Issue 2/2026

The Engineering City Journal

Published by the Engineering Association *Yerevan, February 4, 2026*

The purpose of the Journal is to disseminate knowledge, experience, and best practice in engineering & manufacturing.
The articles are written by practicing professionals in the Engineering City (EC) in Yerevan. They cover issues relevant to their projects, operations, and organizations.
The articles address the challenges of the time. As such, the Journal is also a chronicle of the evolution of the EC, its companies, and its individuals.
It is the aim of the Journal to advance the engineering & manufacturing practice, to enrich the engineering & manufacturing community, and to contribute to the worldwide discourse in engineering & manufacturing.

Table of Contents

AI Corner 1
. Artificial Intelligence as a Baseline Layer of Modern Engineering 1
New Products Corner 6
. In-Vehicle Data Logger from YEA Engineering 6
IB-TDS (Industry-Based Talent Development System) 7
. Business Automation at DigiCore: How Everything Works When I'm Not There 7
EC News Corner 10
. A New Tradition in the EC – The Cafeteria as a Platform for New Product Introduction 10
Armenology 11
. Armenian Equivalents to Foreign Words, 4 11

AI Corner

. Artificial Intelligence as a Baseline Layer of Modern Engineering

by Grigor Shirinyan, DigiCore

How AI is moving onto every kind of hardware and what engineers need to know next.

In the next decade, AI will not be something you "add" to a product at the end. It will be something the product is designed to host from day one - whether that product is a sensor node, an RF system, a piece of embedded hardware, or an application running on an AI-enabled laptop.

This article explains: what is changing in hardware, why models must be optimized, and why every engineer working in RF, software, hardware, embedded systems, or applications will need a working understanding of machine learning.

1. The shift: AI leaves the cloud

Not long ago, the common pattern was simple: devices collected data, sent it to the cloud, and the cloud made the "smart" decision. That model still exists - but it is no longer enough for many real systems.

Today we see a move toward edge computing: doing more computation close to where the data is generated. Cloud AI still matters and will remain essential for training and heavy workloads, but when the computation is pushed closer to the device and is AI-driven, we call it edge AI. The practical reason is straightforward: many systems cannot afford delays, connectivity problems, or constant data transfer.

Three forces are pushing AI onto devices:

- Low latency: interactive applications, real-time control, and RF adaptation often need decisions in seconds or even faster.
- Privacy-by-design: when raw audio, images, telemetry, or user behavior never leaves the device, privacy and compliance are easier to guarantee.
- Cost and reliability: bandwidth, cloud costs, and network outages become less painful when the device can think locally.

Once you accept those constraints, the technical goal becomes clear: run on-device inference efficiently, even on limited hardware.

2. Where AI runs: from microcontrollers to AI PCs

AI does not run on just one "type of computer". It runs across a spectrum of platforms, each with different constraints and strengths. In practice, systems often use both Cloud AI and local compute: the cloud is used for training and heavy computation, while devices run inference locally when latency, privacy, or reliability matters.

Microcontrollers (MCUs) are the smallest class. They often have kilobytes of RAM and strict power limits. Yet they can run compact models through TinyML - useful for tasks like anomaly detection, gesture recognition, or simple classification from sensors.

Embedded Linux devices (small boards or modules running Linux) provide more memory, multicore CPUs, and sometimes accelerators. They are common in robotics stacks, industrial gateways, and edge servers.

Modern consumer and industrial devices increasingly rely on System-on-Chip (SoC) designs, where CPU cores, memory controllers, and I/O live on a single chip. Many SoCs include a dedicated NPU (Neural Processing Unit) to accelerate neural network operations efficiently.

All of this leads to heterogeneous computing: a system where different compute blocks (CPU, GPU, NPU, DSP) share the workload. Performance is no longer only about CPU frequency; it is about mapping the right operations to the right engine.

3. The engineering trick: making models fit

Running machine learning on limited devices is not magic - it is disciplined optimization. The same idea appears again and again: reduce what the model needs (memory, compute, energy) while keeping enough accuracy for the job.

