IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

HIGH IMPACT PRACTICES [HIPS]

CORNERSTONE PROJECTS:

EMBEDDED SYSTEMS

INFORMATION PACKET

2025-2026

INSTITUTE OF AERONAUTICAL ENGINEERING
25 YEARS 2000 2025

I appreciate your interest in the Cornerstone Project (CoP), Department of EEE at the Institute of Aeronautical Engineering!

A cornerstone project (CoP) is typically introduced during the early or middle stages of an academic program at the Institute of Aeronautical Engineering. It focuses on helping students build foundational skills and understand how to apply basic concepts to real-world scenarios. These projects are usually smaller in scope, moderately complex, and designed to strengthen practical understanding of core subjects.

The Cornerstone Projects provide a platform for students to bridge the gap between classroom concepts and industry-relevant skills. By working on hands-on challenges such as Smart Sensor Calibration using Machine Learning students apply core EEE concepts like signal processing, sensor technology, embedded systems, and control theory in practical scenarios. These projects foster critical thinking, innovation, and interdisciplinary collaboration, enabling students to see the real-world impact of their theoretical knowledge. This not only enhances their academic understanding but also prepares them for future roles in research, development, and industry.

Cornerstone Project (CoP) teams are:

- Collaborative Project – This is an excellent opportunity for students who are committed to working towards social developments and emerging needs.
- Project Activity – The project coordinator listed current working areas for offering cornerstone projects with a team size of at least two students. The coordinator allotted mentors based on the work area and facilitated exclusive project laboratories for selected cornerstone project (CoP)students. This cornerstone project (CoP)bridges the gap between academic learning and real-world social applications. It helps enhance the professional development
- Short-term - Each undergraduate student may participate in a project for an assigned period.

The primary goal of Cornerstone Projects in the Department of Electrical and Electronics Engineering (EEE) is to integrate foundational engineering knowledge with practical, real-world problem-solving to foster innovation, critical thinking, and hands-on technical skills.

- Apply theoretical concepts from circuits, electronics, control systems, embedded systems, and signal processing.
- Encourage team-based design thinking and interdisciplinary collaboration.
- Promote awareness of sustainable and socially relevant solutions aligned with global challenges (such as the UN Sustainable Development Goals).
- Prepare students for industry, entrepreneurship, and advanced research through experiential learning.
- Strengthen skills in circuit design, embedded systems, signal processing, IoT, power systems, and control engineering.
- Encourage students to design original solutions to engineering problems using creative approaches and emerging technologies.
- Cultivate the ability to work effectively in interdisciplinary teams, reflecting real-world engineering environments.
- Provide opportunities to work with tools and platforms like MATLAB, Arduino, Raspberry Pi, LabVIEW, and simulation software.
- Align project outcomes with social, environmental, or community needs—often by mapping them to the **UN Sustainable Development Goals (SDGs)**.
- Inspire students to explore deeper concepts, conduct experiments, and pursue independent or faculty-guided research.
- Develop adaptability and curiosity that prepare students for emerging technologies and continuous learning.

The research theme of this AI based projects also focuses on the challenges presented by the Sustainable Development Goals (SDGs).

IARE Sustainability Development Goals (SDGs) highlighted with Blue Colour Font	
SDG #1	End poverty in all its forms everywhere
SDG #2	End hunger, achieve food security and improved nutrition and promote sustainable agriculture
SDG #3	Ensure healthy lives and promote well-being for all at all ages
SDG #4	Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all
SDG #5	Achieve gender equality and empower all women and girls
SDG #6	Ensure availability and sustainable management of water and sanitation for all
SDG #7	Ensure access to affordable, reliable, sustainable and modern energy for all
SDG #8	Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all
SDG #9	Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation
SDG #10	Reduce inequality within and among countries
SDG #11	Make cities and human settlements inclusive, safe, resilient and sustainable
SDG #12	Ensure sustainable consumption and production patterns
SDG #13	Take urgent action to combat climate change and its impacts
SDG #14	Conserve and sustainably use the oceans, seas and marine resources for sustainable development
SDG #15	Protect, restore and promote sustainable use of terrestrial ecosystems, sustainably manage forests, combat desertification, and halt and reverse land degradation and halt biodiversity loss
SDG #16	Promote peaceful and inclusive societies for sustainable development, provide access to justice for all and build effective, accountable and inclusive institutions at all levels
SDG #17	Strengthen the means of implementation and revitalize the Global Partnership for Sustainable Development

