extensibility, integration, modularity

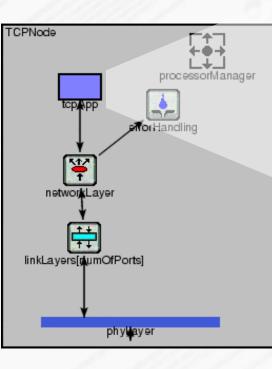


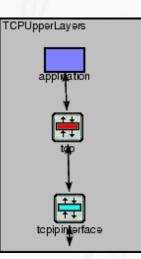
Model Structure

Component-oriented approach:

- The basic building block is a **module**.
- Simple modules can be grouped to form **compound modules**.
- Modules are **connected** with each other.







Defining the Behaviour

Behaviour is encapsulated in **simple modules**. A simple module:

- sends messages,
- reacts to received messages
- collects statistics

Simple modules are programmed in C++.

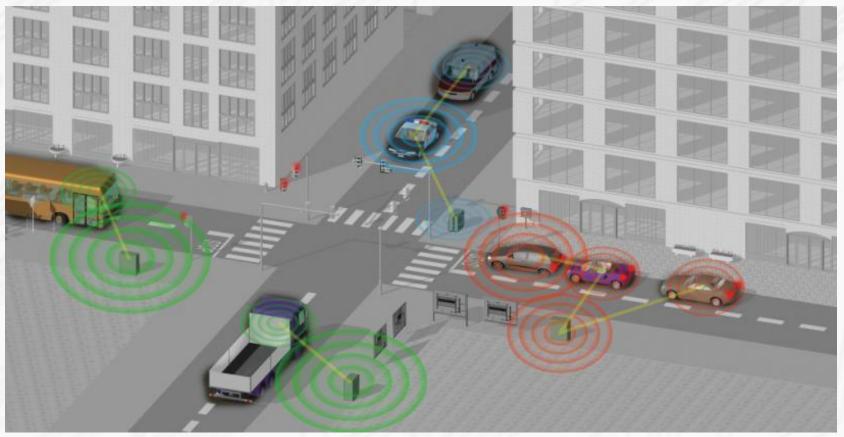


Simulation Models Available for Several Domains

- **Communication network protocols**: TCP, IPv4/IPv6, Ethernet, VoIP, WiFi, ad-hoc wireless networks...
- Automotive protocols: CAN, LIN, DC-BUS, FlexRay, IEEE 802.1 AVB
- Wired and wireless sensor networks
- Support for Hardware-in-the-Loop simulations

