



SERVOTECH

Award Winning Engineering Services

www.ServotechInc.com

**YOUR INNOVATION DREAMS
POWERED BY
EMBEDDED SOFTWARE**



**MECHATRONICS, EMBEDDED CONTROL SOFTWARE, SYSTEM DESIGN,
ROBOTICS, EV POWERTRAIN, BATTERY MANAGEMENT SYSTEMS,
ELECTROHYDRAULIC MOTION CONTROL SYSTEMS, REMOTE CONTROL, AUTOMATION,
AUTONOMY, PERCEPTION SYSTEMS, HIL/SIL/MIL TESTING, INFORMATION
TECHNOLOGY AND BIG DATA AREAS CAD, FINITE ELEMENT ANALYSIS**

Software Tools



Email: info@ServotechInc.com | **Phone:** 312-376-8103 | www.ServotechInc.com





SERVOTECH

Award Winning Engineering Services



... WITH TEAMWORK APPROACH, WE CONTINUOUSLY TRAIN YOUNGER ENGINEERS BY OUR TEAM OF SENIOR ENGINEERS. WE BELIEVE IN LIFE LONG LEARNING...

RECRUITMENT

We recruit top talent around the globe. We provide a work environment that is based on habit of excellence, integrity and mutual respect for everyone. All of our business relationships are based on a "win-win" principle for everyone involved. We aim to be the best global technology services company in the world. Top talent is what we look for in our recruitment. The candidates do not necessarily have to be experts in our field. As long as the candidates have the intellectual talent and moral values, we will teach them.

Our recruitment philosophy is based on what a famous college basketball coach said: "We are looking for great athletes. We will teach them how to dribble the ball". Likewise, we are looking for smartest engineers, we will teach them the details of our technology.

TRAINING

With teamwork approach, we continuously train younger engineers by our team of senior engineers. We believe in life long learning. Everyday, we get some work done, try to learn something new and better, and have a little fun while doing all that.

If you think you are one of the best talents in the world in your field, and are interested in working in an environment where excellence and integrity are the core values, we want you!

OUR OFFICES

Servotech Inc.

329 W. 18th. St. #301,
Chicago IL, 60616 USA

Servotech Global

2nd cross, 2nd Main Rd, Stage 1,
Kengeri Sattelite Town, Bengaluru,
Karnataka 560070, India

Servotech A.S.

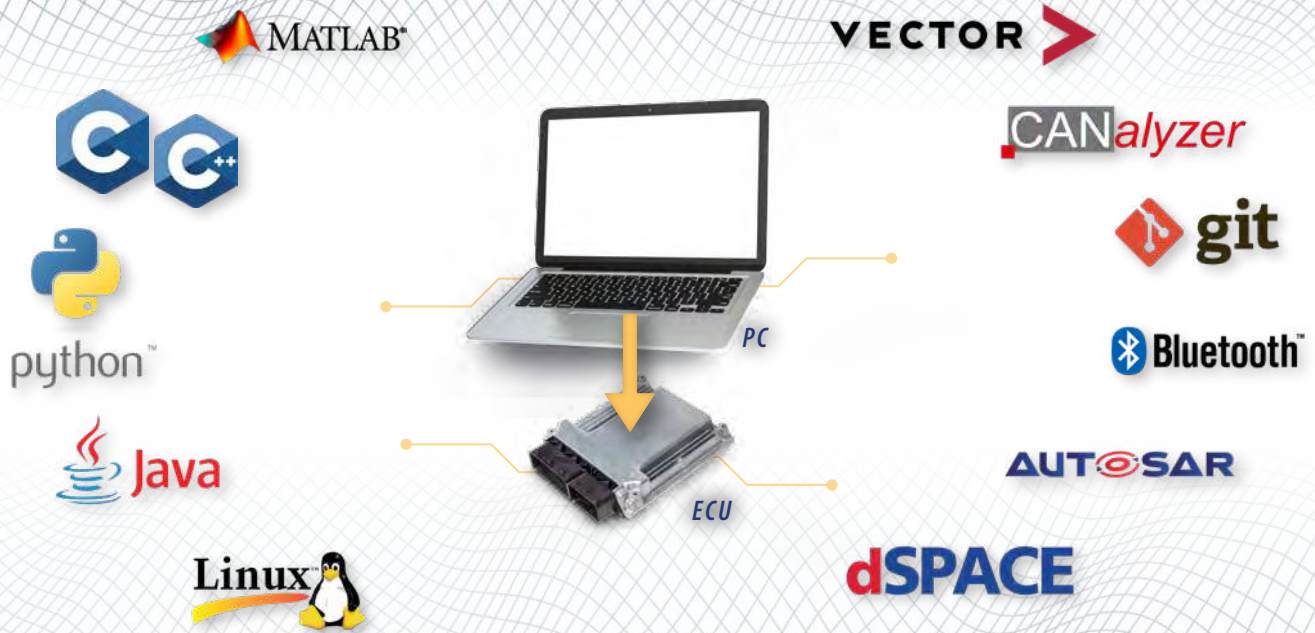
ITU Arikent, Suite 510
Istanbul, Turkey





The software development and testing process has the following steps:

- Develop the control algorithm in Matlab/Simulink Environment. For simulation purposes, develop a “plant” model.
- Matlab/Simulink code can be further extended with C/C++ functions using the S-functions interface.
- Simulate, debug, test, and evaluate the control system in the non real-time, Laptop’s Matlab/Simulink environment.
- Auto-code the control algorithm for the Target ECU: generates C code.
- Build the C code for the target ECU and generate executable code: using the C Compiler tool.
- Download/Flash the executable code to ECU: using the Calibration and Tuning Software tool.
- Run the code on the ECU, tune, calibrate, test, and validate it while running on the ECU, using the Calibration and Tuning Software on the Laptop PC.



Embedded software is the vital component that is the “brains” in all of the devices used in our civilization, from microwave ovens, cars, tractors to airplanes. Embedded software is the program that runs on the embedded controller hardware, referred to as ECU (electronic control unit) or ECM (electronic control module), which is a printed circuit board (PCB) in one package that includes a microcontroller, IO interface, and often power amplifier circuits. The exact content and shape of the ECU vary depending on the application market, i.e. ECUs used in vehicle applications are different than the ECUs used on microwave ovens or medical equipment. The smarter the functionalities of the device, the more sophisticated the embedded software needs to be.

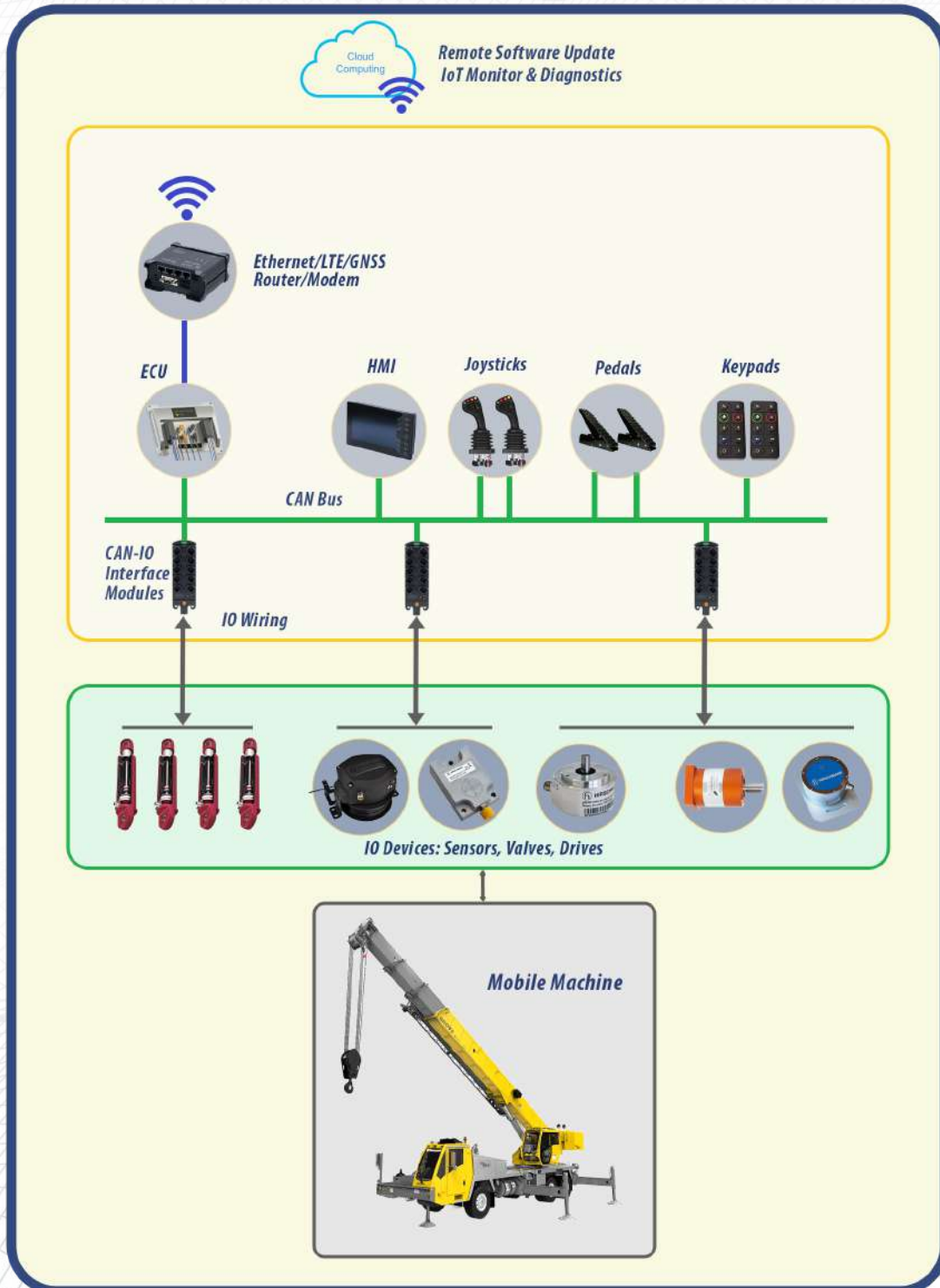
As systems become more and more complex, so do the complexity of embedded software. For instance, the embedded software on a microwave oven is much simpler than the embedded software on a self-driving car. Equally important aspect of embedded software development is its testing and validation; how do we make sure it is reliable. Extensive simulation testing technologies are utilized in testing and validation, such as model in the loop (MIL), software in the loop (SIL), hardware in the loop (HIL) testing. Depending on what is at stake (lives or inconvenience), the extent to which embedded software is tested and validated varies in different applications.