Model compression is the umbrella term for these techniques. The most common tools are:

- Quantization: using lower-precision numbers (for example, 8-bit integers instead of 32-bit floats). This usually reduces model size and speeds up inference.
- Pruning: removing weights or channels that contribute little, making the model smaller and sometimes faster.
- Knowledge distillation: training a small "student" model to imitate a large "teacher" model, often keeping good accuracy at a fraction of the cost.

At a deeper level, teams increasingly use hardware–software co-design: choosing model architectures that match the target device's memory hierarchy and the set of operations that the accelerators can execute efficiently.

The result: models that are not only accurate in a notebook, but reliable in a product.

4. Constraints affecting the process

The model with the highest benchmark score is not automatically the right choice for deployment. In real projects, the deployed model is the one that meets the system constraints, and selecting it is a process: choosing the candidate model, optimizing it, benchmark testing, and validating it. This process also depends on choosing the system itself and clearly defining what constraints the system imposes.

In other words, deployment is not a simple “pick the best model on paper.” It is a process where performance must survive real constraints: latency, memory, compute, thermals, cost, reliability, and the environment where the device operates.

Energy efficiency is one of the key constraints, especially in embedded or thermally limited devices. Over time, fatigue (FATIC) or aging, external environmental impact, and the nature of the sensor itself can cause missing data over time (if you don’t treat the data accordingly).

Another major challenge is data drift: the real-world input distribution changes compared to what the model saw during training. A model should be maintained after deployment (monitoring, updating, re-validation) - these touches taking into account real sensor behavior, missing data, drift, and system constraints.

5. Why every engineer needs AI basics

As AI becomes part of shipped systems, companies need specialists who can make AI reliable over time. This is where the ML engineer role grows.

An ML engineer is not only someone who trains a model. They build the end-to-end system: data pipelines, evaluation, deployment, monitoring, updates, and rollback strategies.

As AI becomes a default capability of platforms, engineers in every one of these domains will interact with ML processes, particularly models - even if they do not train them.

Let’s see how it could be used in different applications.

Glossary

RF Engineering

ML can help with channel estimation, interference detection, signal classification, and adaptive resource management. To use these tools safely, RF engineers need to understand what the model is optimizing and where it can fail.

Embedded Engineering

On-device inference turns raw sensor streams into decisions. Engineers must reason about memory limits, latency budgets, quantization effects, and robustness - and also power usage for a model power budget.

Hardware Engineering

Hardware AI changes how we design, test and validate systems. It also helps us to bring the idea to the hardware requirements.

Software Engineering

ML sits inside products (search, ranking, personalization, anomaly detection). Engineers must interpret metrics, detect regressions, and understand data pipelines.

Application Engineering

In application engineering, we must account for the application requirements from the start. The same device may be used in different places, environments, and configurations, so the requirements can change depending on where and how it is used. If we do not consider these differences, the system may work well in one setting but degrade or fail in another.

6. What this means for the next decade

If you zoom out, a simple pattern emerges:

- Hardware is becoming AI-capable by default (SoCs with NPUs, edge modules, efficient runtimes).
- Models are becoming smaller and smarter through compression and co-design.
- Organizations will need AI-literate engineers everywhere.

AI on hardware becomes more realistic and will be a part of engineering roadmap.

Key concepts (short explanations)

Cloud AI

AI training and/or inference performed in centralized cloud infrastructure.

Edge computing

Computing done close to where data is generated (devices, gateways, local modules).

Edge AI

AI inference performed on edge devices rather than in a remote data center.

On-device inference

Running a trained model locally on a device.

Microcontroller (MCU)

A small, low-power computer on a chip used in embedded products.

TinyML

Methods and tools for running ML models on microcontrollers and other ultra-low-power devices.

Embedded Linux

An embedded device that runs a Linux operating system.

System-on-Chip (SoC)

A single chip that integrates CPU cores, memory controllers, and I/O (and often accelerators).

Neural Processing Unit (NPU)

A hardware accelerator optimized for neural-network operations with high performance per watt.

Heterogeneous computing

A design where multiple compute engines (CPU, GPU, NPU, DSP) share workloads.

Model compression

Techniques that reduce model size and compute cost while keeping performance acceptable.

Quantization

Reducing numeric precision (e.g., FP32 to INT8/INT4) to decrease memory usage and speed up inference.

