Chapter I: Introduction to Embedded Electronics

I Introduction

I.1 Context of Embedded Electronics

Embedded electronics, a cornerstone of modern systems, addresses major industrial challenges by integrating three key components: sensors, microcontrollers or processors, and actuators.

Challenge: In industrial environments, the main objectives are to optimize operations, reduce maintenance costs, and improve productivity. Embedded electronic systems play a crucial role in achieving these goals by enabling intelligent control, real-time monitoring, and autonomous decision-making across machines and processes.

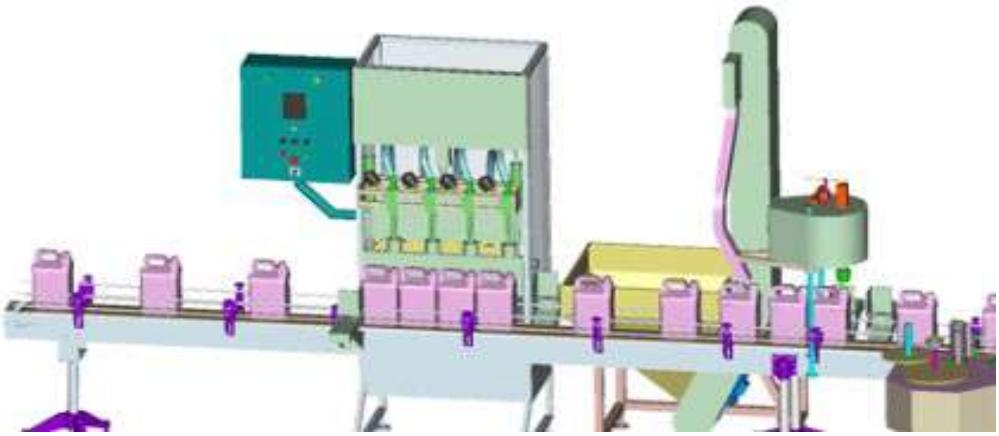

Figure I. 1 - Example of an Automated Production Line Using Embedded Electronics

Proposed Solutions:

I.1.1 Solution Based on Combinational Logic:

Advantages: Conceptual simplicity and low initial cost.

Limitations: Lack of flexibility for complex tasks and difficulty in adapting to system modifications or evolving functional requirements.

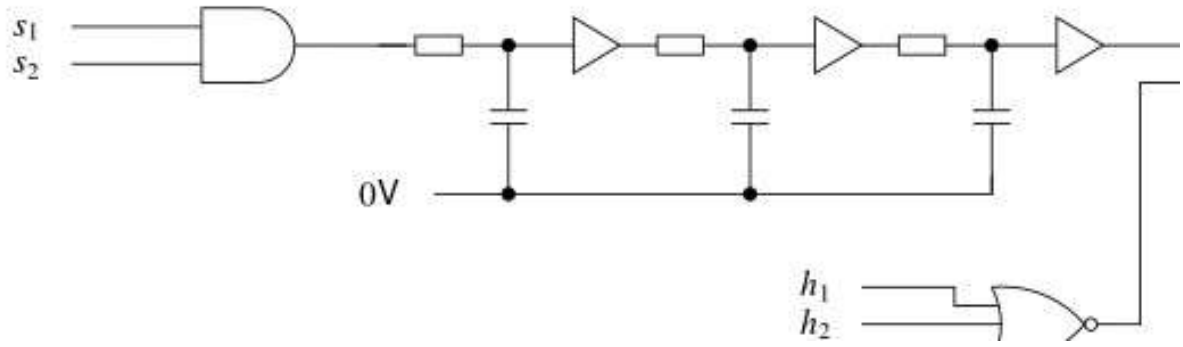

Figure I. 2 - Combinational Logic Diagram

I.1.2 Solution Based on Traditional Electronics:

Advantages: High adaptability to changing requirements and efficient management of complex systems.

Limitations: Increased costs due to circuit complexity, the need for advanced technical expertise, and hardware constraints that may limit scalability or integration.