High Impact Practices (HIPs) – Power Electronics **Information Packet: 2025-26**

The following research domains are recommended for HIPs-Power Electronics Projects, and selected students should find the research gap and frame the problem statements from any one of the themes below**.**

1. Smart Sensor Calibration using Machine Learning **(SDG#3, SDG #9, SDG#11, SDG #12, SDG#13)**
2. Implement Voice Recognition Models on Microcontrollers to Control Electronic Systems **(SDG#3, SDG #4, SDG#7, SDG #9, SDG#11, SDG #12)**
3. Implement AI Models to detect and isolated hardware faults in real-time **(SDG#7, SDG #9, SDG#11, SDG #12, SDG #13)**
4. Smart Traffic Light Controller with Machine Learning **(SDG #9, SDG #12)**
5. Analyze Real-Time Biomedical Signal **(SDG#3, SDG #8, SDG#9, SDG #11, SDG #13)**
6. Use Embedded Cameras and Machine Learning Models for Defected Detection in Manufacturing **(SDG #8, SDG#9, SDG #11, SDG #12, SDG #13)**
7. Hardware Acceleration for ML **(SDG #4, SDG#7, SDG #9, SDG #12, SDG #13)**

In order to participate in Power Electronics Projects, you must formally apply and be accepted by the project coordinator. To proceed, please mail to the project coordinator, Dr. Damodhar Reddy, Professor and Head, Dept. of EEE, Email Id: dr.damodharreddy@iare.ac.in. This will bring up all available open positions tagged as Power Electronics projects. When submitting a project document and an updated résumé, include a statement regarding why you are interested in working with the team to which you are applying. Please note that participation by the power electrics project team requires registration for the accompanying research statement from any of the specified domains. More information will be provided to all selected Power Electronics project applicants who have been offered a position. If you have any questions about a particular team, please contact the team's faculty mentor(s). We encourage you to contemplate this fascinating new opportunity. We look forward to receiving your application submission.

Smart Sensor Calibration using Machine Learning

GOALS

The primary goal of smart sensor calibration using machine learning is to **automate the calibration process**, eliminating the need for manual intervention and traditional calibration techniques. Conventional sensor calibration methods are often static, time-consuming, and prone to errors—especially when deployed in dynamic environments or over long periods. Machine learning offers an intelligent alternative by learning complex relationships between raw sensor outputs and accurate reference values, allowing for real-time, adaptive calibration.

Another key objective is to **enhance the accuracy, stability, and reliability** of sensor data over time. Sensors in embedded systems are susceptible to noise, environmental conditions, signal drift, and hardware aging. Machine learning models can identify these patterns and apply correction factors dynamically, ensuring consistent and high-fidelity data for downstream systems. This is especially critical in applications such as healthcare, industrial automation, robotics, and IoT, where accurate sensor readings are essential for safety and performance.

The project also aims to **support real-time, on-device (edge) calibration**, using lightweight ML models that can run on microcontrollers or embedded processors. By deploying these models directly on hardware platforms like Arduino, ESP32, or STM32, the system becomes capable of **self-calibration at the edge**, without the need for internet connectivity or cloud-based computation. This drastically reduces latency and power consumption while improving the responsiveness and autonomy of the system.

METHODS & TECHNOLOGIES

- Smart sensor calibration using machine learning improves the accuracy and reliability of sensor outputs in real time.
- The core methodology involves, Collecting large sets of sensor data both raw and reference-calibrated. Training machine learning (ML) models to learn the mapping between inaccurate and corrected sensor values.
- This enables automatic correction of sensor readings during operation, especially under varying environmental or operational conditions.
- The process typically begins with data acquisition and preprocessing; Sensor readings are collected over a range of operating conditions. Data is compared with a trusted reference instrument.
- **Linear regression** and **polynomial regression** for simple, linear or mildly nonlinear sensor responses.
- **Artificial Neural Networks (ANNs)** and **Support Vector Machines (SVMs)** for modeling complex or nonlinear relationships.
- **Decision trees** and **random forest classifiers** for robust, interpretable anomaly correction and sensor state classification.
- **Autoencoders** and **unsupervised learning methods** for anomaly detection in unlabelled datasets.
- These tools enable integration of ML models into devices such as:
    - Arduino
    - ESP32
    - STM32
    - Raspberry Pi
- Edge AI tools allow real-time inference on the sensor node, enabling immediate sensor reading correction without cloud dependency.