Pruning

Removing redundant weights or channels from a model to reduce size and sometimes increase speed.

Knowledge distillation

Training a smaller model (student) to mimic a larger model (teacher).

Hardware–software co-design

Designing the model and deployment pipeline with the target hardware in mind.

Latency

The time it takes for a system to produce an output.

Energy efficiency

How effectively a system performs inference under a limited power/thermal budget.

Power budget

A defined limit for how much power the model/system may consume to stay stable and safe.

Data drift

A change in input data patterns after deployment, which can reduce model quality over time.

New Products Corner

. In-Vehicle Data Logger from YEA Engineering



Pic. One-Page Data Sheet for the In-Vehicle Data Logger
For an extensive 17-Page Datasheet, please contact YEA Engineering @ info@yeae.am

IB-TDS (Industry-Based Talent Development System)

. Business Automation at DigiCore: How Everything Works When I'm Not There

by Grigor Shiriniyan, DigiCore

Why We Needed Automation

I had to leave Armenia for some time, and one of the main things we wanted to do in DigiCore was to automate some of our processes. The most important goal was not just to keep the business running, but to protect the philosophy and the idea of DigiCore. DigiCore was never meant to be "just another course". It was created as a place where students start building a real engineering mindset and see a clear path into their future. Automation became a way to keep this idea alive even while I am not physically in Armenia.

Shared Philosophy and Dedicated Training

To do that, one of the key steps was to find people who share our philosophy, concepts, and ideas. DigiCore works with people who see education the same way we do and who understand why we are doing all of this. We also turned our main course into a dedicated training program for students. This program is not just a set of random lessons. It is a structured path where:

- students gradually build a strong, fundamental understanding of engineering,
- every new topic is connected to the previous one,
- each step is connected to a real-world idea or application
- and students constantly see why what they are learning matters.

In this dedicated training, students do not simply follow instructions. They are encouraged to think, ask questions, and make connections. The idea is that, by the end of the course, they don't just remember what they did in lessons, but understand how engineers think and how engineers solve problems. Our goal is also to show them that the educational programs in Engineering City are continuous. DigiCore is not an isolated point; it is the beginning of a longer journey. We want students to see that:

- there are more programs after DigiCore,
- these programs can continue up to university level,
- and there are real opportunities to work and grow as engineers in the same environment.

So this dedicated course is not just about knowledge. It is also about orientation: helping students see themselves as future engineers and helping them understand that they are already inside a bigger system with real prospects.

Automation of Lesson Processes

On top of that, we automated several important processes around the lessons themselves. Before each lesson, tutors receive a notification about the upcoming class. This reduces the chance of scheduling confusion and keeps everyone in rhythm. After each lesson, DigiCore trainers receive a notification to write a short report and fill in a short questionnaire about the lesson and the students. These reports and questionnaires are not long or complicated, but they are consistent. They include:

- what the main topic of the lesson was,
- how well students understood the key idea,
- which moments worked especially well,
- where students struggled or lost focus,
- and what should be repeated or changed in the next lesson.

There are also fixed times when we sit down, talk, and decide whether something should be improved or changed. This might be about the pace of the course, the difficulty of the tasks, or the way we explain certain concepts. By automating

these processes, DigiCore doesn't have to depend on someone "remembering to check" or "remembering to ask". The system itself pushes trainers to reflect and share information regularly.

Periodic Calls With Students: Knowledge and Emotion

Another key part of the system is periodic calls with students. These calls are organised at specific points in the course, not just when there is a problem. They are checkpoints built into the learning process. During these calls, we focus on two main dimensions:

1. Knowledge

- Can the student explain what they have been doing in lessons?
- Do they understand the main ideas, or are they just repeating steps from memory?
- Can they connect different topics to each other?
- Do they see the logic behind the tasks and projects?

2. Emotion and motivation

- Do they actually enjoy the lessons?
- Do they feel that "wow" effect when something works?
- Are they curious and asking questions?
- Do they feel more confident about engineering, or more scared and confused?

These calls are not exams. They are conversations. Students can share what they like, what confuses them, and what they would like to do more of. The information from these periodic calls is extremely valuable for DigiCore. It shows whether the program and the automated processes are really working from the student's point of view. If many students, from different groups, give similar signals—positive or negative—it becomes a clear message for us to keep something or to change something. Periodic communication with students in this way helps DigiCore stay close to students and understand their needs.