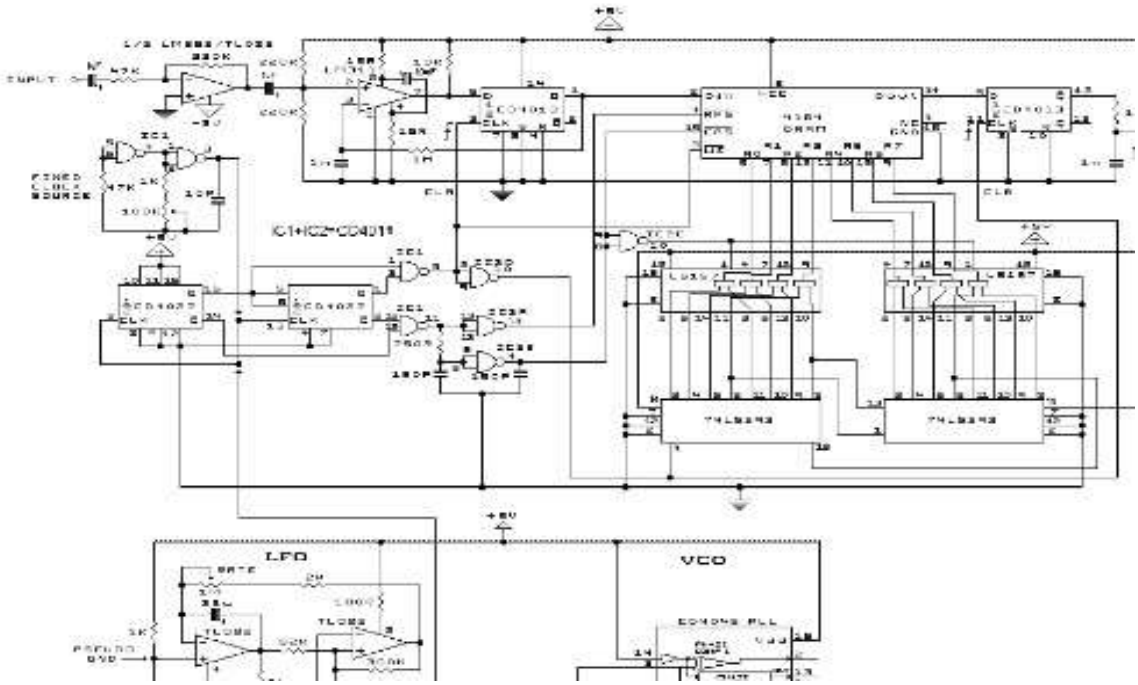

Figure I. 3 - Traditional Electronics Model

I.1.3 Solution Based on Embedded Systems:

Advantages: High flexibility, capability to manage complex tasks, and connectivity enabling remote monitoring and control.

Limitations: Higher initial cost and reliance on specialized technical expertise.

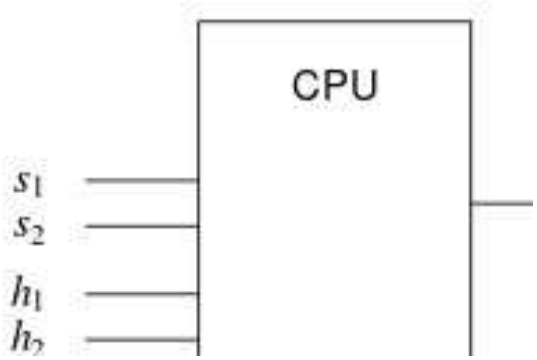

Figure I. 4 - Architecture of an Embedded System

Overcoming Limitations through the Transition from Hardware to Software (Embedded Systems):

- **Initial Costs:** Although the upfront investment in embedded systems may be higher, it is offset by reduced maintenance and operational costs in the long term.
- **Complexity:** Embedded systems offer programmability that allows for more flexible and scalable task management, reducing operational complexity.
- **Dependence on Expertise:** Thanks to user-friendly programming interfaces and advanced development environments, embedded systems have become more accessible, lowering the need for deep hardware expertise.
- **Flexibility:** The programmability of embedded systems enables rapid adaptation to industrial process changes, providing superior flexibility compared to fixed hardware solutions.
- **Connectivity and Remote Monitoring:** Embedded systems support networking capabilities, enabling remote supervision, centralized control, and seamless integration with other digital systems.

The shift toward embedded systems thus represents a transition to a more intelligent, adaptable, and software-oriented approach, where programmability and connectivity effectively overcome the traditional limitations of purely hardware-based designs.

I.2 Projects Developed at Our School

I.2.1 Project 1: Automated Attendance Management System

This project demonstrates a practical application of RFID (Radio Frequency Identification) technology in the educational domain. It involves the design and implementation of an automated attendance management system aimed at accurately monitoring student attendance within academic institutions. The main objective is to facilitate the identification of absentee students and provide reliable indicators of attendance for teachers and administrators.

