FOR FACULTY · 10 MINUTES

# AI in Engineering Education

*Engineers use Embedded Systems to create Artificial Intelligence.*
*This is what your students need — and what this resource provides.*

ECE · EEE · CSE · Mechanical Engineering · Faculty · Curriculum Designers

Rev 1.0 June 2026

# The World Your Graduates Are Entering

Every intelligent physical system being built today — across every engineering discipline — has one thing in common:

**It makes decisions autonomously. It does not wait for a human to interpret data and act.**

| Engineering discipline | Where AI is already present in physical systems |
|---|---|
| Electrical Engineering | Smart grid fault detection, power quality monitoring, transformer health |
| Electronics Engineering | Predictive maintenance, battery management systems, sensor fusion |
| Mechanical Engineering | Rotating machinery health, vibration analysis, structural monitoring |
| Automotive Engineering | ADAS, engine fault diagnosis, battery state estimation, noise detection |
| Aerospace Engineering | Flight control, structural health monitoring, navigation under uncertainty |
| Biomedical Engineering | Wearable diagnostics, ECG anomaly detection, implant monitoring |

The question is no longer whether your graduates will encounter AI in their careers.

**The question is whether they will be the ones who build it — or the ones who depend on those who do.**

# The Embedded System Is Where Engineering Begins

In a four-year engineering programme — the creation of embedded systems hardware is where hands-on engineering learning begins. It is the foundation of application engineering across every electrical and mechanical discipline.

**What students already build:**

- Microcontroller circuits
- Sensor interfacing — ADC, I2C, SPI
- Actuator control — PWM, motor drive
- Firmware in C — interrupt driven
- Signal processing — FFT, filtering
- Communication — UART, CAN, Ethernet
- Real time control loops
- Power management circuits

**This is already AI-ready hardware.**

Every skill the student has built maps directly to AI deployment:

- ADC → feature extraction input
- Flash memory → model weight storage
- C firmware → inference engine
- Interrupt loop → inference trigger
- FFT → signal feature extraction
- UART/CAN → decision output
- Control loop → intelligent actuation
- Low power design → TinyML deployment

# The Natural Next Step — Making It Intelligent

AI does not replace the embedded system. It is the next layer built on top of it — using the same hardware, the same C skills, the same sensor understanding the student already has.

**The progression is natural and continuous:**

**Year 1–2**

### Embedded System

Build the hardware — MCU, sensors, actuators, firmware in C. If student uses ready hardware, he loses the real hands-on creation experience of Engineering. So, they must, design, build and test.

**Year 3**

### Signal Processing

Understand the data — FFT, filtering, feature extraction

**Year 4**

### AI Enabled System

Add intelligence — model in flash, inference engine in C, autonomous decision

**Creating an AI enabled embedded system is the capstone of intelligent electrical, electronics, and mechanical engineering education.**

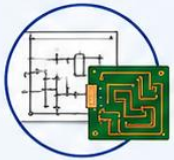
# RE-ARCHITECTING THE FOUR-YEAR RUNWAY: ONE DEPLOYABLE PRODUCT PER TEAM

Building expertise through a progressive pipeline from fundamentals to field deployment.


## STUDENT DEVELOPMENT ⟶ PIPELINE

### Years 1 & 2: THE FOUNDATION

**Component & Datasheet Mastery**
- Read a data sheet
- Electronic design basics



**Schematic & 2-Layer PCB Design**
- Design schematic
- 2-layer PCB layout
- Design rule basics

**Basic Analog Sensor Interfacing**
- Connect simple analog sensors
- Read and analyze analog signals

### Year 3: THE INTELLIGENCE

**Multi-layer Hardware Bring-up**
- Test and validate multi-layer hardware
- Debug with lab instruments


**Bare-Metal Firmware & TinyML**
- Program, embedded C/C++
- Code optimization
- TinyML development & deployment

**Core System Integration**
- Integrate all core components
- Validate in real-world like conditions

### Year 4: THE DEPLOYMENT

**Rugged IP-Rated Enclosures**
- Select and design robust enclosures
- Ensure sealing, thermal and mechanical reliability

**Real-world Field Stress Testing**
- Environmental testing
- Long duration reliability
- Performance validation in field conditions

**The Goal: A Working Physical Box**
- Deployable end-product
- Ready for real-world use and demo

### OUTCOMES AT THE END OF YEAR 4

**Deployable Product**
A fully functional, rugged, field-tested physical product.

**Industry-Ready Skills**
Hands-on experience across hardware, firmware, testing and system integration.

**Team Collaboration**
Cross-functional teamwork to build and deliver a real-world product.

**Real-World Impact**
Solutions that solve real problems and create measurable impact.

### THE GOAL

By the end of Year 4, each team will deliver one deployable, field-tested physical product that demonstrates engineering excellence, innovation, and impact — ready for industry, incubation, or startup.