Embedded Software versus Non-Embedded Software

Embedded software is a type of computer software that is “embedded” in a device that we use. The differences between “embedded software” and general purpose “software” are as follows:

- Embedded software deals with real world IO in real-time. A few seconds or even milliseconds of delay can have serious consequences, including loss of life, in some embedded control applications. Whereas such delays in general software may simply result in inconvenience to the user.
- Embedded software is the “brain” of a device. The smarter the “brain” is, the smarter the device is.
- Embedded software reliability is paramount importance because lives may be riding on it, hence must be tested extensively using exhaustive testing techniques using MIL/SIL/HIL technologies. Non-embedded software generally does not have this strict requirement on reliability.
- As the markets demand more and more sophisticated devices, with more reliability and lower cost, the embedded software development task necessarily becomes more sophisticated, must be reliable, and development cost must be low.
- Security is a fundamental concern for embedded software and embedded computers, since virtually everything will shortly have IIoT connectivity, opening a box for many opportunities as well as security vulnerabilities.

Mobile Machine Control System Based on CAN Bus





SERVOTECH

Award Winning Engineering Services

Hardware

- *ECU*
- *HMI*
- *Joysticks*
- *Pedals*
- *Keypads*
- *IO Interface Modules*
- *Ethernet LTE GNSS Modem*

Cables

Power and IO Connector Cables

CAN Bus Cables

Ethernet Cables

Programming USB/Ethernet Module Cables

Software

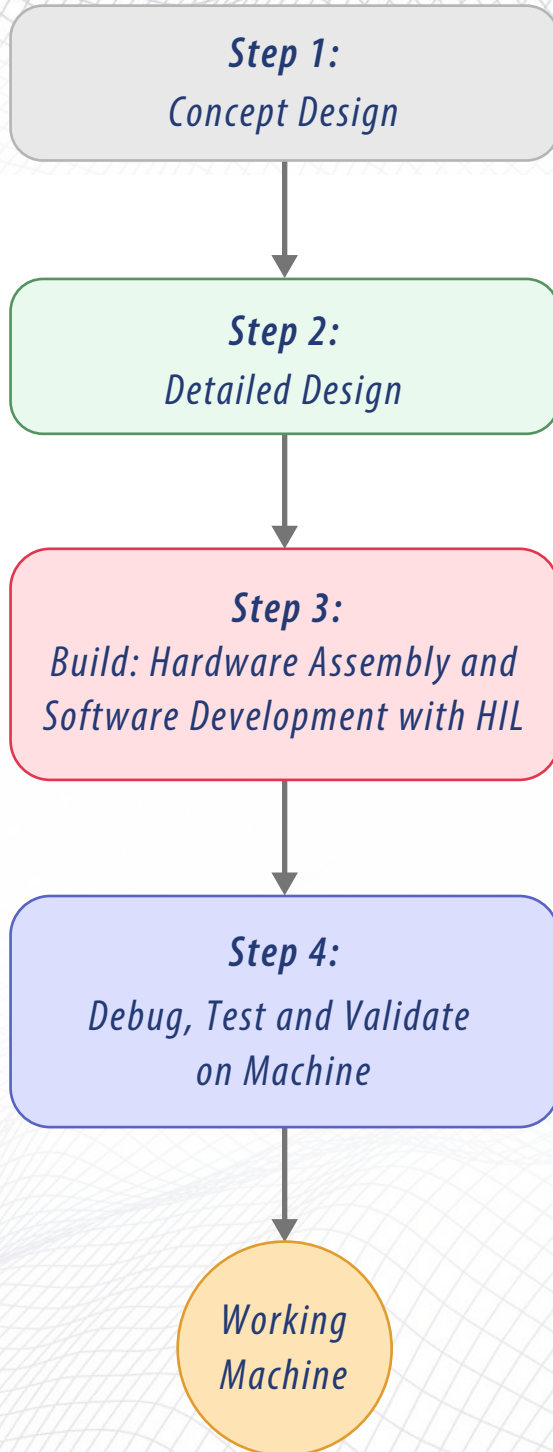
- *CODESYS Vxx SPyy Patch zz for each Master Device*
- *CODESYS Add-on Device Support Libraries for Devices*
- *CAN Bus Slave Nodes: EDS Files*
- *Application Software for the Machine*

Applications

Mobile Machine Market: Tractors, Lift Trucks, Cranes, Dozers, Backhoe Loaders, Telescoping Service Trucks, Excavators, Wheel Loaders, Planting Machines, Sprayers, Harvesters,...



Engineering Process



*Concept Design of the Proposed Control System
Estimated Project Cost
Detailed Design Cost*

*Bill of Materials
Electrical Wiring Diagrams
Hydraulic Circuit Diagrams
Application Software Requirements & Architecture
Updated Project Cost*

*Procurement of Bill of Materials
Electrical Wiring of Control System Components
Application Software Developed and Tested in
HIL Environment*

*Working Control System
Documentation:*

- *Bill of Materials*
- *Wiring Diagrams*
- *Application Software*

Maintenance Plan and Service Agreement



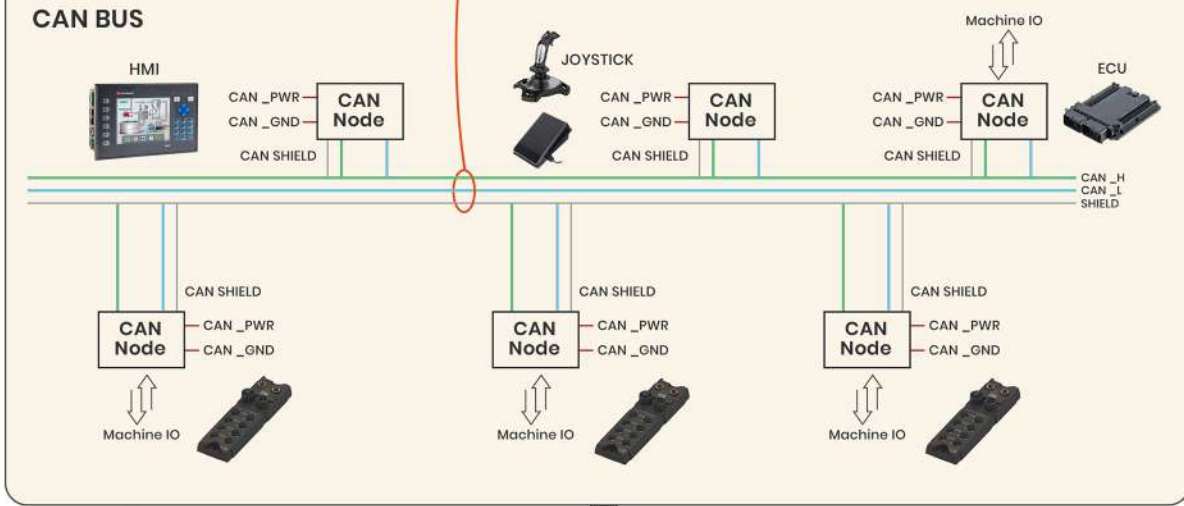
CAN Bus Testing, Diagnostics and Validation Engineering



Laptop Software: for CAN Bus Interface, Data Recording, Analysis, CANalyzer, CanKing, CODESYS, Matlab/Simulink, Python, C++



CAN BUS





CAN Bus Testing, Diagnostics and Validation Engineering

Markets

Automotive, Construction, Agriculture, Mining, Marine, Warehouse Equipment.