The system architecture is composed of three integrated components:

1. **Electronic Component:** Each student carries an RFID card. Upon entering the classroom, the card is detected by an RFID reader connected to an Arduino UNO microcontroller. The reader captures the radio frequency signal emitted by the card and sends the student's ID to the Arduino for preliminary processing, such as activation and attendance recording.
2. **Telecommunication Component**: The attendance data collected by the Arduino are transmitted to a central server via an Ethernet module. This network link acts as the communication bridge between the physical hardware and the information system.
3. **Software Component**: The software part includes a server, a database, and a web application. The database stores student, teacher, course, and schedule data. The web interface—developed using PHP/MySQL and HTML/CSS—enables real-time attendance tracking, report generation, and statistical analysis. It provides multi-platform access from computers, tablets, and smartphones.

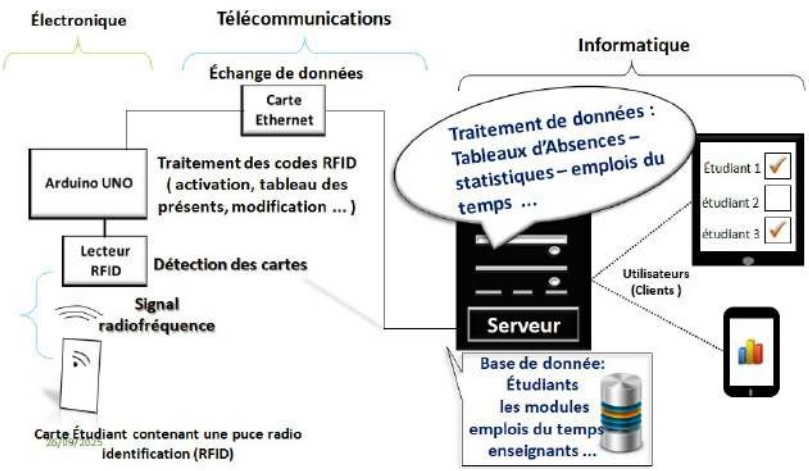

Figure I. 5 - General architecture of the automated attendance management system based on RFID technology.

This figure illustrates how the RFID card is detected by the reader connected to the Arduino, the data are then transmitted through the Ethernet module to the server, which processes and stores the information in the database. The web application allows teachers and administrators to visualize attendance in real time and generate statistical reports. The diagram highlights the synergy between electronics, telecommunications, and computer science in developing a reliable embedded system.

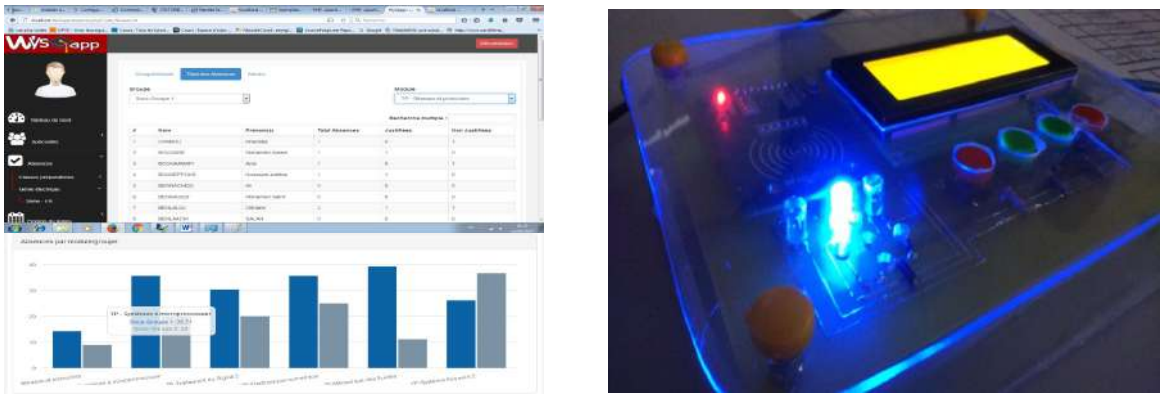

Figure I. 6 - Developed hardware prototype and associated software interface for attendance management and monitoring.

The prototype demonstrates the physical implementation using Arduino and the RFID reader, while the web interface showcases the available functionalities for attendance tracking, schedule visualization, and statistical reporting. Together, they validate the feasibility and practical relevance of the proposed system in real-world educational settings.

This work was presented under the title: “Design and Implementation of an Embedded System for Effective Attendance Management”, at the NewMat’21 – 1st International Conference: New Trends on Innovative Construction Materials, held at ESSA Tlemcen (Algeria), on March 22–23, 2022.

I.2.2 Project 2: Development of a 3D Printer – Version 1

The second project conducted as part of the embedded systems course focused on the design and implementation of a first-version 3D printer. The idea emerged from the need to produce custom plastic components for scientific projects while significantly reducing the high costs associated with commercial 3D printing services. As engineering students, the

proposed solution was to build the printer from scratch using readily available components and open-source software.