MAJORS & AREAS OF INTEREST

Smart sensor calibration using machine learning is an interdisciplinary field that draws from a variety of engineering and computing disciplines. It lies at the intersection of electronics, embedded systems, artificial intelligence, and data science, making it relevant to multiple academic majors and areas of research.

- **ECE (Electronics and Communication Engineering) students and professionals** are well-suited for this topic because they have, Deep knowledge of sensor technology, signal conditioning, and embedded system design., Understanding of different sensor types (temperature, pressure, proximity, inertial).
- **EEE (Electrical and Electronics Engineering) students** benefit from
    - Strong foundation in instrumentation, control systems, and signal processing.
    - Skills for analyzing sensor output, managing noise and drift.
    - Expertise in analog and digital electronics, crucial for sensor-based systems.
- **CSE (Computer Science and Engineering) and AI/Data Science majors** contribute
    - Machine learning expertise to develop and train models for sensor data calibration.
    - Skills in data preprocessing, algorithm selection, model training, and deployment on edge devices.
- **Mechatronics and Robotics students** find this area relevant due to
    - Reliance on accurate sensor inputs for control, navigation, and environment interaction.
    - Importance of real-time AI calibration for safe and efficient robot operation despite sensor degradation or changing conditions.
- **Research and professional applications** of smart sensor calibration with ML include
    - IoT (Internet of Things)
    - Smart agriculture
    - Environmental monitoring
    - Industrial automation
    - Healthcare devices
    - Automotive systems
- The topic bridges
    - Hardware and software
    - Theory and application
    - Research and industry
- It offers valuable learning opportunities and career potential for students interested in building smarter, autonomous, and adaptive sensor systems using AI and embedded technologies.

Implement Voice Recognition Models on Microcontrollers to Control Electronic Systems

GOALS

The primary goal of this project is to develop a **voice-controlled embedded system** by implementing lightweight **voice recognition models** on microcontrollers. The system will recognize specific voice commands to control various electronic devices, enabling a smart, hands-free interface suitable for real-world applications such as home automation, assistive technology, and IoT.

Develop a model capable of recognizing a limited set of voice commands (e.g., "ON", "OFF", "FAN", "LIGHT") using speech datasets and machine learning techniques. Use TinyML or TensorFlow Lite to compress and optimize the model for deployment on microcontrollers with limited computational resources. Integrate the trained model onto a microcontroller (e.g., ESP32, Arduino Nano 33 BLE Sense) capable of real-time audio input and inference.

Connect the microcontroller to control circuits (e.g., relays, actuators, LEDs) to demonstrate voice-activated operation of electronic systems. Achieve reliable voice recognition with minimal latency and high accuracy in noisy or varying environments. Build a system that is energy-efficient and suitable for portable or battery-operated applications. Design the interface to be user-friendly, particularly beneficial for individuals with mobility challenges or in smart automation settings.

METHODS & TECHNOLOGIES

Model Training: Train a speech recognition model (e.g., keyword spotting) on a PC using datasets like Google Speech Commands.
Optimization: Quantize and convert the model using tools like TensorFlow Lite or Edge Impulse to fit microcontroller constraints.
Deployment: Flash the model onto a microcontroller (e.g., Arduino Nano 33 BLE Sense or ESP32).
System Integration: Interface the microcontroller with electronic systems via GPIOs.

Technologies Used:

- **Hardware**
    - Microcontrollers: ESP32, Arduino Nano 33 BLE Sense, Raspberry Pi Pico.
    - Peripherals: Microphone modules, relays, LEDs, actuators.
- **Software & Frameworks**
    - TinyML, TensorFlow Lite for Microcontrollers.
    - Edge Impulse Studio (for no-code model training and deployment).
    - Arduino IDE / PlatformIO.
    - Python (for preprocessing, training, and model testing).
- **Connectivity (if needed)**
    - Bluetooth / Wi-Fi for wireless command updates or logging.
    - MQTT or HTTP protocols for IoT integration.

MAJORS & AREAS OF INTEREST

- **Embedded Systems**

  Real-time signal processing and efficient memory usage on microcontrollers.
- **Edge AI / TinyML**

  Deploying ML models in resource-limited environments for real-world interaction.
- **Human-Computer Interaction (HCI)**

  Enhancing accessibility and usability through voice interfaces.
- **Speech Processing**

  Voice activity detection (VAD), feature extraction (MFCC), and keyword spotting (KWS).
- **Internet of Things (IoT)**

  Smart home/office automation with AI-enabled embedded devices.
- **Power Management**

  Designing low-power voice interfaces suitable for battery-operated systems.
- **Assistive Technology**

  Voice-activated systems for the elderly or people with disabilities.