Tools

Laptop PC, USB/CAN module, Software for CAN bus interface and data analysis; CANalyzer, CODESYS, Matlab/Simulink, C/C++, Python.

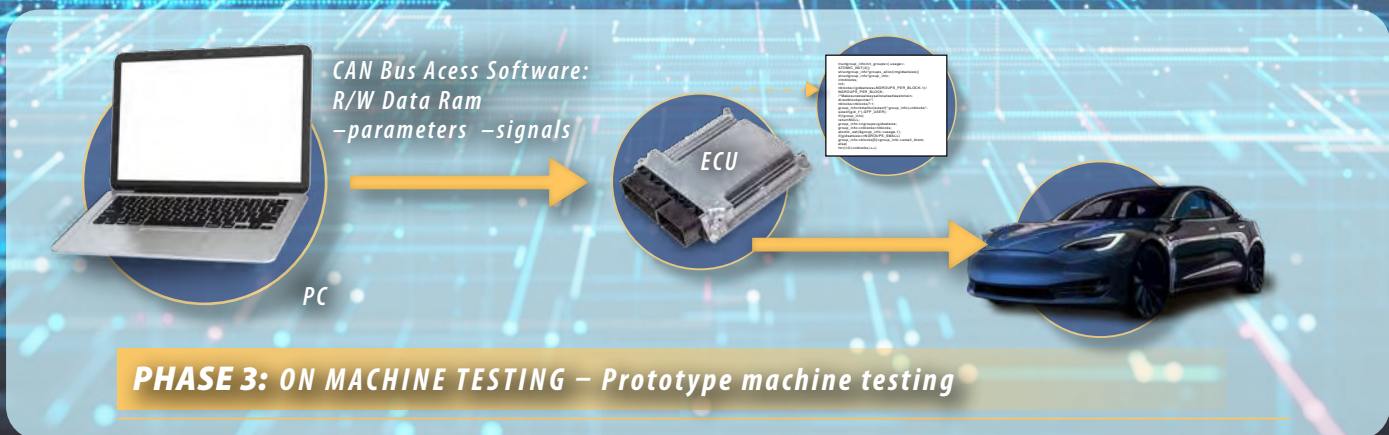
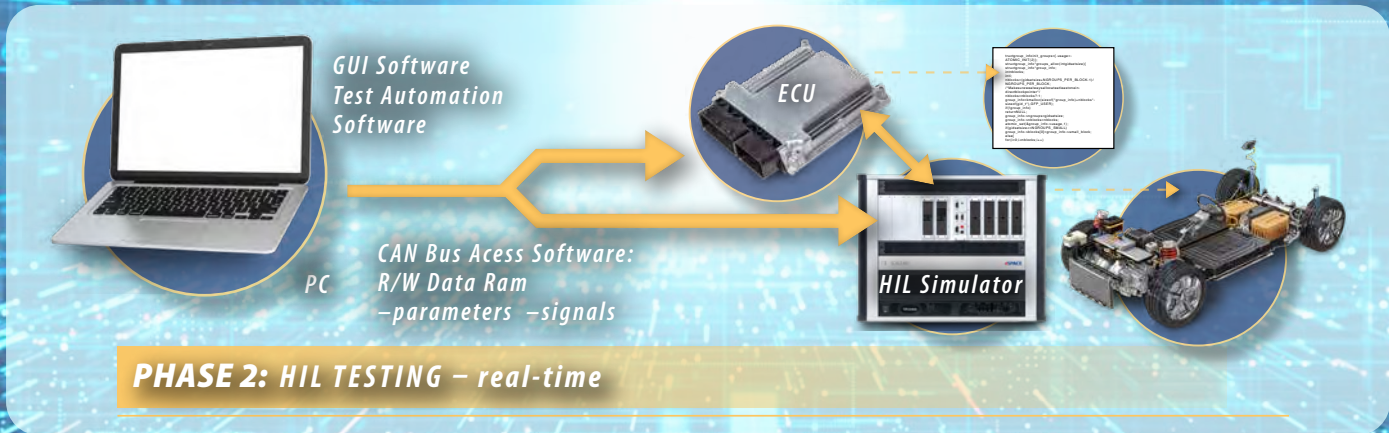
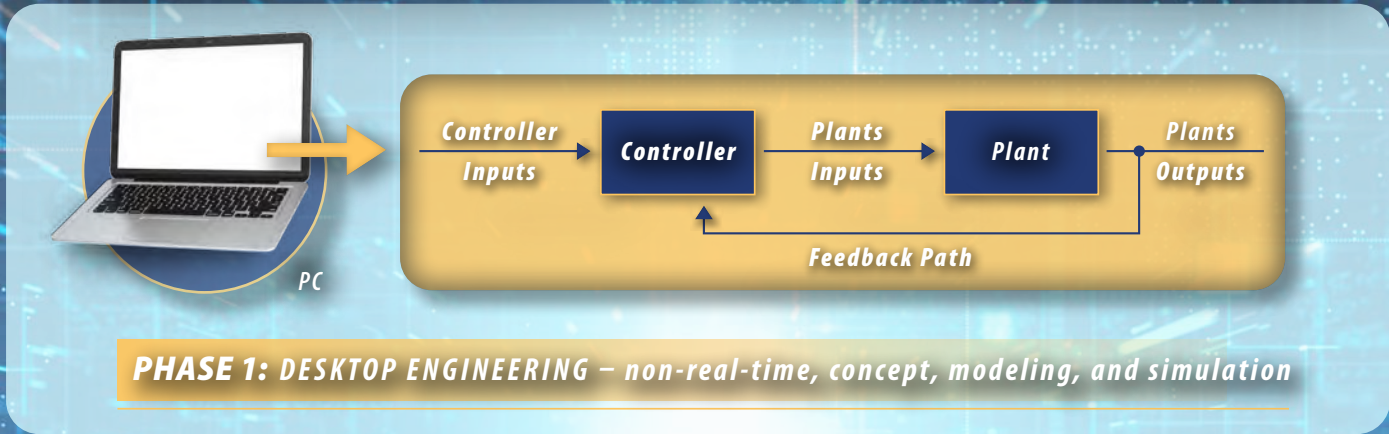
Expertise

CAN bus technology, embedded software; development, testing, diagnostics, debug and validation.

Services

- Design, Test, Diagnostics and Validation of a vehicle control system design based on CAN bus, using various protocols including J1939, CANopen, proprietary.
- CAN bus hardware design, Master and Slave node selections, configuration, programming.
- Test, diagnose, debug and validate CAN bus based operation.
- Strees-testing of existing systems for reliability.
- On-machine and remote data collection and monitoring.
- Application software/firmware upgrade.

An embedded control software is developed in three main phases.





An Embedded controller is a rugged computer hardware , which may include power amplification circuit, with an application-specific software. The embedded controller hardware, also called Electronic Control Unit (ECU) or Electronic Control Module (ECM), is rather standard provided by various suppliers to Original Equipment Manufacturers (OEM). The ECU software, however, is always application-specific and custom developed for each application by an OEM.

PHASE 1–

In Phase 1 of the development process, the control software for the electronic control unit (ECU) is created using software tools like Matlab, Simulink and Stateflow. The software is simulated and analyzed on a non-real-time desktop environment. A detailed dynamic model of the machine called the "plant model," is used for accurate simulations. The software is developed in different layers with defined interfaces between them, including a hardware I/O layer, core logic layer, and application layer. The software contains both the logic, such as control algorithms, and parameters, such as gains for the control algorithm.