### THE JOURNEY

Learn (Fundamentals) → Build (Intelligence) → Validate (Real World) → Deliver (Impact)

ENGINEERING INDIA

# Common Architecture for ML, AI Hardware, Software Solutions

## PHYSICAL WORLD

Machines

Industrial Equipment

Buildings & Infrastructure

Energy Systems

Mobility

Environment

... and more

## SENSOR PLATFORM

**Sense**
Capture the right parameters

**Excite Correctly**
Stimulate / measure under controlled conditions

**Calibrate**
Ensure accuracy and consistency over time

**Anti-foul**
Protect from environmental effects and degradation

**Trust the Reading**
Ensure reliability and repeatability of data

## EMBEDDED AI PLATFORM

**Process**
Filter, clean, transform data

**Infer**
Run ML models / AI algorithms

**Decide**
Determine what to do

**Act**
Control / adjust systems

**Alert**
Notify stakeholders

**Log**
Store events, metrics, context

**Sync**
Transmit securely to cloud / edge or other systems

## GOVERNMENT API

Secure, standardized APIs for integration, compliance and data exchange

## DASHBOARD

Visualize insights, monitor performance, manage alerts and reports

## IMPACT

Better decisions, optimized operations, reduced risk, measurable outcomes

## Cross-Cutting Enablers

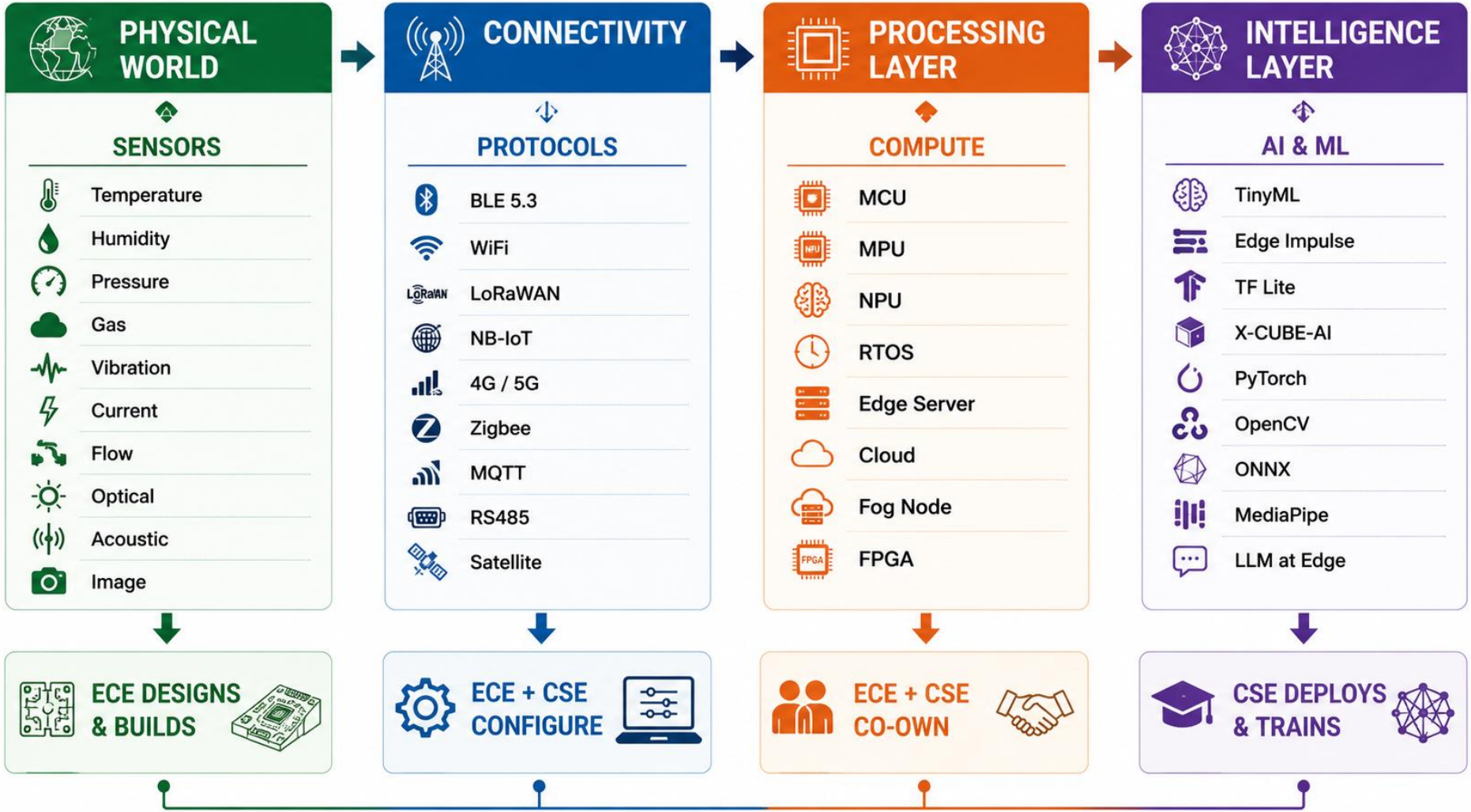
**Security**
End-to-end security & data privacy