Implement AI Models to detect and isolated hardware faults in real-time

GOALS

The primary goal is to design and implement AI models capable of detecting and isolating hardware faults in real-time within embedded or digital electronic systems. This system aims to improve reliability, reduce downtime, and enable predictive maintenance by identifying faults as soon as they occur, thereby minimizing damage and repair costs. The solution should work efficiently within resource-constrained environments, providing accurate and timely diagnostics with minimal false positives.

Develop AI-Based Fault Detection Algorithms: Design and train AI models capable of distinguishing between normal and faulty hardware states based on sensor data, electrical signals, or performance metrics. These algorithms should detect subtle anomalies and early signs of hardware degradation.

Implement Fault Isolation Mechanisms: Extend the detection models to not only identify that a fault has occurred but also isolate the precise hardware module or circuit responsible for the fault. This enhances diagnostic accuracy and accelerates repair actions.

Real-Time Processing and Low Latency: Ensure that the AI models operate in real-time on embedded platforms or edge devices, providing instant fault alerts and diagnostics with minimal processing delay to enable immediate response or corrective actions.

Optimize AI Models for Embedded Deployment: Compress and optimize AI models using TinyML or other model compression techniques to fit resource-constrained embedded processors or microcontrollers, maintaining high accuracy without excessive computational demand.

Enhance System Robustness Against Noise and Variability: Design fault detection systems resilient to environmental noise, sensor inaccuracies, and operational variability, ensuring reliable performance across different hardware conditions and operating environments.

Integrate with Existing Monitoring and Control Systems: Provide seamless integration of AI-based fault detection modules into existing hardware monitoring frameworks, supporting standard communication protocols and enabling automated or manual intervention.

Facilitate Predictive Maintenance and Fault Prevention: Use the insights gained from fault detection and isolation to predict impending failures, enabling scheduled maintenance before catastrophic breakdowns occur, thus improving equipment lifespan and reducing operational risks.

METHODS & TECHNOLOGIES

Sensors and IoT Devices: Deploy sensors to continuously monitor hardware parameters such as temperature, voltage, current, vibration, and error logs.
Data Acquisition Systems: Use high-speed data acquisition hardware and edge computing devices for real-time data capture.
Data Cleaning & Normalization: Apply filtering techniques (e.g., moving average, Kalman filters) to remove noise and normalize data to standard scales.
Feature Extraction: Extract relevant features such as statistical moments, frequency components (via FFT), wavelet transforms, or domain-specific indicators.
Model-based Diagnosis: Use AI models combined with system knowledge to isolate fault components by comparing predicted behaviour vs. actual sensor data.
Rule-based Systems & Expert Systems: Integrate domain-specific rules for narrowing down fault locations.
Explainability & Feature Importance: Apply techniques such as SHAP or LIME to interpret model outputs and help pinpoint fault origins.

MAJORS & AREAS OF INTEREST

Edge AI and Embedded Systems: Deploy lightweight AI models on edge devices or embedded processors for low-latency fault detection.

Stream Processing Frameworks: Use technologies like Apache Kafka, Apache Flink, or MQTT for real-time data ingestion and processing.

Cloud & Hybrid Architectures: Combine edge inference with cloud-based analytics for scalability and historical analysis.

APIs and Dashboards: Implement RESTful APIs and real-time visualization dashboards to notify maintenance teams instantly.

Cross-validation and Testing: Use k-fold cross-validation and real-world testbeds to evaluate model accuracy and robustness.

Online Learning & Adaptation: Incorporate continuous learning mechanisms to update models with new fault patterns.