PHASE 2–

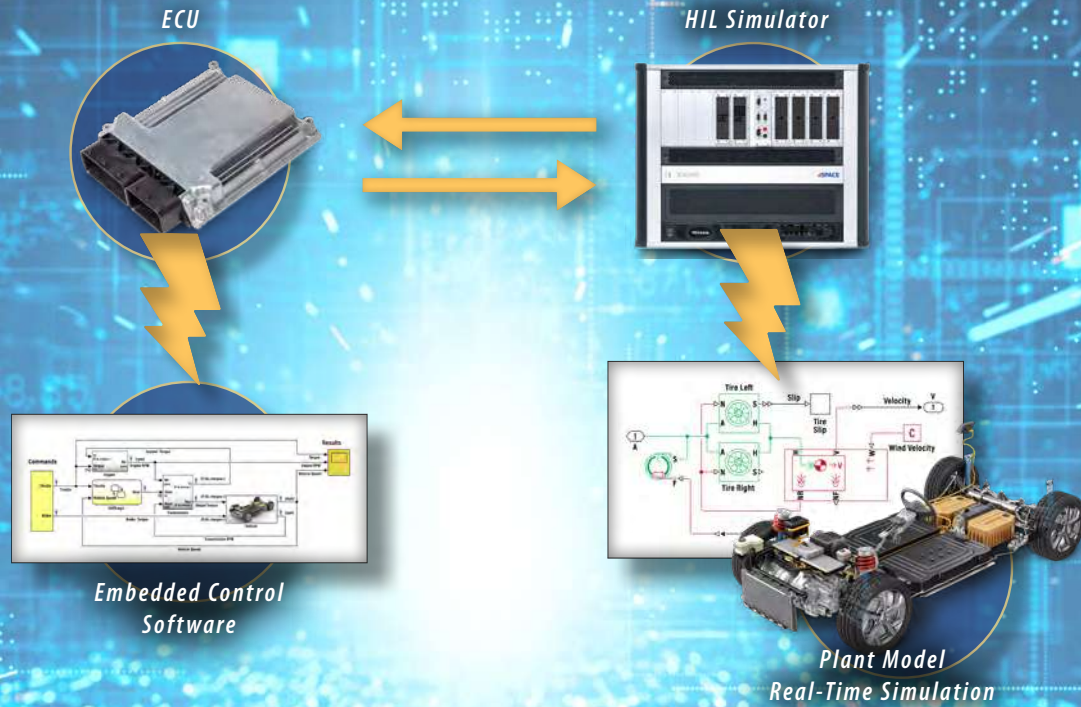
In Phase 2, the control software is tested on a target ECU using a Hardware-in-the-Loop (HIL) system. The software is auto-generated into C-code from the Simulink. The embedded control software runs on the ECU in real-time, as it will on the actual machine ECU. The HIL system simulates the behavior of the actual system and provides real-time input and output signals to the ECU. The plant model is simplified compared to Phase 1 to run in real-time. The HIL system hardware emulates the actual machine's signals. HIL testing allows for the identification and resolution of problems before testing on the actual machine. It is cost-effective and safer than testing on a prototype machine, especially for life-critical applications. The HIL system provides repeatability and allows for testing under extreme conditions that cannot be easily created in the real world. HIL testing is an intermediate step between pure software simulation and pure hardware testing.

HIL testing can be performed by independent engineering teams or companies, providing objective and exhaustive testing. It offers cost savings, better safety, and test repeatability. HIL testing by independent organizations is especially important for OEMs in order to assure impartial testing and validation of their products.

PHASE 3–

In Phase 3, The ECU is tested and tuned on an actual prototype machine. The I/O interfaces, sensors, actuators, and software logic are verified and calibrated. The control algorithm parameters are further tuned to optimize the dynamic performance. The machine's performance and reliability are compared to benchmark results, and the results are documented for production release. Matlab/Simulink files can be auto-code generated and flashed into the ECU using a laptop PC and a USB/CAN bus interface module. The laptop PC connects to the ECU's CAN bus, and MCD software tools implement communication protocols to read/write data memory locations of the ECU.

Various software tools like CANalyzer and CANape are used for HIL testing, enabling monitoring, test condition definition, data collection, and result analysis. These tools facilitate memory read/write access, test plan execution, and automation of test and validation processes.



Why is HIL Software Testing important?

1. HIL testing enables software engineers to test their programs in a virtual setting that closely mimics the actual hardware environment.
2. By detecting and resolving any flaws early in the development process, HIL testing can help to shorten development time and reduce costs.
3. An Extensive sets of scenarios can be tested in an automated process that cannot be tested on the actual hardware.

Improve Software Quality & Reduce the Risk of System Failure

Before the embedded software is installed on the actual hardware, HIL testing enables engineers to detect and fix errors in software. This decreases the possibility of system malfunctions, security hazards, and expensive recalls.

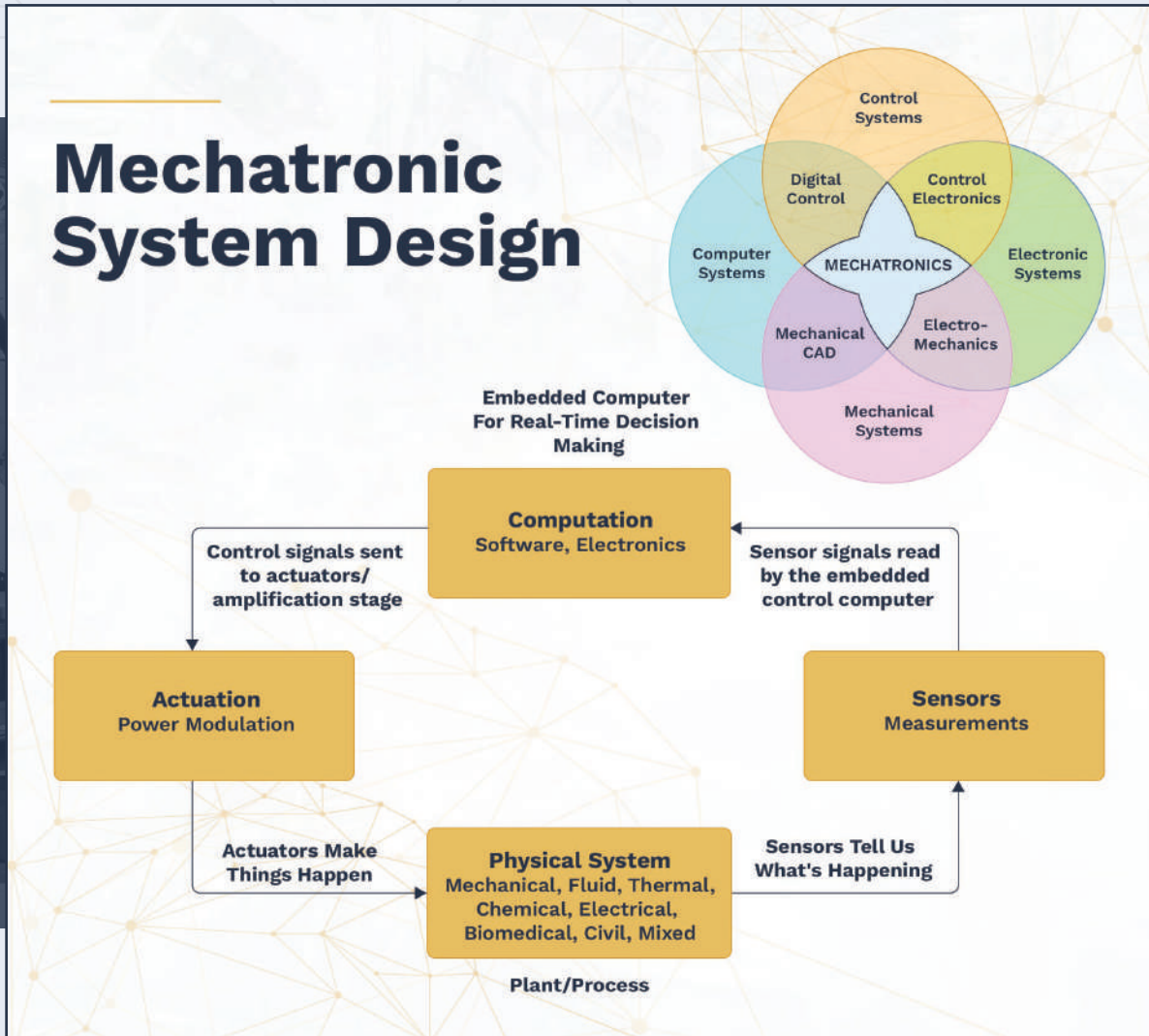
Reduced Development Time and Cost

By finding and fixing any errors or efficiency problems early on in the development process, HIL testing can help reduce development time and expenses. This reduces the need for expensive revision by ensuring that the software will operate properly and securely in the real-world environment.

Increased Test Coverage

Using HIL testing technology, that closely resembles the real hardware environment in which the software will work, enables increased test coverage, including tests that cannot be done on actual hardware, i.e. what happens if one of the wings of the airplane breaks off.

Mechatronic System Design



Mechatronic System Design refers to the synergistic integration of three distinct traditional engineering fields for system-level design processes:

1. Mechanical engineering where the word “Mecha” is taken from,
2. Electrical or electronics engineering, where “Tronics” is taken from,
3. Computer science.