**Connectivity**
Edge, Cloud, IoT protocols

**Data Management**
Quality, governance & lifecycle

**Device Management**
Provision, monitor, update OTA

**Scalability**
From edge to cloud at scale


PHYSICAL WORLD
SENSORS
Temperature
Humidity
Pressure
Gas
Vibration
Current
Flow
Optical
Acoustic
Image
CONNECTIVITY
PROTOCOLS
BLE 5.3
WiFi
LoRaWAN
NB-IoT
4G / 5G
Zigbee
MQTT
RS485
Satellite
PROCESSING LAYER
COMPUTE
MCU
MPU
NPU
RTOS
Edge Server
Cloud
Fog Node
FPGA
INTELLIGENCE LAYER
AI & ML
TinyML
Edge Impulse
TF Lite
X-CUBE-AI
PyTorch
OpenCV
ONNX
MediaPipe
LLM at Edge
ECE DESIGNS & BUILDS
ECE + CSE CONFIGURE
ECE + CSE CO-OWN
CSE DEPLOYS & TRAINS

# Smart Sensor Platforms (SP) – From Medium to Meaningful Intelligence

## SP-1
Electrochemical Sensor Platform

### Soil · Water · Gas · Anything with electrodes in a medium

**Core design:** SS316L or platinum electrodes · AC excitation 1–10 kHz · NABL calibration · anti-fouling wiper or coating · IP67 housing · drift monitoring in firmware

- AC Excitation 1–10 kHz
- Anti-fouling Wiper / Coating
- NABL Calibration
- IP67 Housing
- Drift Monitoring in Firmware

**Solutions**

- Soil Intelligence
- Drip Irrigation
- Aquaculture
- River Quality
- Ground Water
- Effluent OCEMS
- Coal Mine Gas

## SP-2
Optical & Thermal Sensor Platform

### NIR · IR · Camera · Luminescence · Particle counting

**Core design:** optical window material matched to wavelength · anti-fouling coating or heated purge · LED/laser drive circuit · thermal compensation · NABL or CPCB calibration

- Optical Window Matched
- Anti-fouling Coating / Heated Purge
- LED / Laser Drive Circuit
- Thermal Compensation
- NABL / CPCB Calibration

**Solutions**

- Grain NIR Grading
- Pest Acoustic + Camera
- Hyperlocal AQI PM
- Quality Inspection
- Forest Fire + Wildlife

## SP-3
Physical & Mechanical Sensor Platform

### Vibration · Strain · Pressure · Flow · Acoustic emission

**Core design:** MEMS accelerometer or piezo · differential signal conditioning · magnetic or adhesive mount · impact hammer calibration · hermetic sealing · ISO 10816 compliance

- MEMS / Piezo Sensing
- Differential Signal Conditioning
- Magnetic / Adhesive Mount
- Impact Hammer Calibration
- Hermetic Sealing ISO 10816