The main objective was to develop a fully functional 3D printer capable of manufacturing plastic prototypes using materials such as PLA and ABS. The project provided an opportunity to explore multiple embedded system concepts, particularly:

- **Motor control**: precise positioning through stepper motors for the X, Y, and Z axes.
- **Thermal regulation:** control of the heating extruder and heated bed.
- **System interfacing**: communication between the microcontroller and the user interface for process monitoring and control.

At the core of the system lies an Arduino Mega microcontroller equipped with a RAMPS 1.4 shield and stepper motor drivers to manage movement. Limit switches were integrated to calibrate the axes and ensure accurate positioning.

On the software side, the Marlin firmware was installed on the Arduino to interpret G-code commands generated by slicing software such as Cura or Slic3r. This allows the user to go seamlessly from a 3D model designed on a computer to a physical printed object, with real-time monitoring and control of printing parameters through a PC interface.

The prototype, shown in Figure 7, includes the mechanical structure, the electronic control board, and the power modules. This initial version successfully achieved the printing of small test objects, confirming the technical feasibility of the system. The project also demonstrated the pedagogical value of 3D printing, as it enables rapid prototyping—turning ideas into tangible models—while reducing manpower and tooling requirements.

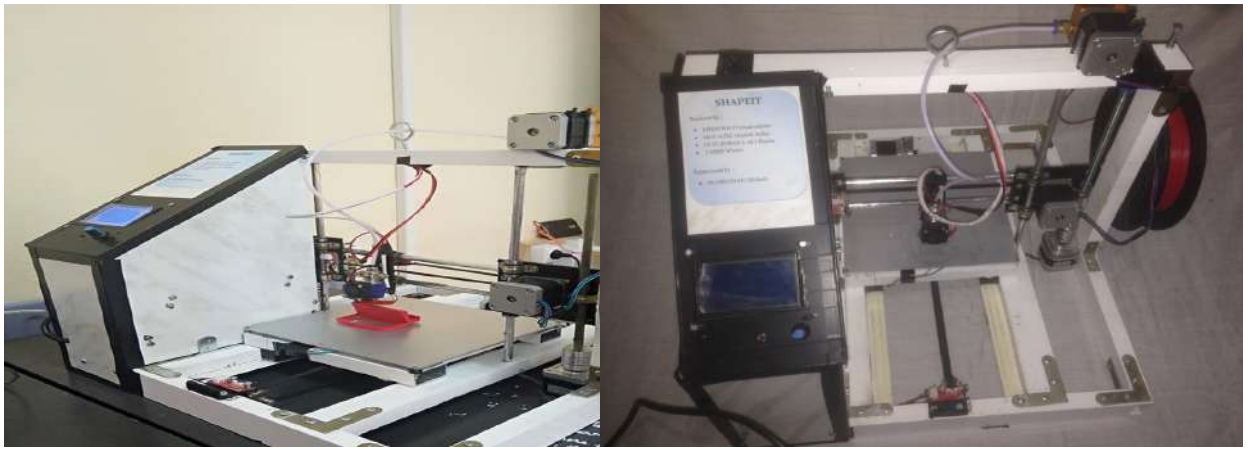

Figure I. 7 - Prototype of the 3D printer (Version 1) developed by engineering students, showing the mechanical frame, control electronics, and software interface.

This first version represents a foundational step in the development of a 3D printer. Future improvements could focus on increasing print speed, enhancing print quality, and improving overall reliability. Beyond its technical aspects, this project exemplifies the integration of mechanical, electronic, and software components within a complex embedded system. It also highlights the interdisciplinary nature of embedded engineering, combining control theory, automation, and digital manufacturing.

This work was presented in the following publication: H. Megnafi, O. Ayad, W. Tabib, A. A. Mouaziz, R. Ould Babaali, I. Medjhoud, "Improved Printing Time by Changing the Mechanical Part of the 3D Printer, Embedded System Application," NewMat'21 – 1st International Conference: New Trends on Innovative Construction Materials, ESSA-Tlemcen (Algeria), March 22–23, 2022.

I.2.3 Project 3: Development of a 3D Printer – Version 2

The second version of the 3D printer was designed with the goal of improving the performance achieved with the initial prototype. This enhanced version integrates a more efficient mechanical architecture, resulting in a threefold increase in printing speed and greater precision in the fabrication of printed parts.

The control system is built around an Arduino Mega 2560 microcontroller combined with the RAMPS 1.4 shield, which serves as the interface between the controller and the actuators. The printer's movement along the X, Y, and Z axes is handled by NEMA 17 stepper motors, driven by A4988 drivers that regulate the current and ensure precise motion control.