Real-Time Data Acquisition and Monitoring

- Sensor technologies for hardware status monitoring
- High-frequency data capture and streaming
- IoT and edge device integration

Data Processing and Feature Engineering

- Noise reduction and signal filtering techniques
- Extraction of meaningful features from raw sensor data
- Handling imbalanced and noisy datasets

AI and Machine Learning Models for Fault Detection

- Supervised learning approaches using labelled fault data
- Unsupervised and anomaly detection models for unknown faults
- Time-series and sequence modelling for temporal fault patterns

Fault Isolation and Root Cause Analysis

- Diagnostic models for identifying fault locations
- Explainable AI to interpret model decisions
- Integration of expert systems and rule-based methods

System Deployment and Real-Time Inference

- Edge computing and embedded AI deployment
- Low-latency model inference and decision-making
- Stream processing frameworks and real-time alerting

Smart Traffic Light Controller with Machine Learning

GOALS

The primary goal of this project is to design and implement an intelligent traffic light control system that leverages machine learning to optimize traffic flow in real time. Unlike traditional traffic light systems that operate on fixed-time schedules, this smart controller will dynamically adjust signal timing based on real-time traffic conditions. By utilizing data from traffic sensors or cameras, the system will analyze vehicle density, queue lengths, and flow patterns to make informed decisions, thereby minimizing congestion and improving overall traffic efficiency.

Another key objective is to enhance road safety and reduce the environmental impact of traffic congestion. The intelligent controller will prioritize emergency vehicles and reduce unnecessary idling time, which contributes to lower emissions and fuel consumption. The system will also be designed to learn and adapt over time, improving its performance as more traffic data becomes available. This adaptive learning approach aims to provide scalable and sustainable traffic management solutions for both urban and suburban environments.

Additionally, the project aims to explore and evaluate different machine learning algorithms—such as reinforcement learning, decision trees, or deep neural networks—to determine which provides the most accurate and responsive traffic control. Emphasis will be placed on creating a cost-effective and easily deployable system that can be integrated with existing traffic infrastructure. Overall, the smart traffic light controller aspires to create smarter cities with smoother traffic flow, reduced travel times, and improved commuter experiences.

METHODS & TECHNOLOGIES

- **System Architecture Components**
    - Data Acquisition
    - Machine Learning-Based Decision Making
    - Signal Control Execution
- **Data Acquisition Technologies**
    - Sensors such as
        - Cameras
        - Inductive loops
        - Infrared sensors
        - Vehicle count
        - Vehicle speed
        - Congestion levels
- **Computer Vision Integration**
    - Use of tools and frameworks such as
        - OpenCV – for image processing and object detection
        - YOLO (You Only Look Once) – for real-time vehicle detection from video feeds

Machine Learning-Based Decision Making: Use machine learning algorithms to analyze traffic patterns and determine optimal traffic light sequences.

- **y Machine Learning Processes**
    - **Data Preprocessing**
    - **Feature Extraction**
    - **Model Training**

MAJORS & AREAS OF INTEREST

- **Computer Science & Engineering**: Focus on machine learning, artificial intelligence, data structures, and algorithm design.
- **Electronics & Communication Engineering:** Involvement in sensor integration, embedded systems, and microcontroller interfacing
- **Electrical Engineering**: Design and control of traffic light circuitry and power systems
- **Information Technology:** Data processing, system integration, and cloud/edge computing'
- **Artificial Intelligence & Data Science**: Development and training of machine learning models for traffic prediction and control.
- **Civil Engineering (Transportation Engineering)**: Understanding of traffic flow patterns, urban planning, and infrastructure layout'
- **Machine Learning & Artificial Intelligence**: Supervised learning, reinforcement learning, deep learning for dynamic decision-making
- **Computer Vision**: Real-time vehicle detection using OpenCV, YOLO, and image processing techniques
- **Embedded Systems**: Use of Raspberry Pi, Arduino, or other microcontrollers for hardware implementation
- **Internet of Things (IoT)**: Communication between sensors, traffic lights, and controllers via IoT protocols like MQTT
- **Smart Cities & Urban Mobility**: Integration of intelligent systems for efficient and sustainable urban traffic management
- **Simulation & Modelling**: Traffic flow simulation using tools like SUMO to test and optimize algorithms
- **Cloud and Edge Computing**: Real-time data processing and storage solutions for large-scale deployments
- **Cyber-Physical Systems**: Interaction between computational algorithms and physical traffic systems

Analyze Real-Time Biomedical Signal

GOALS

The primary goal of this project is to develop a system capable of analyzing biomedical signals, such as electrocardiograms (ECG), in real time. This involves the acquisition of physiological data using appropriate sensors and hardware interfaces that can capture signals with high fidelity and minimal latency. Real-time signal acquisition is crucial for timely diagnosis and immediate decision-making in critical healthcare scenarios.