**Solutions**

- MSME Predictive Maintenance
- Railway Track
- Water Leak Acoustic Correlator
- Flood Warning Pressure
- Deep Ocean BPR

## SP-4
Biometric & Clinical Sensor Platform

### ECG · $SpO_2$ · Temperature · Body-contact — AC coupled mandatory

**Core design:** Ag/AgCl body-safe electrodes · AC coupled front end — no DC path to body · right-leg drive circuit · IEC 60601-1 safety isolation · CDSCO Class B pathway

- Ag/AgCl Electrodes
- AC Coupled Front End (No DC Path to Body)
- Right-Leg Drive Circuit
- IEC 60601-1 Safety Isolation
- CDSCO Class B Pathway

**Solutions**

- Rural RPM Wearable
- Maternal Monitoring
- Mental Health HRV

**Built for Accuracy, Reliability & Compliance** | Calibrated. Protected. Trusted. | From Sensing to Impact.

# The Embedded AI Platform Family — EAP-1 to EAP-3

## EAP-1: TinyML Node

**STM32N6 · Ethos-U55 NPU · 4.8 TOPS · No Linux · Battery powered · Real-time**


- Ultra low power
- Real-time Inference
- Wireless Connectivity

| Design element | Specification |
|---|---|
| Processor | STM32N6 — Cortex-M55 + Ethos-U55 NPU |
| AI framework | X-CUBE-AI — Keras/TF/ONNX → optimised C |
| Best for | Audio classification · vibration anomaly · basic image |
| Power | µA sleep · mA inference · years on battery |
| Connectivity | LoRa STM32WL or BLE nRF5340 |
| POC board | STM32N6570-DK (₹6,000–9,000) |
| Engineering target | Custom 4-layer PCB + STM32N6 chip |

**Solutions**

MF-1 (MSME vibration) · GR-3 (pest acoustic) · AG-4 (livestock) · TR-3 (railway AE)

## EAP-2: Edge AI Gateway

**STM32MP2 / RPi CM4 · Linux + real-time · Multi-sensor fusion · Government API**

- Cloud / API Ready
- Multi-sensor Fusion
- Secure & Updatable

| Design element | Specification |
|---|---|
| Processor | STM32MP257 (A35+M33+NPU) or RPi CM4 |
| AI framework | ONNX Runtime · TF Lite · Python on Linux |
| Best for | Multi-sensor fusion · government API · MQTT broker · OTA |
| Power | Mains or large battery + solar |
| Connectivity | NB-IoT · LoRa gateway · 4G · WiFi |
| POC board | STM32MP157F-DK2 or RPi CM4 + IO Board |
| Engineering target | Custom carrier board — see Appendix C-14 |

**Solutions**

AG-1 · HC-2 · SC-1 · EN-1 · WE-1 · GV-1

— most of A-series

## EAP-3: Vision AI Node

**Jetson Orin Nano · 40 TOPS · CUDA · Multi-camera · DeepStream**

- Multi-camera Vision
- High Performance Inference
- GPU Acceleration

| Design element | Specification |
|---|---|
| Processor | NVIDIA Jetson Orin Nano (40 TOPS) |
| AI framework | NVIDIA DeepStream · YOLO · TensorRT |
| Best for | Defect detection · ANPR · wildlife ID · face recognition |
| Power | 5–10W — mains or large solar |
| Connectivity | GigE Vision (camera) · 4G/5G · WiFi |
| POC board | Jetson Orin Nano Dev Kit (₹22,000–28,000) |
| Engineering target | Custom carrier board — thermal + camera interface |

**Solutions**

MF-3 (quality inspection) · SC-6 (ANPR) · GV-2 (border) · WE-5 (wildlife) · TR-5 (fatigue)

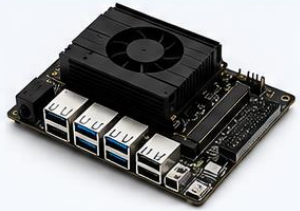
**Common across all EAPs:** NABL / BIS / ISO compliant hardware design · Secure boot & encrypted firmware update · Edge-to-Cloud integration · Modular, scalable & field upgradable

# Embedded AI Platform – Decision Guide

Choose the right platform for your application in 3 simple steps

START

Needs camera? — YES → **EAP-3** Jetson Orin Nano — 40 TOPS



- Best for vision AI, multi-camera, deep learning
- High performance GPU (CUDA)
- Use cases: ANPR, defect detection, wildlife monitoring, quality inspection