On the mechanical side, the structure was redesigned to minimize vibration and improve rigidity. A heated bed was added to enhance the adhesion of printed parts during the initial layers, while the hotend nozzles reach temperatures suitable for a variety of materials such as PLA and ABS. Limit switches were also integrated to establish reference positions for each axis, allowing for automatic calibration.

From an interface and autonomy perspective, a LCD screen with an SD card reader was implemented, enabling standalone operation without the need for a computer connection during printing. A 12V – 20A power supply provides sufficient energy for the heated bed, motors, and extruder heaters. The system operates under the Marlin firmware, configured to support dual-extruder management and optimized axis movement. The G-code files, generated through slicing software such as Cura or Repetier-Host, are directly interpreted by the Arduino, which controls the full printing sequence autonomously.

These hardware and software upgrades significantly reduced printing time while enhancing surface quality, consistency, and precision. This project demonstrates how refining both mechanical and electronic architectures can dramatically improve the performance of a complex embedded system such as a 3D printer.

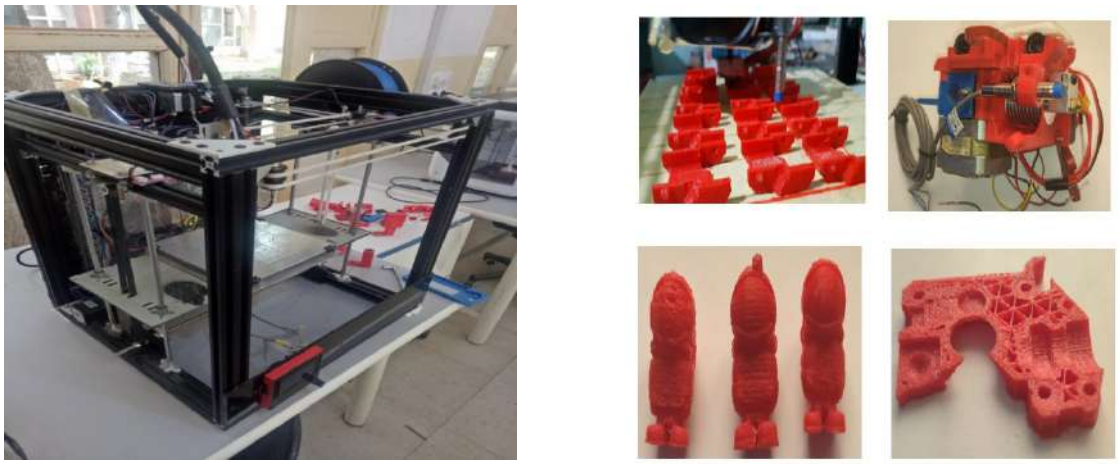

Figure I. 8 - 3D Printer – Version 2.

This work was presented in the following publication: H. Megnafi, O. Ayad, W. Tabib, A. A. Mouaziz, R. Ould Babaali, I. Medjhoud, "Improved Printing Time by Changing the Mechanical Part of the 3D Printer, Embedded System Application," NewMat'21 – 1st International Conference: New Trends on Innovative Construction Materials, ESSA-Tlemcen (Algeria), March 22–23, 2022.

I.2.4 Project 4: Design and Implementation of an Autonomous Watering System Based on PIC18F452

This project focuses on the design and realization of an autonomous irrigation system aimed at optimizing water management for agricultural and gardening applications. The system is entirely self-powered, operating on solar energy through a solar panel coupled with a rechargeable battery, thus ensuring complete energy autonomy. The PIC18F452 microcontroller serves as the central processing unit, managing data acquisition, decision-making, and the control of various sensors and actuators.

The system integrates soil moisture sensors and an LM35 temperature sensor to continuously monitor environmental and soil conditions. Based on predefined thresholds, the microcontroller automatically controls electrovalves to activate watering when necessary. This automated process ensures that plants receive the right amount of water while preventing waste, contributing to more efficient and sustainable irrigation management.

The user interface includes a 16×2 LCD display and push buttons, allowing users to select operation modes (automatic or manual) and configure system parameters. Furthermore, a Wi-Fi communication module enables remote supervision and control, allowing the user to monitor environmental data and system status via a smartphone or computer.

Figure I. 9 - Block diagram of the autonomous watering system.

The block diagram illustrates the main components of the system: the solar energy unit (panel and battery) supplies power to the control circuit; the sensors provide environmental data to the PIC18F452, which processes the inputs and drives the electrovalves accordingly. The LCD display and Wi-Fi module facilitate local and remote interaction, respectively.