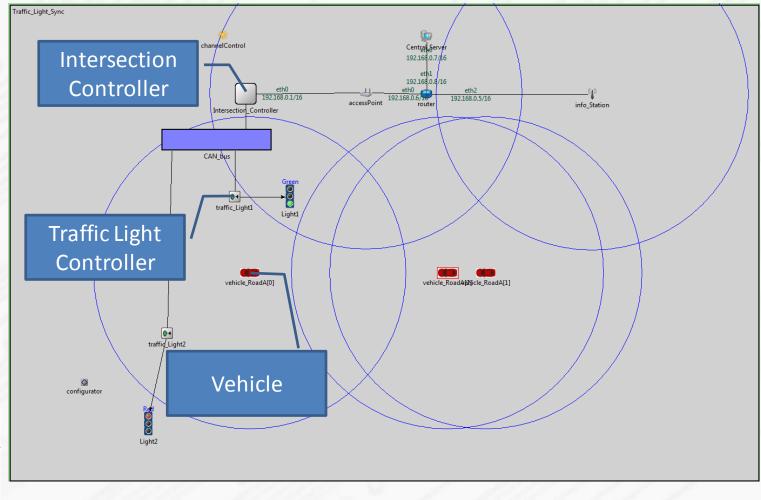


Scenarios



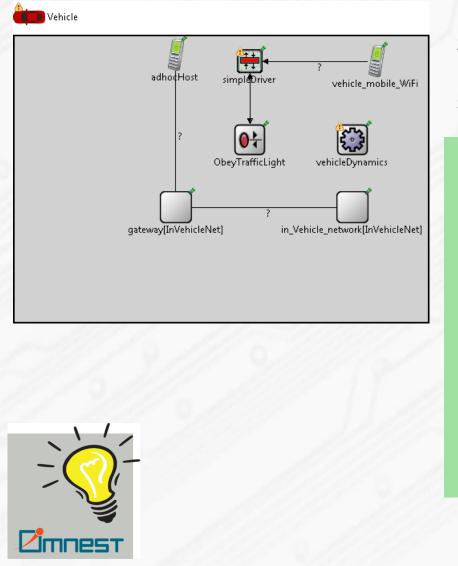


Example application using both in- and inter-vehicle communication

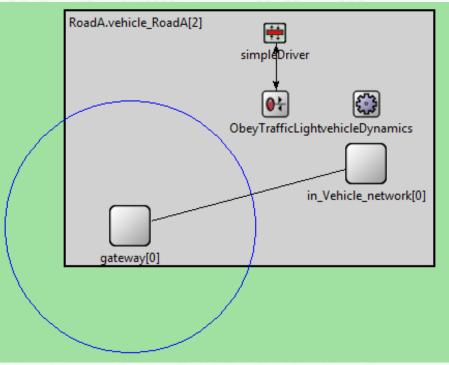


Vehicle Model

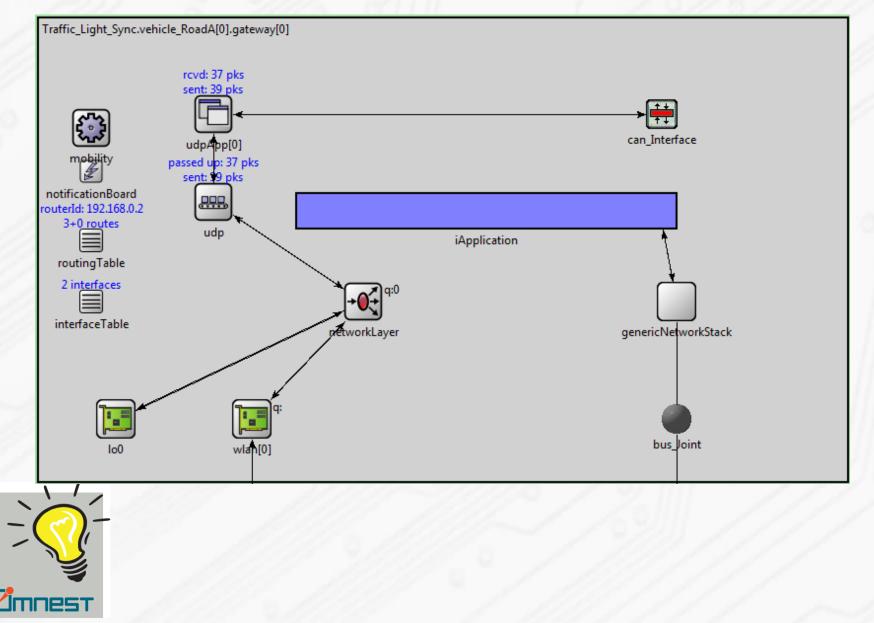
General Vehicle Model



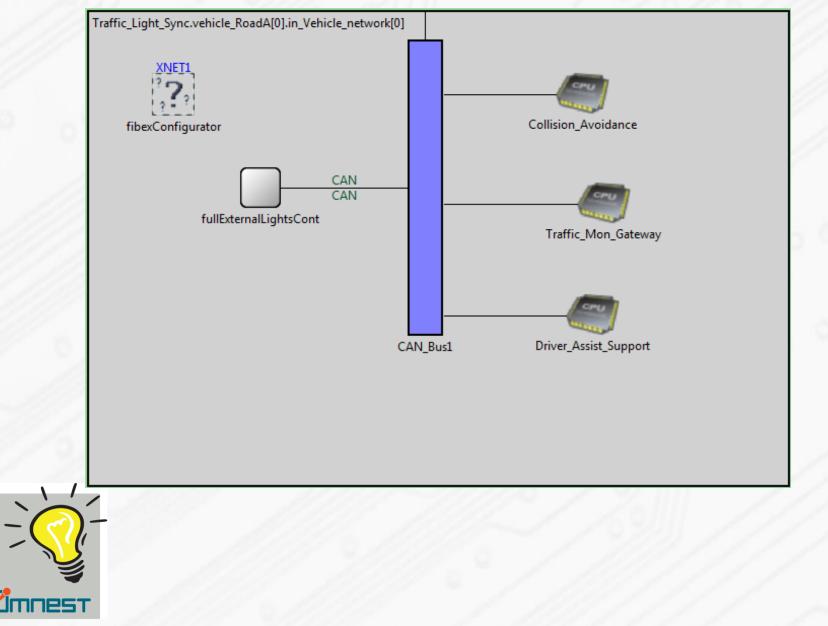
Specific configuration of the Vehicle Model in one of the cars during the simulation