Another key goal is to extract meaningful features from the pre-processed signals. For ECG, this might involve detecting PQRST waveforms, calculating heart rate, and identifying RR intervals. These features are essential for understanding physiological conditions and identifying abnormalities. Advanced methods, including time-domain, frequency-domain, and wavelet analysis, may be employed to extract and interpret signal characteristics effectively.

A major focus of the project is on the implementation of real-time pattern recognition and classification. By applying machine learning or deep learning algorithms, the system aims to automatically detect abnormalities such as arrhythmias or other critical events. The models will be trained and validated on real-time data streams or annotated datasets, enabling intelligent and adaptive health monitoring.

Visualization and user feedback are also integral goals. The system will offer real-time display of biomedical signals and key metrics through an intuitive interface, such as a desktop dashboard or mobile application. This will not only aid clinicians in monitoring patients but also provide alerts in case of abnormal readings, enabling prompt medical intervention.

METHODS & TECHNOLOGIES

To analyze real-time biomedical signals such as ECG, the first step involves acquiring the signals using appropriate hardware. Commonly used devices include
Biomedical Sensors: ECG electrodes, EMG patches, EEG caps, etc.
Acquisition Boards: Arduino with ADC modules, Raspberry Pi, or commercial devices like BIOPAC or NI DAQ systems.
Wearables: Smartwatches, fitness bands, or chest straps with built-in biosensors.
Real-time systems require low-latency and reliable data transmission. Technologies used include
Bluetooth Low Energy (BLE) – Common in wearables for short-range wireless communication.
Wi-Fi / MQTT Protocols – For higher-bandwidth and long-range data streaming to cloud or local servers.
Serial/USB Communication – For wired, low-delay transfer in prototyping or clinical settings.
Preprocessing is essential to remove noise and artifacts from raw biomedical signals. Common methods include

Machine Learning & AI

To analyze patterns and detect abnormalities, machine learning and deep learning models are deployed

- **Classical ML**: SVM, Random Forest, k-NN for signal classification (e.g., normal vs arrhythmia).
- **Deep Learning**: CNNs and LSTMs for automatic feature learning and high-accuracy classification.
- **Libraries/Frameworks**: TensorFlow, PyTorch, Scikit-learn.

MAJORS & AREAS OF INTEREST

Biomedical Engineering

This is the primary field of study for analyzing biomedical signals. It combines engineering principles with medical sciences to develop technologies for healthcare. Students and professionals in biomedical engineering focus on

- Signal acquisition and sensor design
- Physiological signal interpretation (e.g., ECG, EEG, EMG)
- Biomedical instrumentation and device development

Electronics and Communication Engineering (ECE)

ECE is crucial for the hardware and signal processing aspects of real-time biomedical signal analysis. Areas of interest include

- Analog and digital signal processing
- Embedded systems and microcontroller programming
- Communication protocols for medical data transmission

Computer Science & Engineering (CSE) / Information Technology (IT)

The computational aspect of biomedical signal analysis heavily involves CSE and IT disciplines. Key areas of interest include

- Machine learning and artificial intelligence for health data interpretation
- Real-time data streaming and cloud computing
- Software development for mobile health apps and dashboards

Data Science & Artificial Intelligence

This interdisciplinary field is becoming increasingly important in biomedical signal analysis. Data scientists focus on

- Big data analytics for healthcare
- Predictive modelling and classification of physiological conditions
- Deep learning for automatic feature extraction from ECG and EEG signals

Electrical Engineering

Electrical engineering contributes to the understanding of bio-signal behavior in terms of voltage, current, and noise. Key interests include

- Circuit design for bio-signal amplifiers
- Filtering techniques and analog-to-digital conversion
- Low-power system design for wearable health monitors

Neuroscience and Physiology

For those studying signals like EEG or EMG, a strong foundation in biological systems is important. Areas of interest include

- Neural signal processing
- Brain-computer interface (BCI) design
- Muscle and nerve activity analysis

Use Embedded Cameras and Machine Learning Models for Defected Detection in Manufacturing

GOALS

The primary goal of this project is to leverage embedded cameras and machine learning models to automate and enhance defect detection in manufacturing processes. By integrating high-resolution imaging with real-time data processing, the system aims to identify and classify defects with high accuracy and speed, minimizing human error and reducing inspection time. The objective is to improve product quality, increase operational efficiency, and enable predictive maintenance by detecting anomalies early in the production line. Ultimately, this approach seeks to create a scalable and cost-effective solution that ensures consistent manufacturing standards and supports continuous improvement in quality control systems.