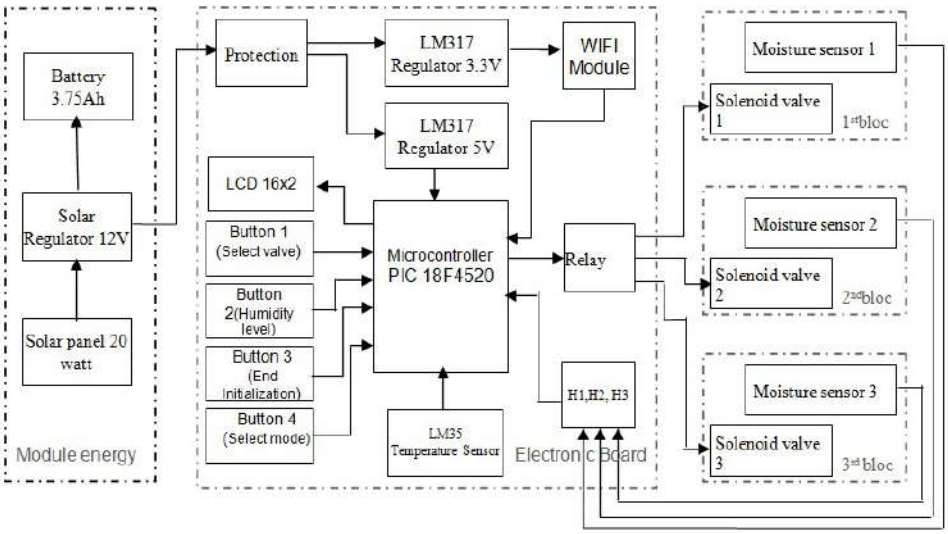

Figure I. 10 - Prototype of the autonomous watering system.

The developed prototype demonstrates the integration of hardware and software subsystems in a compact, functional design. It validates the feasibility of an energy-autonomous, intelligent irrigation controller capable of real-time adaptation to environmental changes.

This project exemplifies how embedded systems can be leveraged to address sustainability challenges by combining renewable energy, automation, and intelligent control.

This work was presented in the following publications:

- A. Chellal, H. Megnafi, A. Benhanifia, Design and Conception of an Autonomous Watering System, Garden (a Multi-Application Watering System), NewMat'21 – 1st International Conference: New Trends on Innovative Construction Materials, ESSA-Tlemcen (Algeria), March 22–23, 2022.
- H. Megnafi, A. A. Chellal, A. Benhanifia, *Flexible and Automated Watering System Using Solar Energy*, in *Artificial Intelligence and Renewables Towards an Energy Transition 4*, Springer International Publishing, 2021, pp. 747–755.

I.2.5 Project 5: Development of an Intelligent Urban Lighting System

This project focuses on the design and implementation of an IoT-based smart public lighting system aimed at significantly reducing energy consumption while improving operational efficiency and sustainability in urban environments. The developed prototype utilizes an ESP8266 NodeMCU microcontroller, which manages and coordinates the operation of the various modules within the system.

The system's power supply is provided by a photovoltaic solar panel, connected to a voltage regulator and a charging circuit that stores energy in a lithium battery, ensuring complete energy autonomy. This solar-based design not only supports renewable energy use but also enables continuous operation even in areas without access to the electrical grid.

A PIR (Passive Infrared) motion sensor is integrated for presence detection, allowing the system to automatically adjust lighting intensity based on pedestrian or vehicle movement. When no movement is detected, the light operates at a reduced intensity to conserve energy; it brightens immediately upon motion detection. A 16×2 LCD display provides real-time system status and data visualization, while a high-efficiency LED module is used for illumination.

The ESP8266 microcontroller, equipped with built-in Wi-Fi capability, ensures IoT connectivity and remote supervision. This allows monitoring and control through an online dashboard, enabling urban management authorities to track energy consumption, system performance, and maintenance needs in real time.

This smart lighting architecture integrates renewable energy, intelligent control, and wireless communication, resulting in a system that is energy-efficient, autonomous, and well-suited for smart city applications.

Figure I. 11 - Simplified schematic of the intelligent urban lighting system and the developed prototype.

The figure illustrates the system's functional structure, showing the interaction between the solar power subsystem, the control and sensing units (ESP8266 and PIR sensor), and the LED lighting module. It also depicts how the IoT layer enables remote communication and system monitoring.

This work was presented in the following publication: Imen Souhila Bousmaha, Hicham Megnafi, Hamza Benhadouga, Imene Hemarid, IoT Applications in Smart Public Lighting Management, The International Conference on Applied Science and Engineering (ICASE-22).