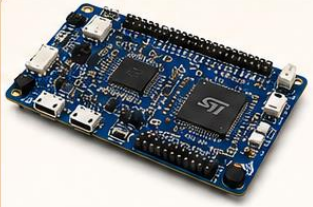
NO

Needs Linux? — YES → **EAP-2** STM32MP2 / RPi CM4 — Python + API

- Best for Linux, multi-sensor fusion, API, connectivity
- Runs Linux (Python, Docker, MQTT, API)
- Use cases: gateway, data aggregation, cloud integration, OTA, edge analytics

NO

Needs battery? — YES → **EAP-1** STM32N6 — TinyML — µA sleep



- Best for low power, real-time inference
- TinyML with Ethos-U55 NPU (4.8 TOPS)
- Use cases: vibration monitoring, audio classification, environment sensing, battery-powered nodes

**Quick Summary**

- Camera needed? → Choose EAP-3
- Need Linux / Python / API? → Choose EAP-2
- Need battery operation? → Choose EAP-1

# INTELLIGENT CONNECTED SYSTEMS

## Analogous to Intelligent Humans and Many Humans Collaborating in a Society

An Intelligent Connected System senses, thinks, acts and communicates. Many such systems connect and collaborate to create collective intelligence.

### 1. THE INDIVIDUAL – INTELLIGENT HUMAN

A single human is intelligent and can act independently.

**SENSES**
(Eyes, Ears, Skin, etc.)
Perceive the environment

**MIND**
(Brain)
Thinks, learns, decides and plans

**ACTIONS**
(Muscles & Limbs)
Executes actions in the real world

**ENERGY**
(Heart)
Provides and sustains energy

A human can sense, think, act and survive on their own.

### 2. THE INTELLIGENT SYSTEM – THE INDIVIDUAL AGENT

An intelligent system can operate autonomously.

**SENSORS**
Capture data from the environment

**EDGE AI**
(MCU/MPU/NPU + AI/ML)
Processes data locally, learns, decides and makes predictions

**ACTUATORS**
Executes actions through motors, valves, relays, etc.

**POWER SYSTEM**
Supplies and manages energy

Autonomous · Adaptive · Real-time · Reliable

An intelligent system senses, thinks, acts and operates independently.

### 3. THE COMMUNICATION CHANNEL

Enables systems to share information.

**CONNECTIVITY**
(Wired, Wireless, LPWAN, Cellular, Bluetooth, etc.)
Transmits data to and from other systems

**PROTOCOLS**
(Language & Rules)
Defines how systems communicate and understand each other

**GATEWAYS**
Bridges between local systems and remote networks / cloud

**SECURITY**
Ensures authentic, secure and reliable communication

The communication channel enables exchange of information and collaboration.

### 4. THE SOCIETY – COLLECTIVE INTELLIGENCE

Many intelligent systems collaborate to solve bigger problems and create greater impact.


Multiple systems connect, share, learn and make better decisions – together as a society.

### STANDALONE vs CONNECTED

**STANDALONE INTELLIGENT SYSTEM**
(Like a single individual)

- Operates independently
- Makes local decisions
- Works without connectivity
- Suitable for real-time and offline use

**CONNECTED INTELLIGENT SYSTEMS**
(Like people in a community)

- Share data and context
- Collaborate and coordinate
- Learn from collective data
- Create system-level insights

### EXAMPLE APPLICATIONS

| | | | |
|---|---|---|---|
| Smart Irrigation | Predictive Maintenance | Patient Monitoring | Traffic Management |
| Fleet Tracking | Smart Buildings | Environmental Monitoring | Energy Management |

### MAPPING TABLE: HUMAN ANALOGY vs INTELLIGENT CONNECTED SYSTEMS

| Human Analogy | Intelligent Connected Systems | Functional Role | Required for Standalone Operation? |
|---|---|---|---|
| Senses (Eyes, Ears, Skin) | Sensors | Observe the environment | Yes |
| Mind / Intelligence | Edge AI (MCU/MPU/NPU + AI/ML) | Think, learn, decide | Yes |
| Actions (Muscles & Limbs) | Actuators | Execute actions | Yes / Optional |
| Energy (Heart) | Power System | Supply and sustain energy | Yes |
| Communication (Voice, Language) | Connectivity (Wired / Wireless) | Share information | No |
| Society / Community | Multiple Systems & Cloud | Collective intelligence | No |

### KEY TAKEAWAY

An Intelligent Connected System is like an intelligent human – it can sense, think, act and survive on its own. Connectivity enables many such systems to collaborate and create collective intelligence – like a society.