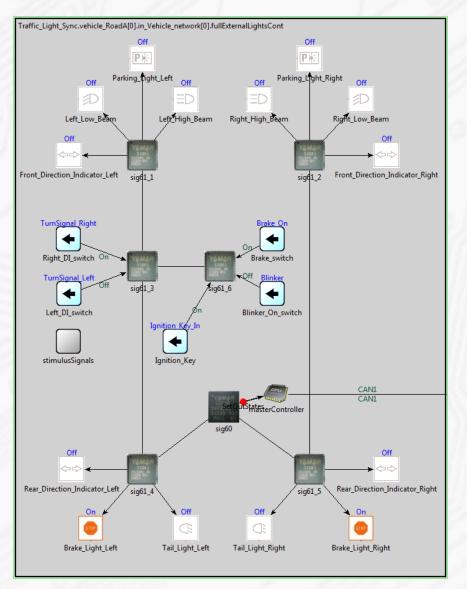
Gateway Model



In-Vehicle Network Model

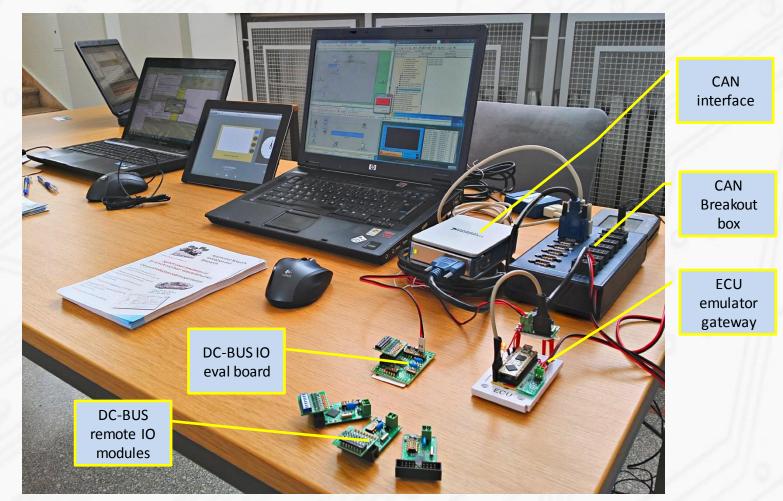


Full External Lights Control model





Demonstration Setup







Use inheritance

Network/systeminheritance

- RoadA (network)
 - a 🔚 Intersection (network)
 - Protected_Intersection (network)
 - a 🔚 Traffic_Light_Sync (network)
 - MasterControllerTestSuite (network)
 - ControllerBoxTestSuite (network)

Module inheritance

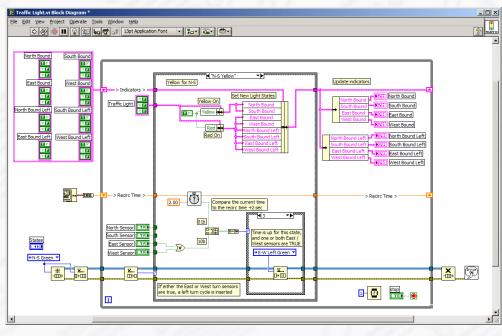
- a 🔚 NodeBase (compound module)
 - a 🔚 inetxnetGatewayBase (compound module)
 - a 🔚 GeninetxnetGateway (compound module)
 - InVehicleGateway (compound module)
 - TrafficLightControlerBox (compound module)
 - Traffic_Light_cont (compound module)
- Initial parameter inheritance (Configuration)
 - RoadA (network)
 - a 🔚 Intersection (network)
 - Protected_Intersection (network)
 - a 🔚 Traffic_Light_Sync (network)
 - MasterControllerTestSuite (network)
 - ControllerBoxTestSuite (network)