A mechatronic system is a computer-controlled mechanical system, where the computer is an embedded computer, not a general-purpose computer, that is used for control decisions, and interfaced to the real world device via sensors and actuator interfaces.

Mechatronic System Design

A mechatronic system has the following components:

1. Mechanical system (i.e. EV power train)
2. Control computer (i.e. ECU)
3. Sensors (i.e. speed)

Mechatronic system design begins with the concept of mechanical system design that includes electrically controlled actuators, and sensors to measure variables that need to be controlled. Good design based on sound scientific and engineering principles forms the foundation of the overall quality of the system design. Computer control should always enhance the system capabilities, instead of trying to compensate for poor mechanical designs.

Once the hardware design is done, the next component of the system that defines its “brains” (intelligence) is the embedded software that goes into the ECU. Often, the ECU hardware, actuators, and sensors are standard off-the-shelf components. Software is always application-specific. Furthermore, as there is no limit on intelligence, then there is no limit on how “smart” and reliable we can design application software. It is common that over eighty percent of all mechatronic design engineering time is spent on software development.

Embedded software is different than general-purpose software in many different ways, even though they are both software; such as

Embedded software must run in real-time, and there can be life-and-death consequences of a few seconds of delay.

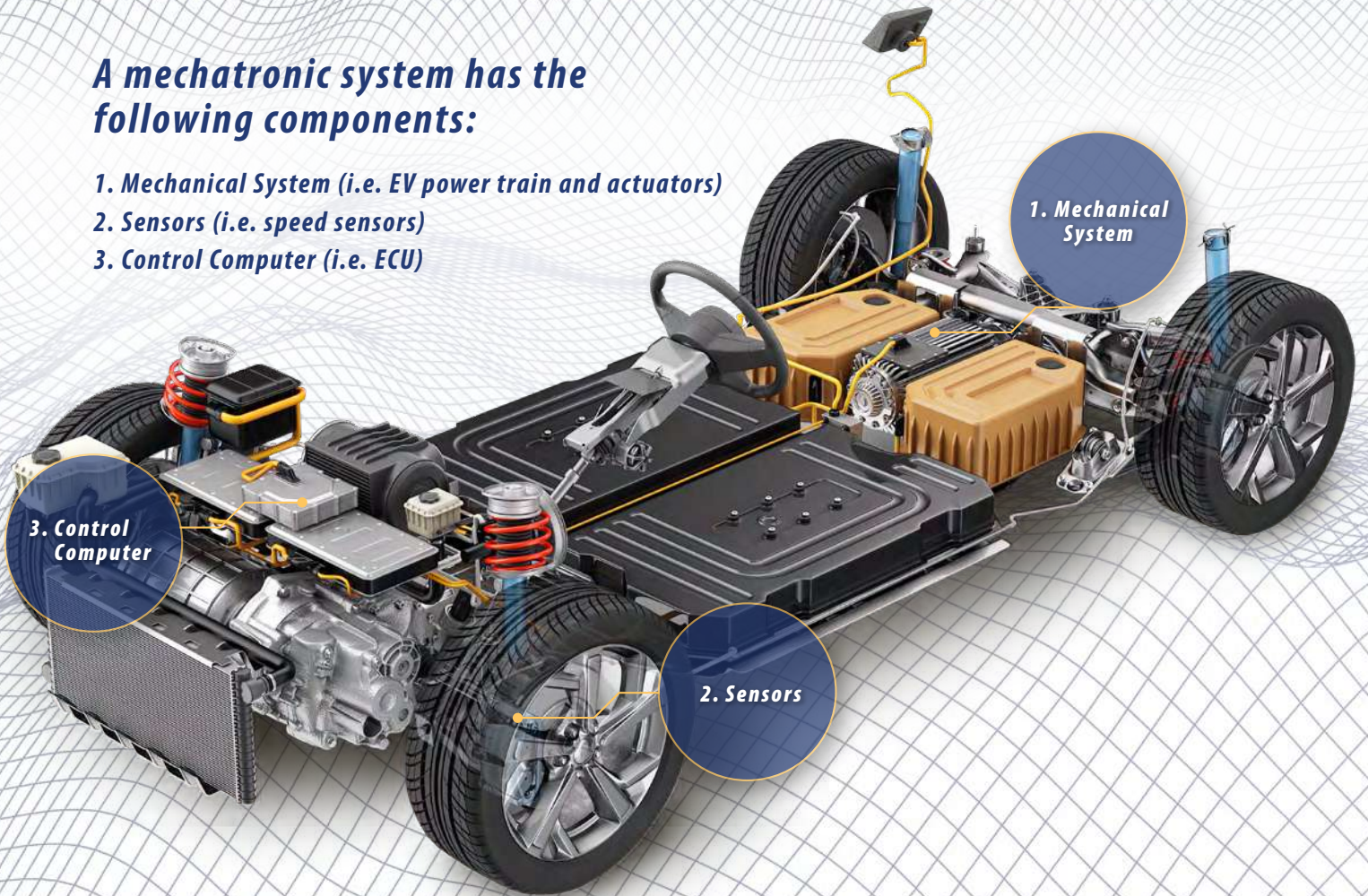
Embedded software must be reliable as lives may be riding on it.

Embedded software has limited CPU and memory resources.

Because of these critical requirements, expertise, discipline, experience, and pure intellectual capabilities of the development team is the most critical element of product development.

A mechatronic system has the following components:

- 1. Mechanical System (i.e. EV power train and actuators)***
- 2. Sensors (i.e. speed sensors)***
- 3. Control Computer (i.e. ECU)***



Mechatronic system design begins with the concept of mechanical system design that includes electrically controlled actuators, and sensors to measure variables that need to be controlled. Good design based on sound scientific and engineering principles forms the foundation of the overall quality of the system design. Computer control should always enhance the system capabilities, instead of trying to compensate for poor mechanical designs.

Once the hardware design is done, the next component of the system that defines its “brains” (intelligence) is the embedded software that goes into the ECU. Often, the ECU hardware, actuators, and sensors are standard off-the-shelf components. Software is always application-specific. Furthermore, as there is no limit on intelligence, then there is no limit on how “smart” and reliable we can design application software. It is common that over eighty percent of all mechatronic design engineering time is spent on software development.

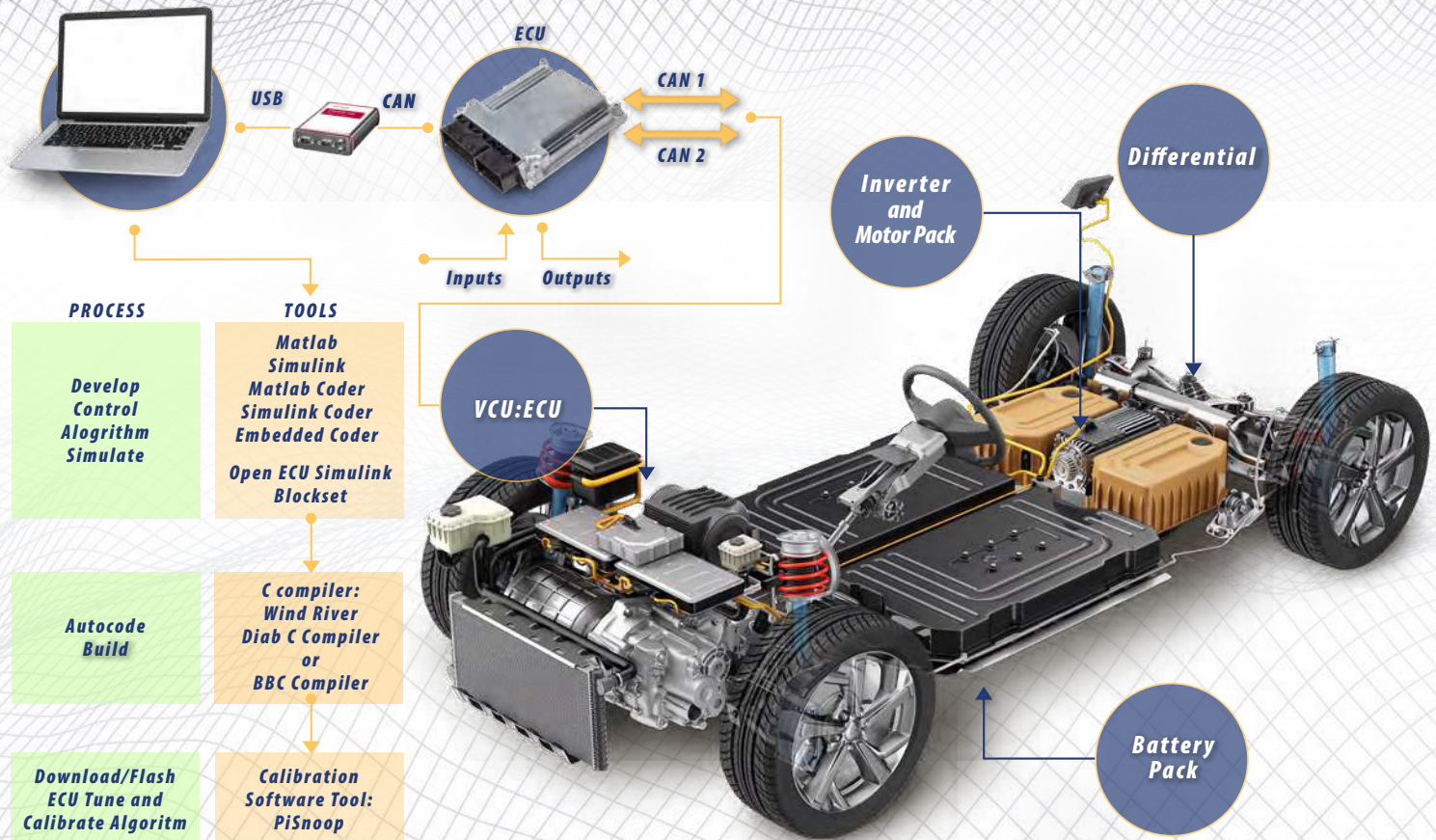
Embedded software is different than general-purpose software in many different ways even though they are both software