I.3 Fundamentals of Microcontrollers and Microprocessors

I.3.1 Role and Importance of Microprocessors

The microprocessor is the central processing unit (CPU) of a computing system, designed to execute general-purpose tasks. It operates based on either a Von Neumann or Harvard architecture and is composed mainly of three essential elements: the Arithmetic and Logic Unit (ALU), the Control Unit (CU), and a set of internal registers. These components work together to process instructions stored in memory, enabling the execution of complex computational sequences.

The microprocessor functions by fetching, decoding, and executing instructions, managing multiple tasks efficiently through the coordination of its control unit and the support of a multitasking operating system. Modern microprocessors integrate advanced architectural concepts such as multi-core structures, instruction pipelining, and hierarchical cache memory systems (L1, L2, L3) to optimize performance and reduce latency. Many also include specialized processing units, such as SIMD (Single Instruction, Multiple Data) and integrated GPUs, to accelerate parallel data processing and graphical computations.

The importance of microprocessors lies in their high computational capability and flexibility. They constitute the core of a board spectrum of applications, ranging from personal computing systems and servers to large-scale data centers and embedded systems that demand substantial processing capabilities. In the context of embedded electronics, microprocessors are employed in advanced systems such as autonomous vehicles, robotics, industrial control systems, and smart IoT devices, where they handle real-time signal processing, machine learning algorithms, and complex control operations.

In summary, the microprocessor represents the computational intelligence of modern digital and embedded systems, combining speed, precision, and adaptability to meet the demands of high-performance applications.

I.3.2 Role and Importance of Microcontrollers

A microcontroller is an integrated solution specifically designed to perform control and monitoring functions within embedded systems. Unlike general-purpose processors, it brings together on a single chip a CPU, various types of memory (ROM, RAM, EEPROM or Flash), and a set of input/output peripherals that allow direct interaction with the physical environment.

In addition to its core processing unit, a typical microcontroller includes functional modules such as analog-to-digital converters (ADC), digital-to-analog converters (DAC), timers, and serial communication interfaces (UART, SPI, I²C, CAN). Some modern versions even embed wireless communication modules like Bluetooth or Wi-Fi, enabling remote control and IoT integration.

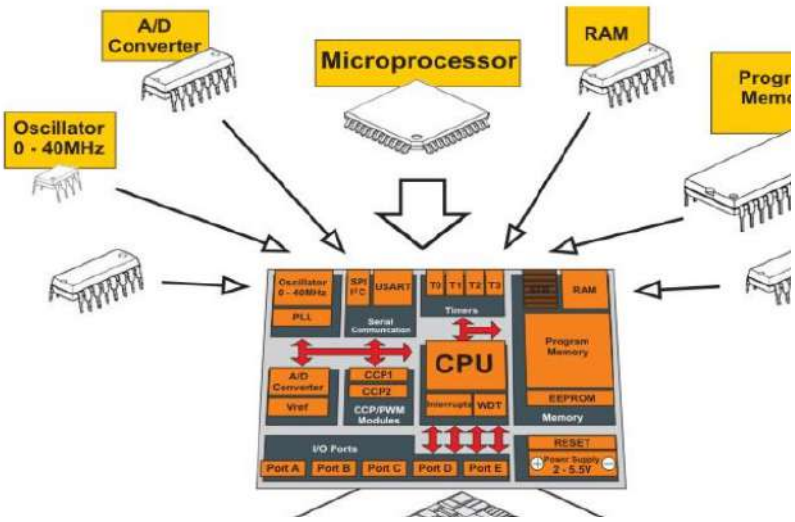

Figure I. 12 - Simplified schematic of the intelligent urban lighting system and the developed prototype.

The main role of a microcontroller is to execute specific, real-time control tasks with precision and reliability. It continuously acquires data from sensors, processes it according to programmed control algorithms, and actuates the corresponding outputs, such as regulating temperature, controlling motor speed, and triggering safety mechanisms.

Unlike microprocessors, microcontrollers generally operate without a full operating system, relying instead on directly programmed loops or lightweight real-time operating systems (RTOS). This allows deterministic behavior, a key requirement for time-sensitive applications.

Their importance is fundamental across a wide range of domains, including industrial automation, automotive embedded systems (ABS, airbag systems, electronic fuel injection), medical devices, IoT (Internet of Things), and robotics. By combining computational capability, energy efficiency, and compact design, microcontrollers enable the creation of intelligent, autonomous, and highly optimized electronic systems.