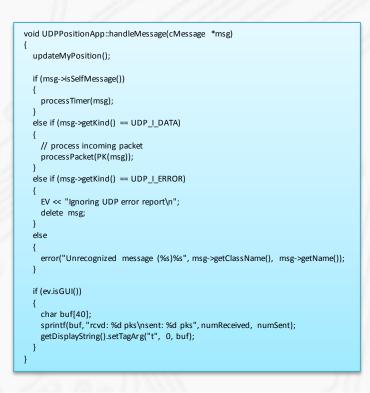
- a 🔚 Traffic_Light_Sync
 - VehicleNum = 1 (NED default applied implicitly)
 - Configurator : IPv4NetworkConfigurator
 - 🔅 🌞 channelControl : ChannelControl
 - vehicle_RoadA[VehicleNum]: Vehicle
 - vehicle_RoadB[VehicleNum] : Vehicle
 Intersection_Controller : IinetxnetGatewayBase
 - b Itraffic_Light1 : Traffic_Light
 - Itraffic_Light2 : Traffic_Light
 - CAN_bus: Bus
 - Light1 : Simple_Traffic_Light
 - > Light2 : Simple_Traffic_Light
 - > 🔯 Central_Server : StandardHost
 - > 😬 router : Router
 - 🛛 😃 accessPoint : AccessPoint
 - I info_Station : Info_Station
 - Intersection_Controller.ethg[*].channel : Eth100M
 - Intersection_Controller.communication[*].channel : IdealChannel
 - Iraffic_Light1.SetLight.channel: DelayChannel
 - Iraffic_Light2.SetLight.channel: DelayChannel
 - CAN_bus.port[*].channel : IdealChannel
 - CAN_bus.port[*].channel : IdealChannel
 - Central_Server.ethg[*].channel : Eth100M
 - router.ethg[*].channel : Eth100M
 - AccessPoint.ethg[*].channel : Eth100M

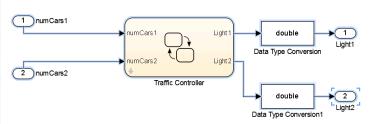


Reuse existing code and custom libraries

Application layer control code can be integrated for testing functionality









Reuse existing models

Models written in:

- C/C++
- Simulink
- SystemC
- NI Model Interface Toolkit compatible models

Continuously growing scientific and student community is developing models for OMNEST/OMNeT++

- Over 10,000 academic installations
- About 300 publications each year, growing steadily in number (Google Scholar data)
- Open source models



Network Analysis (network architect)

- Jitter, latency analysis
- Signal propagation time check
- Bandwidth analysis
- Event diagrams