Embedded software must run in real-time, and there can be life-and-death consequences of a few seconds of delay.

Embedded software must be reliable as lives may be riding on it.

Embedded software has limited CPU and memory resources.

Because of these critical requirements, expertise, discipline, experience, and pure intellectual capabilities of the development team is the most critical element of product development.



The Embedded Software Development Tools

Hardware

1. Laptop PC
2. USB/CAN Interface Module

Software on the Laptop PC

1. Windows 10 (64 Bit)
2. Matlab
3. Simulink
4. Matlab Coder
5. Simulink Coder
6. Embedded Coder
7. Simulink Block Set for ECU Support
8. C Compiler (for target ECU i.e. Wind River)
9. Calibration and Tuning Software, i.e. CANape by Vector

The embedded software development and testing process consists of several steps, which are as follows:

1. Develop the control algorithm in the Matlab/Simulink environment:

In this step, the control algorithm is designed and implemented using the Matlab/ Simulink software. The algorithm is typically developed using a graphical interface, allowing for easy visualization and manipulation of the control logic.

2. Simulate, debug, test, and evaluate the control system:

Once the control algorithm is developed, it is simulated in the non-real-time environment of a laptop using Matlab Simulink. This allows for thorough testing and evaluation of the control system's performance under different scenarios and inputs. Any issues or bugs in the algorithm can be identified and debugged at this stage.

3. Auto-code the control algorithm for the Target ECU:

After the control algorithm has been verified through simulation, it is auto-coded to generate C code. Auto-coding is the process of automatically translating the algorithm from the Matlab/Simulink environment into executable C code that can run on the target Electronic Control Unit (ECU).

4. Build the C code for the target ECU and generate executable code:

In this step, the generated C code is compiled using a C Compiler tool. The compiler translates the human-readable C code into machine-readable binary code that can be executed by the target ECU.

5. Download/Flash the executable code to ECU:

Once the C code is compiled and transformed into executable binary code, it needs to be downloaded or flashed onto the target ECU. This process involves transferring the code from the development environment to the ECU using specialized tools and protocols.

6. Run the code on the ECU, tune, calibrate, test, and validate it:

After the executable code is loaded onto the ECU, the control system is executed on the actual hardware. The system is tuned, calibrated, tested, and validated in real-time using Calibration and Tuning Software running on a laptop PC. This allows for fine-tuning of control parameters, configuration of parameters for different product models, performance evaluation, and validation of the control system's behavior under real-world conditions. HIL Testing is highly effective before testing on an actual vehicle.

Overall, these steps ensure a systematic approach to developing and testing control systems, starting from algorithm development and simulation, to auto-coding, building, downloading, and finally validating the control system on the target ECU.



• **SYSTEM MODEL DEVELOPMENT**
System, Sub-Systems, Components

• **CONTROL SOFTWARE DEVELOPMENT**
Simulink, S-Functions

• **SIMULATION GUI**

• **SIMULATION, ANALYSIS, OPTIMIZATION**

• **FLASH CODE TO ECU, DEBUG & TUNE**

In the realm of modern engineering, the concept of Model-Based Design (MBD) emerges as an efficient, accurate, and collaborative development processes. Central to this approach is the synergy between MATLAB and Simulink, two software tools that uphold the intricate architecture of MBD, shaping the way complex systems are designed.

Understanding Matlab and Simulink

Matlab and Simulink, developed by MathWorks, are integral to Model-Based Design.

Matlab: A high-level programming language, Matlab is widely used for mathematical computations, data analysis, and algorithm development.

Simulink: A graphical environment, Simulink allows users to model, simulate, and analyze.

Fundamentals of MATLAB and Simulink

Building Mathematical Models:

MBD's foundation rests on translating abstract concepts into tangible forms—an interplay of equations and symbols within MATLAB. MATLAB's symbolic mathematics shapes models into Simulink-compatible elements, setting the stage for system realization.

Dynamic System Simulation:

Simulink transforms block diagrams into an engineer's playground, sculpting the behavior of dynamic systems. These graphical arrangements interconnect system components, choreographing a simulation-ready performance.

Control System Design and Analysis:

Simulink hosts the choreography of precision in control system design. Crafting algorithms and tuning controllers become artistic pursuits on a virtual stage. Engineers compose symphonies of responses, observing the harmony between stability and performance.

Model Validation and Verification:

Prior to the real-world debut, validation and verification take the spotlight. Simulations meticulously test scenarios and probe model boundaries, ensuring accuracy. This meticulous process guarantees a reliable performance.

Real-Time Code Generation:

MBD's pinnacle arrives as virtual elegance transforms into reality. Simulink's magic converts models to code, then to life. Embedded systems embrace the code, as algorithms infuse hardware with vitality.

Rapid Prototyping and Iteration:

Rapid Prototyping and Iteration: Prototypes, swift and ephemeral, materialize in moments. Iteration is their rhythm, a dance between adjustments and insights. MBD offers engineers a pas de deux with time, fostering innovation through rapid cycles of modification, simulation, and refinement.

MBD finds applications in various industries:



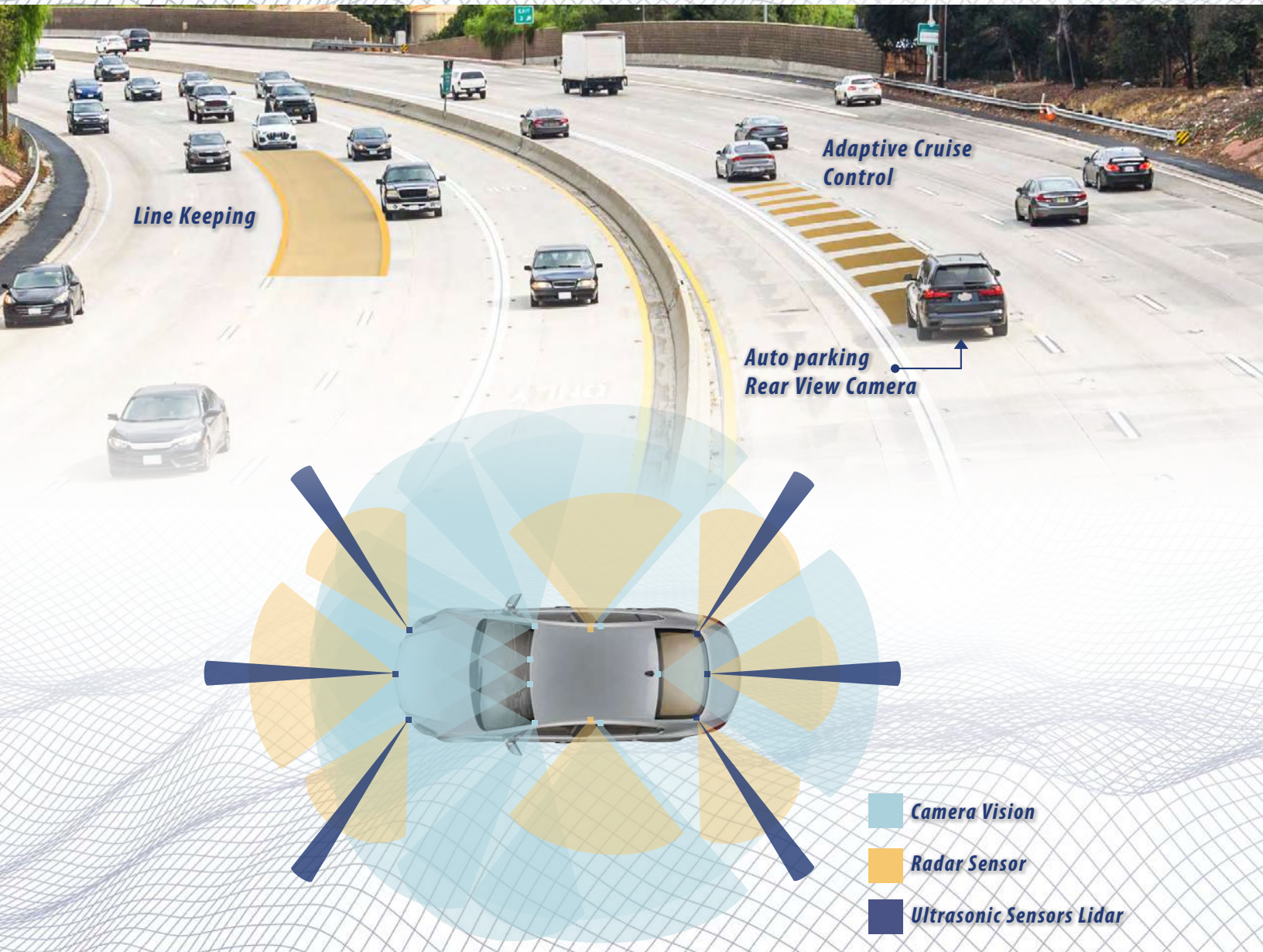
Automotive
Design and test vehicle control systems.