### THE COMPLETE ECOSYSTEM

Individual Intelligent System (Autonomous) + Communication Channel (Share & Exchange) + Collective Intelligence (Analyze & Learn) = Smarter Society · Better Decisions · Greater Impact

### IN ONE LINE

From intelligent individuals to a connected society – Intelligent Connected Systems make the world smarter, safer and more sustainable.

# Three Ways to Add Intelligence to an Embedded System

Any embedded system that uses one of these three approaches to make intelligent decisions can be called an AI system. All three are valid engineering choices.

| | Name | What it means | Example students can build |
|---|---|---|---|
| 1 | Rule-Based AI | Engineer writes explicit mathematical or logical rules. No data needed. No training. | STM32 with Kalman Filter for drone stabilisation or Fuzzy Logic for motor control |
| 2 | Machine Learning Based AI | Machine learns patterns from structured sensor data. Engineer defines features. Model deployed to MCU. | STM32 detecting bearing faults from vibration and temperature feature vector |
| 3 | Deep Learning Based AI | Machine learns from raw unstructured input — images, audio. Network discovers its own features. | STM32 with Ethos NPU detecting wake word from microphone or defect from camera |

**The choice is driven by:**

**Nature of input**

*Structured numbers or raw unstructured signal?*

**Complexity of decision**

*Can a human write the rule or not?*

**Data available**

*No data / labeled data / abundant raw data?*

# The Hardware Reality — From Student Project to Industry

The same AI concepts apply across a spectrum of hardware — from the microcontroller your students already use to the edge AI modules used in industry. The progression is continuous.

| Hardware | What students know it as | AI role | AI framework |
|---|---|---|---|
| STM32 Cortex-M4 | Standard microcontroller lab hardware | TinyML inference — Rule-Based or ML Based AI | STM32Cube.AI — CMSIS-NN |
| STM32N6 Cortex-M55 + Ethos NPU | Next generation MCU with AI accelerator | TinyML + vision and audio inference | STM32Cube.AI with NPU support |
| ESP32-S3 | Popular IoT microcontroller | TinyML inference — keyword spotting, gesture | Edge Impulse — TFLite Micro |
| Raspberry Pi | Single board computer — Linux | Edge AI inference — larger models | TensorFlow Lite — ONNX Runtime |
| NVIDIA Jetson Orin | Edge AI module — industry standard | Edge AI — vision, LLM, continuous learning | TensorRT — PyTorch |
| Cloud GPU — workstation | Training infrastructure | Model training — never deployed to edge | PyTorch — TensorFlow |

A student who has programmed an STM32 in C is three steps away from deploying their first AI model on it:
***Collect sensor data → Train model on laptop → Flash INT8 model to STM32 via STM32Cube.AI***

# Machine Learning based AI Implementation Process — End to End

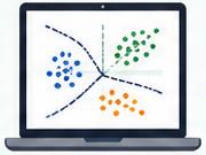
## 1 Instrument the System

Collect structured sensor data over time.

## 2 Label the Data


Label the data — Normal / Fault / Degraded or continuous measurement targets.

## 3 Select Algorithm

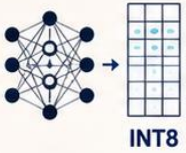


- Decision Tree
- Random Forest
- SVM
- Shallow Neural Network

## 4 Train on Workstation or Cloud

Train the model and validate accuracy against held-out data.

## 5 Quantize Model

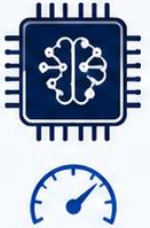



Compress the trained model to INT8 for efficient embedded deployment.

## 6 Deploy to MCU


Deploy via OTA, JTAG, DFU, or any supported update mechanism.

## 7 Inference Engine Runs on MCU



Decision made in microseconds on every new reading.

## 8 Retrain Periodically

As new field data is collected...

## 9 Retrain and Redeploy Improved Model

Use new field data to retrain, re-validate and deploy improved model continuously.