A key focus is to enhance the accuracy and consistency of defect detection. Machine learning models will be trained using image data that includes both defective and non-defective items. This enables the system to identify even minor anomalies that may be missed by the human eye, resulting in precise and uniform quality assessments across large volumes of manufactured products.

To further improve manufacturing efficiency, the system will support real-time monitoring and decision-making. Embedded cameras placed at strategic points along the production line will capture images continuously. These images will be processed instantly using edge-computing or on-device machine learning models, allowing for immediate detection of defects and quick response to issues as they arise.

The automation of inspection tasks is expected to significantly reduce operational costs and inspection time. By minimizing the need for manual labour and expediting the detection process, manufacturers can achieve faster throughput without sacrificing product quality. This is particularly valuable in high-volume manufacturing settings where efficiency and speed are critical.

METHODS & TECHNOLOGIES

Embedded Cameras: High-resolution industrial-grade cameras will be deployed at key points on the production line to capture product images in real time.
Edge Devices: Devices like **NVIDIA Jetson Nano/Orin**, **Raspberry Pi with Coral Edge TPU**, or **Intel Neural Compute Stick** will be used for on-device ML inference, enabling low-latency decision-making.
Illumination Control: Proper lighting (e.g., LED ring lights, diffused lighting) ensures consistent image quality and defect visibility.
Image Preprocessing: Includes operations like grayscale conversion, noise reduction (Gaussian blur), edge detection (Sobel/Canny), normalization, and contrast enhancement to prepare input data for the model.
Data Annotation Images of defective and defect-free products are labelled using tools like Labelling or CVAT to create training datasets.
Model Training: Supervised learning methods, primarily using **Convolutional Neural Networks (CNNs)**, are applied for image classification, object detection, or segmentation, depending on the defect type.
Transfer Learning: Pre-trained models such as **ResNet**, **MobileNet**, **YOLOv5**, or **EfficientNet** may be fine-tuned on the manufacturing dataset to accelerate training and improve performance.
Anomaly Detection: For unseen defects, unsupervised or semi-supervised methods like **Autoencoders** or **One-Class SVMs** can be used to detect deviations from the norm.
TensorRT (for NVIDIA platforms) or **OpenVINO** (for Intel) is used to optimize ML models for deployment on edge devices.
ONNX format ensures model interoperability across different platforms.
Real-time inference is triggered by image capture events, with classification results returned in milliseconds for immediate defect flagging.

MAJORS & AREAS OF INTEREST

This interdisciplinary project lies at the intersection of several key engineering and technology domains. The successful implementation of defect detection using embedded cameras and machine learning involves expertise across the following majors and areas of interest

Machine Learning & Deep Learning: Focus on training models (e.g., CNNs, YOLO) for image classification, object detection, and anomaly recognition.

Artificial Intelligence (AI): Broader interest in intelligent systems capable of mimicking human decision-making in real-time inspection tasks.

Data Science & Analytics: Includes data collection, preprocessing, annotation, model evaluation, and performance metrics analysis.

Embedded Systems Design: Development and deployment of edge-based systems using microcontrollers, SoCs (like NVIDIA Jetson or Raspberry Pi), and integration of camera modules.

Digital Image Processing: Application of filters, transformations, edge detection, and preprocessing techniques for optimal image input to ML models.

Sensor Integration: Working with cameras, lighting systems, and other vision-related sensors for accurate image acquisition.

Industrial Automation: Understanding the automation of inspection processes and integration of vision systems with production lines.

Quality Control & Assurance: Application of engineering standards to maintain and improve product quality through automated defect detection.

Production Line Optimization: Use of data from defect detection systems to improve process reliability, efficiency, and throughput.

Vision-Based Inspection Systems: Use of computer vision in robotics for real-time defect inspection and alignment with robotic arms or conveyors.

System Integration: Combining mechanical, electrical, and software components into a unified system that can interact with the production environment.

Software Development: Building control systems, user interfaces, and dashboards for monitoring defect detection outputs.

Edge Computing & IoT: Leveraging edge devices for real-time processing and IoT protocols (like MQTT or HTTP) for communication with factory systems.

Hardware Acceleration for ML

GOALS

The primary goal of implementing hardware acceleration for machine learning (ML) is to significantly enhance the speed, efficiency, and performance of ML model execution on embedded and high-performance computing platforms. Traditional processors, such as general-purpose CPUs, are not optimized for the parallel processing demands of modern ML algorithms, particularly those involving deep neural networks. Hardware accelerators—such as Graphics Processing Units (GPUs), Field Programmable Gate Arrays (FPGAs), Application-Specific Integrated Circuits (ASICs), and Tensor Processing Units (TPUs)—are designed to execute ML workloads more efficiently by enabling parallelism and reducing computation time.