Aerospace
Develop flight control systems and avionics.



Medical Devices
Design and simulate medical equipment and devices.



The self-driving car decides on the command signal to subsystems (throttle command, brake command, steering command) based on large collection of sensors. Each sub-system then would have its own closed or open loop local controllers to achieve the command signals. This requires the car to have an electronically controlled powertrain, and an electronically controlled steering system (which is called an “x-by-wire” control system). The biggest challenges in this technology is the accurate detection of other cars and objects around the car and road conditions.

This is referred to as the “perception” problem in autonomous driving. If we can accurately determine the actual road conditions, then in the software we can decide what to do. The intelligence of the self-driving car is the sensors and software running in the on-board embedded controllers (multiple embedded controllers with a local communication network, such as CAN bus) as well as the server software running in cloud computing platform on internet-connected supercomputers to assist and supervise millions of self-driving cars. It is important to note that the software involved in autonomous vehicle control involves software running on typically embedded controllers on the vehicle and software running on servers on the internet which communicates with the embedded controllers in real-time.



Current developments are grouped under the name of “autonomous driver assistance systems” (ADAS).

It is a huge challenge to develop sufficient confidence in embedded software to make completely autonomous control decisions and take control actions. Therefore, the current trend is to release these capabilities as autonomous (or automatic) driver assistance systems (ADAS). The ADAS software module are developed as real-time embedded signal processing algorithms that take the processed sensory data (GPS, camera, radar/lidar, ultrasonic sensors, etc.) and make decisions, then send control signals to the engine-transmission / EV motor-drive, brake, and steering sub-systems. Standardized approaches to software development are developed and practiced (i.e. AUTOSAR) in terms of communication between modules and software layers and are intended to increase reliability and reduce development costs by increasing reusability.

PRECISION FARMING TECHNOLOGY: Embedded Control Software



US Department of Agriculture estimates that in year 2022 the farmers increased their yield by 48% by the use precision farming technology. This is an amazing productivity improvement in food production. Imagine a Farmer is walking around in a farm inspecting each plant, and spraying some liquid on each plant as needed. Precision farming technology does the same thing, except it does it at least 1000 times faster and more accurately with autonomous farming equipment.

Precision farming technology refers to the use of state of art sensors such as cameras, Lidar, Radar, GPS and embedded edge plus cloud computers to plan-measure-control the individual plant condition and deliver the correct spray material or operation that is optimal for that plant in real time. For example, a spraying machine (driven remotely or autonomously), equipped with cameras (i.e. 24 cameras, 5000 frames/sec mounted on a 80 feet long boom), utilize the vision system data to assess the condition of plants as it is passing over, and decide on **what** and **how much** and **where** to spray the necessary fertilizer or water in an optimal manner. When this concept is applied to the soil preparation, planting and spraying equipment, it leads to a very precisely controlled farming, hence increased productivity. Precision farming technology is about precision delivery of liquid spray or product by determining “what” and “how much” and “where” to deliver in real time with millisecond accuracy.

PRECISION FARMING TECHNOLOGY: Embedded Control Software

There are three main subsystems in precision farming technology: sensors (cameras, GPS, Lidar, radar), electrically controlled actuators (valves for spraying or tilling actuator control), and embedded computers (i.e. 12 GPUs) to process the data and take action in real time within a few milliseconds as the machine is moving over the field at 20 miles per hour speed. Offline planning using GPS surveys is often used to plan the operation for the whole farm, whereas the Online system is used to do the necessary work in millimeter accuracy.

The application of the precision farming technology includes the following phases of farming

- ✔ Soil preparation: tilling, fertilizing, irrigating.
- ✔ Planting: seed spray by volume, individual baby plant placement in soil.
- ✔ Spraying at different times during a growth season against pesticides, additional fertilizing and/or irrigation.
- ✔ Harvesting: collection of the product and storage for transportation.

Vision cameras, LIDAR, Radar, GPS are well developed technologies. Solenoid controlled valves used in spray equipment are also rather well developed.

The “brains” of the operation is the embedded software that runs on the CPUs/GPUs of the embedded controller on the machine. The embedded software must process all the sensor data reliably and on-time so that precision delivery can be made at the right location (where), right amount (how much) and the right spray liquid (what). The success of the system depends on the reliable, accurate and real-time operation of the system, all of which depends on how well the embedded control software works with the hardware.

System components: All integrated over CAN bus technology using CANopen or J1939 protocol

HMI: Human Machine Interface device for setup and configuration for application.

ECU: Electronic Control Units with CPUs/GPUs, the brain hardware.


SENSORS: Camera, LIDAR, GPS, Radar.

ACTUATORS: Solenoids for various valves and electric motor actuators for spraying and motion.

SOFTWARE: Using CODESYS, Matlab/Simulink, C/C++, Python. Embedded software to implement the “brains” and connect all these devices to work together.

Servotech has decades long expertise and experience in embedded control software development for many industries.