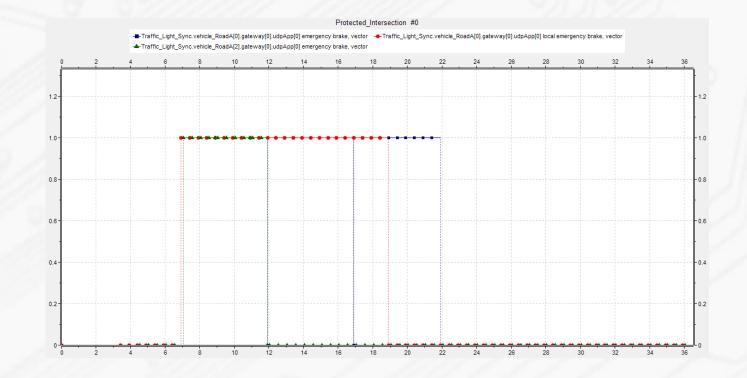


19s 360ms	+572us 925ns	+572us 935ns	+800us	+1ms 500(+1ms 562us 524ns	+1ms 562us 530ns	+1ms 600us +1ms 930us	+2ms 395us 863ns 331ps	+2ms 420us	+2
Traffic Light Sync.vehicle RoadA[1].vehicleDynamics							Position: 19s 360ms 572us 921ns 665ps		
Traffic_Light	Sync.vehicle_Road/	A[1].ObeyTrafficLight					Range: 1	Lms 927us 78ns 333ps	
Traffic Light	Sync.vehicle Road/	A[1].in_Vehicle_network	(0);fullExternalLight	sCont.sig61 1					
		A[1].in_Vehicte_netil526							
Traffic_Light_	Sync.ve#1d26R6ad/	A[1].in Viethiole detwork	(0).fullExternal(Light	scont.sig61_3	4 #1	2632			
Traffic_Light_	Sync vehicle Road	52506_Vehicle_network	(0).fullExternalLight	sCont.sig60					
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	(0) fullExternalLight	sCont.Front_Direction_Indicator_Left 📌	#152626				7
Traffic_Light_	Syncorehicle_Road/	A[1].in_Vehicle_network	(0) fullExternalLight	sCont.Left_Low_Beam					
Traffic_Light_	Sync vehicle_Road	A[1].in_Vehicle_network	(0) fullExternalLight	BCopt/Left_High_Beam	ReadCMDAnsw				
Traffic_Light	SyncheelmitetetRoad	A[1],in_Vehicle_network	c[0].fullExternalLight	sCont.Right_High_Beam	MDUngu				
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	<[0].fullExternalLight	sCont_Right_Low_Beam	ReadCM	DAnsw			
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	<[0].fullExternalLight	sCont.Front_Direction_Indicator_Right					
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	(0) if editest etmails ight	sCont.sig61_4					
Traffie 152600	Sync.vehicle_Road/	A[1].in_Vehicle_network	c[0] fullExternalLight	sCont.sig61_5 #152622					
Traffic_L#3H502	Minc.vehicle_Road/	A[1].in_Vehicle_network	c[0];fullExternalLight	sCont.Rear_Direction_Indicator_Left	#152629				
Traffic_Light_	Sync.vehicle_Road/	A[1],in_Vehicle_network	c[0].fullExternalLight	sCont.Rear_Direction_InditatortRight					
Traffic_Light_	Sync.vehicle_Road/	A[1] in_Vehicle_network	c[0].fullExternalLight	sCont.Brake_Light_Left		ReadCMDAnsw			
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	c[0].fullExternalLight	sCont.Brake_Light_Right		X			
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	c[0].fullExternalLight	sCont.Tail_Light_Left					
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	<[0] fullExternalLight	sCont Tail_Light_Right					
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	c[0];fullExternalLight	sCont.sig61_6					
Traffic_Light_	Sync.vehicle_Road/	A[1].in_ ##5264_I network	<[0].fullExternalLight	sCont.Parki# <u>b52fdfli</u> t_Left					
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	<[0].fullExternalLight	sCont.Parking_Light_Right					
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	<[0].fullExternalLight	sCont.masterController.uart			D. ICHIDA		
Traffic_Light_	Syhc.vehicle_Road/	A[1].in_Vehicle_network	k[0].fullExternalLight	sCont.masterController.serialMaster			≢152635		
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	k[0]:fullExternalLight	sCont.masterController.lightControlAp	p l		#152638 CMDAN	SW	1
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	k[0];fullExternalLight	sCont.masterController.SateFlow_Logic			≆152641	npus	
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	k[0];fullExternalLight	sCont.masterController.Assertions				#152644	
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	<[0].fullExternalLight	sCont.masterController.model_Assertio	ns				
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	k[0].fullExternalLight	sCont.stimulusSignals.StimulusProfiles	Dist				
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	k[0].fullExternalLight	sCont.stimulusSignals.StimulusProfile					
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	k[0]:fullExternalLight	sCont.stimulusSignals.Assertions					
Traffic_Light_	Sync.vehicle_Road/	A[1].in_Vehicle_network	k[0];fullExternalLight	sCont.stimulusSignals.model_Assertion	s				1
Traffic_Light_	Sync.vehicle_Road/	A[1].gateway[0].notifica	ationBoard						
Traffic_Light_	Sync.vehicle_Road/	A[1].gateway[0].routing	Table						
Traffic_Light_	Sync.vehicle_Road/	A[1].gateway[0].interfac	ceTable						
Traffic_Light_	Sync.vehicle Road/	A[1].gateway[0].udp							1
i0us	+572us 925ns	+ 572us 935ns	+800us	+1ms 500(+1ms 562us 524ns	+1ms 562us 530ns	+1ms 600us +1ms 930us	+ 2ms 395us 863ns 331ps	+2ms 420us	+2



Full virtual testing takes advantage of the high performance simulation kernel

- 500k-1000k events/sec
- Complex systems run at 5-10x real time.
- Scalable by using parallel simulation
- Check signal behavior and timing





Reuse models in component testing and system integration for **Rest-Bus Simulation**