- Adaptive Improvement
- Higher Accuracy over Time
- Efficient Embedded Performance
- Lower Bandwidth & Compute Cost
- Secure & Reliable Deployment

# The Journey Every Engineer Must Understand

Every AI system — regardless of complexity — follows the same end-to-end journey. This is the mental model every engineering graduate must carry.

**RAW DATA → ALGORITHM → TRAINING → MODEL → INFERENCE ENGINE → DECISION**

| Step | Plain language meaning | Where it happens |
|---|---|---|
| Raw Data | Measurements from sensors — temperature, vibration, current, images, audio | Field — the physical system |
| Algorithm | The mathematical method that defines how the machine will learn from that data | Engineer's choice — workstation |
| Training | Running the algorithm on data until the machine learns accurately | Laptop or cloud — compute intensive |
| Model | The result — a file of learned numbers deployed to the embedded device | Travels from training to device |
| Inference Engine | The C program that reads the model and applies it to new sensor data | Runs on MCU — written in C |
| Decision | The intelligent output — classify, detect, predict, alert, control | Edge device — where action is needed |

**Training always happens where compute is abundant.**

*Inference always happens where the decision is needed — on the embedded system the student built.*

# What Every Engineering Graduate Must Be Able to Do

A graduate who has completed a well-designed AI in embedded systems module should be able to:

| Capability | What it looks like in practice |
|---|---|
| Define AI precisely | Distinguish Rule-Based AI from ML Based AI from Deep Learning Based AI — and explain when each is appropriate |
| Understand the end-to-end journey | Explain what training is, what a model is, what an inference engine is — and where each lives in a physical system |
| Deploy a model to an MCU | Take a trained model, quantize it, generate a C array, flash it to an STM32 or ESP32, and run inference |
| Choose the right approach | Given a sensor input and a decision requirement — identify which of the three AI approaches is appropriate |
| Use precise terminology | Never confuse training with inference, model with engine, learning with deployment — use terms correctly in context |
| Read and understand AI literature | Follow a technical paper, application note, or product brief about embedded AI without being lost in terminology |

*None of these require a mathematics or computer science background beyond what an engineering graduate already has.*

**They require a clear conceptual foundation, precise terminology, and hands-on deployment experience.**

# The Terminology Problem — Why It Matters in the Classroom

One of the most persistent barriers to AI adoption in engineering education is imprecise terminology. When foundational terms are used loosely — students build understanding on unstable ground.

| What is commonly said | What it actually means | Why the difference matters |
|---|---|---|
| "The AI is learning" | The device is running inference — not learning. Learning happened during training. | Student deploys wrong hardware — expects MCU to improve itself |
| "Machine Learning algorithm" | Could mean the learning method, the inference operator, or just any AI approach | Student cannot communicate precisely with industry colleagues |
| "Running AI on the device" | Almost always means frozen model inference only — no active learning | Student misunderstands what the device is actually doing |
| "The AI model" | In different contexts means the system, the weight file, or the quantized array | Student cannot read documentation or application notes correctly |
| "Edge AI chip" | A chip optimised for inference — not for training | Student selects wrong hardware for wrong phase of AI pipeline |

**Precise terminology is not pedantry. It is the foundation of flawless engineering.**

*A faculty member who teaches AI terminology precisely gives every student a professional advantage that lasts their entire career.*

# What This Resource Offers — And How to Use It

This resource is a complete, self-contained AI reference for engineers — built specifically for physical systems engineers, not for software or data science audiences.

| Component | Content | Time | Best used by |
|---|---|---|---|
| Main deck — 100 slides (AI-What Every Engineer must know about) | Foundations through deployment — AI, ML, DL, hardware, terminology. Short case studies on ML based Battery management system implementation, DL based key word spotting | 3 hrs / 1 hr foundations | Students with embedded background — self study or lecture |
| Rotating machinery case study | End to end ML — sensors to Phase 3 RUL — 9 slides + Word doc | 1 hr | Students with embedded background |
| This faculty overview — 10 slides | The pedagogical case — why, what, how | 10 minutes | Faculty — curriculum planning |

**Three entry points for an engineering faculty:**

→ **Never taught AI before:** *Start with this deck. Then Slides 1–13 of the main deck.*

→ **Teach embedded systems already:** *Main deck foundations — Three Approaches — and the case studies.*

→ **Want to integrate into existing course:** *Use the algorithm tables and inference engine slides as lecture inserts.*