CAD/FEA DESIGN AND ANALYSIS

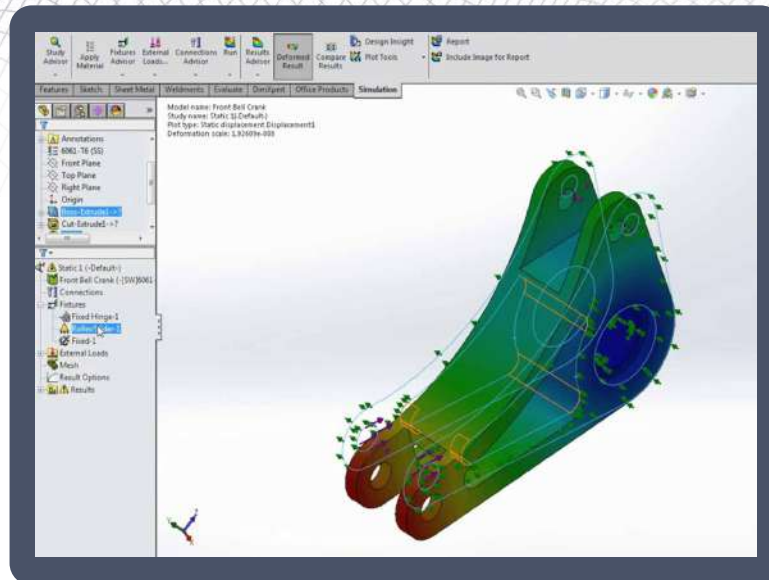


**CAD/FEA
Design and
Analysis**

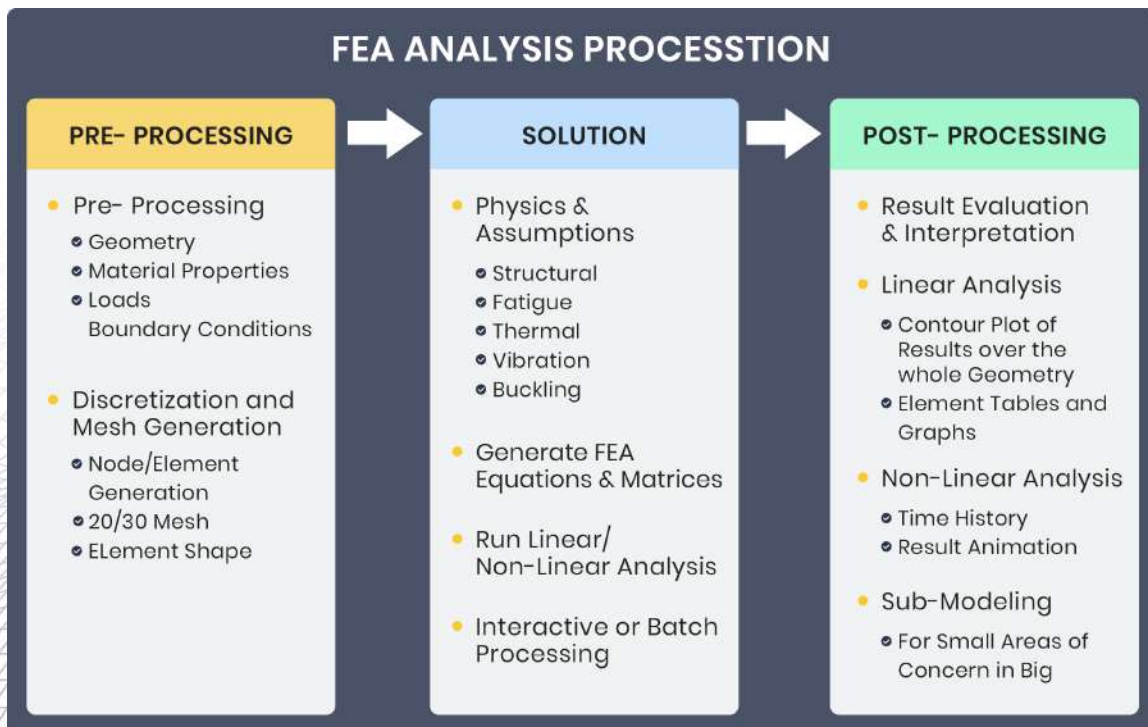
- Mechanical Design
- FEA Analysis: Thermal, Stress, Flow
- FMEA: Failure Mode and Effect Analysis
- Plant Design & Engineering
- Digital Factory & Simulation

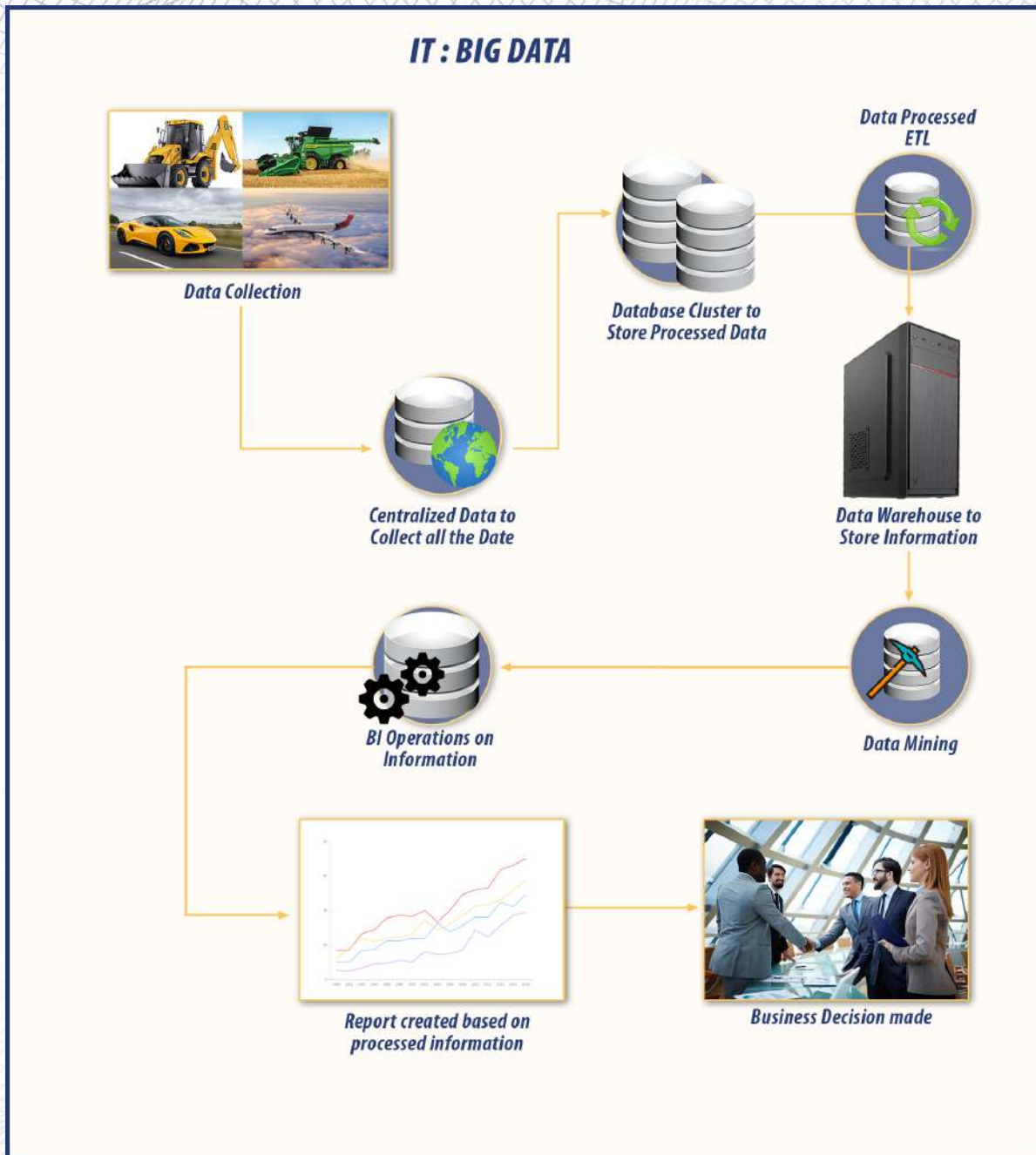
Computer Aided Design (CAD) software tools (i.e. AutoCAD, Inventor, SolidWorks, Creo) are used to design the mechanical system using 3D solid modeling, geometric dimensioning and tolerancing standards and practices, animation for geometric functional validation, integration to Finite Element Analysis (FEA) software tools for analysis, integration to controllers for Hardware-in-the-Loop (HIL) testing visualization.

Finite element analysis (FEA) software tools implement the physics-based mathematical equations and numerical solutions in the background. All general-purpose FEA software tools (ANSYS, Abacus, etc.) provide a graphical user interface (GUI) to define the problem and desired simulation conditions: that is to define the 3D geometry, material properties (i.e. a rectangular plate made of cast aluminum), simulated conditions (external load conditions and boundary conditions). Then the FEA software provides tools to automatically customize finite element mesh, constructs the physics-based equations, and solves them using a selected numerical solution method.



The results are then presented in the form of field variable distribution over space (for static simulations) and time (for dynamic simulations). For instance, the simulated results can be stress, strain, temperature, pressure, and fluid speed as a function of location in space (x,y,z) and time.





In the modern business landscape, the ability to make informed decisions quickly is paramount. Servotech is at the forefront of this evolution, offering comprehensive support to our customers throughout the entire decision-making process. From data capture to analysis and actionable insights, we ensure that every step leads to smart, data-driven business decisions.



Capturing and Analyzing Data

The journey toward smarter business decisions begins with the accurate capture of real-time data. Servotech employs advanced sensors and cutting-edge data acquisition systems to collect vast amounts of real-time data from engineering systems. This data, often referred to as “big data,” is then processed using state-of-the-art data analytics software tools. These tools leverage sophisticated mathematical algorithms and statistical methods, enhanced by artificial intelligence, to analyze and interpret the data efficiently.

A Paradigm Shift in Product Support and Maintenance

The exponential growth in the availability of real-time data has revolutionized product support and maintenance strategies. No longer do businesses rely solely on reactive maintenance practices. With real-time sensor data at their disposal, companies can now adopt predictive and prescriptive maintenance models. This shift not only enhances operational efficiency but also minimizes downtime, leading to significant cost savings and prolonged equipment lifespan.

Optimizing Operations and Engineering Design

Real-time data analytics extends beyond maintenance. By continuously monitoring system conditions, businesses can identify potential weaknesses in engineering designs. This proactive approach allows for timely design modifications, ensuring that products remain robust and capable of meeting evolving challenges. The insights gained from real-time data empower businesses to optimize operations and make informed decisions regarding engineering design changes.

Servotech’s expertise in data capture, analysis, and application is instrumental in driving smart business decisions. By harnessing the power of real-time data, we help businesses optimize their operations, enhance maintenance strategies, and continuously improve product designs. This holistic approach ensures that our customers are equipped to make decisions that lead to sustained growth and success.