- Use real hardware together with prototype hardware or connect an existing production component to the rest of the system
- External Ethernet adapter
- External WiFi
- NI-XNET CAN, LIN, FlexRay

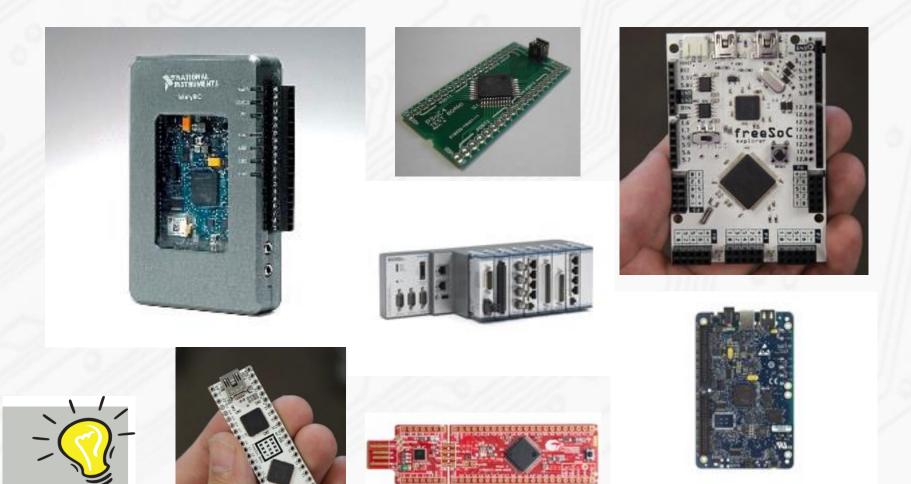




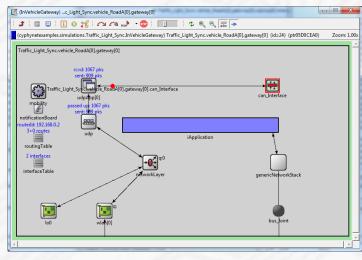




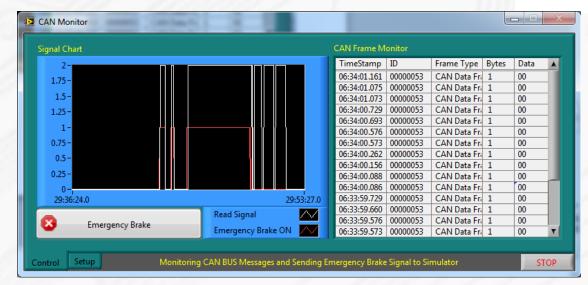
Use COTS hardware to quickly prototype the new component



Capture and visualize bus traffic with your favorite tools



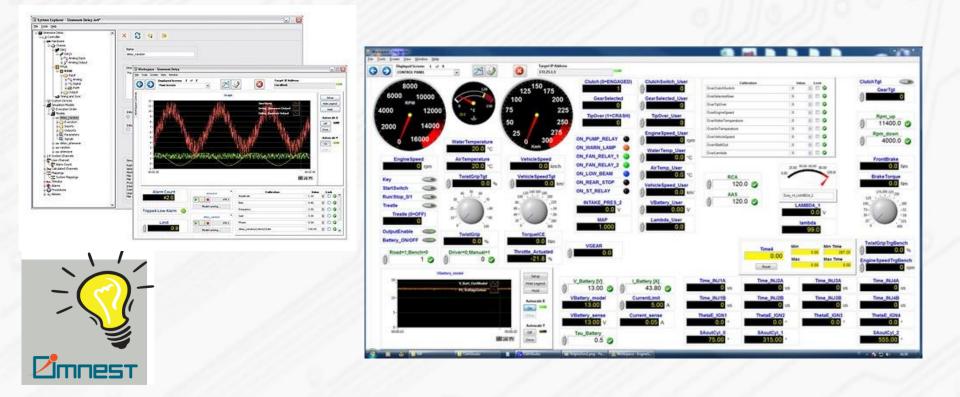






Reuse models and simulation components in hardware validation testing

- Send the components to a Real-Time hardware testing environment such as NI-VeriStand
- Network configuration and connections can be exported to an NI-VeriStand System Definition file



Thanks you for your attention!

Come to the table, ask us questions, and visit <u>www.omnest.com</u>!



From Ideas to Solutions With Confidence