Another critical goal is to enable real-time inference on edge devices without relying heavily on cloud-based processing. This is particularly important in latency-sensitive applications such as autonomous vehicles, medical diagnostics, industrial automation, and smart surveillance, where immediate decisions based on sensor or visual input are crucial. By using hardware acceleration, embedded systems can achieve faster response times while maintaining energy efficiency.

Additionally, hardware acceleration supports the deployment of complex ML models in resource-constrained environments by reducing power consumption and memory usage. This makes it feasible to integrate intelligent features into small, portable, or battery-powered devices—contributing to the growth of TinyML and AIoT (Artificial Intelligence of Things). Finally, the goal is to make ML more accessible and scalable across industries by creating cost-effective and flexible acceleration solutions that can be customized for various application needs and hardware configurations.

METHODS & TECHNOLOGIES

Hardware acceleration for machine learning leverages specialized hardware components and optimized techniques to boost the performance, efficiency, and scalability of ML workloads. The goal is to execute ML algorithms—especially deep learning models—faster and with lower power consumption than general-purpose CPUs. Below are the key methods and technologies involved:

Technology: GPUs are designed for parallel processing and can handle thousands of threads simultaneously.
Application: Widely used for training and inference in deep learning due to their ability to process large matrix operations efficiently.
Frameworks: Compatible with ML libraries like TensorFlow, PyTorch, and CUDA (NVIDIA's parallel computing platform).
Technology: Developed by Google specifically for ML workloads, TPUs are application-specific integrated circuits (ASICs) optimized for tensor operations.
Application: Ideal for large-scale training and inference in cloud-based ML platforms such as Google Cloud ML Engine.
Quantization & Pruning: Reduce model size and computational complexity while maintaining accuracy.
Model Compilation: Tools like TensorRT (NVIDIA), TVM, and ONNX Runtime optimize trained models for specific hardware.
Parallelism & Scheduling: Efficient use of hardware threads and task scheduling to maximize resource utilization.
TensorFlow Lite / PyTorch Mobile: Lightweight versions of popular frameworks for deploying ML models on edge devices.
ONNX (Open Neural Network Exchange): A unified format that allows models trained in one framework to run efficiently on multiple hardware platforms.
Microcontroller Support: AI-specific microcontrollers (e.g., ARM Cortex-M with ML capabilities) allow for TinyML applications.

MAJORS & AREAS OF INTEREST

Hardware acceleration for machine learning is a multidisciplinary field that sits at the intersection of electronics, computer science, artificial intelligence, and systems engineering. Its development and application involve various academic majors and technical domains, each contributing essential knowledge and expertise.

Machine Learning & Deep Learning: Core algorithms that require acceleration for real-time inference and efficient training.

Parallel and Distributed Computing: Focused on optimizing ML workloads across GPUs, TPUs, and multi-core systems.

AI Systems Design: Development of scalable, performance-oriented ML architectures that can run on specialized hardware.

Digital Electronics & VLSI Design: Essential for designing and understanding hardware accelerators like ASICs, FPGAs, and microcontrollers.

Embedded Systems: Implementation of ML models on edge devices such as Jetson Nano, Raspberry Pi, or custom embedded boards.

Signal Processing: Preprocessing and transforming input data before feeding it to hardware-accelerated ML models.

Hardware Architecture Design: Creating efficient computation units optimized for specific ML operations (e.g., matrix multiplication, convolution).

Power Systems & Energy Efficiency: Ensuring low-power, high-performance designs for mobile and embedded ML applications.

Control Systems Integration: Using AI-powered embedded systems in feedback loops and real-time control applications.

Software-Hardware Co-Design: Developing optimized software that takes full advantage of underlying hardware capabilities.

ML Framework Integration: Adapting TensorFlow Lite, PyTorch Mobile, or ONNX for deployment on accelerators.

Cloud & Edge Deployment: Implementing ML workflows from model training in the cloud to inference on the edge.

Edge AI Applications: Real-time AI decision-making in robotics, drones, autonomous systems, and smart devices.

AI-Powered Automation: Use of accelerated ML in intelligent control and industrial automation systems.