Families as an Important Part of the Feedback Loop

Since DigiCore works directly with families, they are a very important part of our feedback loop. Families choose where their children study, they see them at home every day, and they often notice changes earlier than anyone else. That's why we try to keep families informed and involved in what is happening at DigiCore. We talk with families to understand:

• how their children feel after lessons,
• whether they are more motivated and confident,
• whether they talk about DigiCore and their projects at home,
• and whether the experience matches what the family expected.

Their feedback becomes another "sensor" in our system. If several families from different groups mention the same thing — a problem or a positive change — it is a clear signal that we need to pay attention to it and maybe adjust something in our processes or in our teaching.

Social Media as a Continuation of the System

Social media is not a separate element for us; it is a continuation of the same system of communication and transparency. Through our social media channels we:

• post about upcoming courses and new groups,
• share results of DigiCore students from their daily studies, • show small moments from lessons and projects,
• and remind everyone what DigiCore stands for.

This helps keep families informed, even if we don't talk with them personally every day. They can see what DigiCore is doing, what students are working on, and what kind of progress is happening inside the center. For DigiCore, social

media is another way to keep our philosophy and ideas visible and to show that the system is alive and active, even when I am not in Armenia.

What Automation Gives to DigiCore

All of these elements — people who share our philosophy, the dedicated training course for students, automated lesson processes, periodic calls with students, feedback from families, and ongoing communication through social media — form one system. Automation gives DigiCore several important things:

- Stability: Work does not depend on one person being physically present. The processes keep running even when I am away.
- Consistency: The quality of lessons and the core idea stay the same from group to group and from week to week.
- Visibility: We have a clear picture of what is happening with lessons, students, and families, instead of relying only on guesses.
- Early problem detection: Issues can be noticed and fixed quickly, before they grow into something bigger.
- Focus on the idea: Instead of spending all our energy on manual control and constant “firefighting”, we can focus on developing the concept, the philosophy, and the long-term path for students.

There is still a lot that can be improved at DigiCore. The system is not finished, and it will continue to change and grow. But even now, this automation helps DigiCore keep what matters most: the quality of education, the idea behind the center, and the belief that it is building a real future for young engineers.

EC News Corner

. A New Tradition in the EC – The Cafeteria as a Platform for New Product Introduction

by Rama, Engineering Association

We always battled with the challenge of announcing about the newest products that have been designed, developed, and produced in the EC to the wide EC community.

We realized that the Cafeteria is the place where most of the engineers come together daily, so why not make the cafeteria the News platform for the EC.

In the below picture you can see how the newly produced "In-Vehicle Data Logger" from YEA Engineering has been displayed in the middle of the Cafeteria during lunch time.

And you will be surprised to know that two sales leads were generated from this display.



Pic. The Cafeteria as an effective News Platform for the EC.



Pic. The "In-Vehicle Data Logger" from YEA Engineering

Armenology

. Armenian Equivalents to Foreign Words, 4

Հայոց լեզուն՝ ի տարբերություն, օրինակ, անգլերենի և ռուսերենի, հետևողական պայքար է մղում ինքնամաքրման՝ փորձելով յուրաքանչյուր օտար բառի դիմաց ստեղծել դրա համարժեք հայերեն տարբերակները։

Անի Մակինյան
Հայոց լեզվի և գրականության ամբիոնի դեկավար
«Շիրակացու ճեմարան»

Օտար բառեր	Հայերեն համարժեքներ
գլազուր	ջնարակ
գոնկա	1.մրցարշավ, 2.հետապնդում
դաունլոդ /հմկրգչ./	ներբեռնել
դեսերտ	աղանդեր
դիսկուրս	խոսույթ
դոպինգ	խթանիչ
դրայվ	եռանդ
էպիզոդիկ	դրվագային, դիպվածային

© 2025 The Authors of the articles

All materials contained in *The Engineering City Journal* may be used with reference to the Authors.

For feedback on the *Journal*, its content, and its articles please send to info@engineeringcity.am.

To submit articles, please send to info@engineeringcity.am. Articles can be in any language.