I.3.3 Differences Between a Microcontroller and a Microprocessor

Although microcontrollers and microprocessors share a similar core principle — executing instructions to process data — they differ significantly in architecture, functionality, and applications. These distinctions determine their suitability for specific types of embedded or general-purpose systems.

A) Architecture:

A microprocessor is primarily designed for high-speed computation. It lacks integrated memory and peripherals, relying on external components such as RAM, ROM, and I/O controllers to operate. This modular architecture provides high flexibility and scalability but results in greater hardware complexity and power consumption. Conversely, a microcontroller follows a System-on-Chip (SoC) architecture, integrating the processor, memory, and input/output peripherals within a single chip. This integration minimizes energy consumption, reduces size and cost, and simplifies system design. For example, an Intel Core i7 operates with multiple cores clocked at several gigahertz and large cache memories, making it ideal for multitasking and computationally intensive tasks. In contrast, a PIC18F452 microcontroller runs at about 40 MHz with a few tens of kilobytes of memory but efficiently performs precise control operations in embedded applications.

B) Functionalities:

A microprocessor is designed to run complex operating systems such as Windows or Linux, handling multiple applications simultaneously. It manages virtual memory, cache

hierarchies, and vector instruction sets, enabling efficient execution of scientific computations, AI algorithms, and multimedia processing.

A microcontroller, on the other hand, focuses on real-time control and hardware interaction. It includes dedicated modules such as watchdog timers to prevent software crashes, PWM (Pulse Width Modulation) units for motor control, and built-in communication interfaces (SPI, I²C, UART, CAN) for direct connection with sensors and actuators.

C) Typical Applications:

Microprocessors are typically used in computers, servers, and high-performance systems requiring intensive computation — such as AI processing, scientific simulations, and video rendering.

Microcontrollers are primarily used in embedded systems, where reliability, real-time operation, and low power consumption are essential. They power devices such as drone control boards, mobile robots, smart home systems, IoT devices, and energy controllers in smart grids.

In summary, the microprocessor dominates environments where performance and multitasking are priorities, while the microcontroller excels in embedded applications requiring compactness, autonomy, and deterministic real-time control.

I.3.4 Defining the Specifications for Selecting a Processor or Microcontroller

Selecting the appropriate processor or microcontroller is a critical step in the design of an embedded system, as it determines the overall performance, cost, and scalability of the final product. This choice must be guided by a well-defined specifications document (cahier des charges) that outlines the system's objectives, constraints, and operational requirements. The hardware and software architecture of the embedded platform will ultimately depend on this initial decision.

The synthesis diagram highlights two major scenarios. When a project must manage multiple applications simultaneously or when the requirements are not fully defined, a microprocessor is the preferred choice due to its expandable memory, high processing power, and scalability. Conversely, for a dedicated, application-specific project, a microcontroller is better suited, as it integrates the processor, memory, and I/O peripherals within a single compact and energy-efficient chip.

In order to establish the specifications for choosing the right component, several selection criteria must be evaluated:

- Processing Capacity: The computational performance required is a decisive factor. A microprocessor is ideal when the system must handle complex algorithms, multitasking, or real-time data processing at high speed. Conversely, for applications with well-defined and repetitive tasks such as control, regulation, or monitoring, a microcontroller is sufficient and more cost-effective.
- Integrated Peripherals: The level of integration is another major consideration. Microcontrollers typically include embedded peripherals such as ADC/DAC converters, timers, serial communication interfaces (UART, SPI, I²C), and PWM modules, minimizing the need for external components. Microprocessors, however, require additional hardware but support advanced operating systems such as Linux, Windows Embedded, or Android, offering greater flexibility in software development.
- Reliability and Safety: In safety-critical applications such as aerospace, medical devices, or automotive electronics, certified microcontrollers designed for functional safety standards (e.g., ISO 26262, IEC 61508) are often preferred. In contrast, microprocessors are more common in high-performance environments requiring extensive software capabilities, such as industrial automation or embedded AI systems.

A well-structured requirements document should therefore balance these parameters — performance, integration, reliability, power consumption, and cost — to guide the optimal selection between a microcontroller and a microprocessor.

I.3.5 Specific Project Requirements

The choice between a microcontroller and a microprocessor depends largely on the specific needs of the application. In industrial control systems, where real-time monitoring of sensors and actuators is essential, the microcontroller is naturally favored due to its low latency, simplicity, and direct hardware control capabilities. Conversely, in multimedia or high-performance applications—such as computer vision systems, smart devices, or connected objects requiring complex data handling—the microprocessor becomes indispensable. Its ability to manage rich graphical interfaces, large databases, and complex communication protocols makes it the preferred option for such demanding environments.

Flexibility is another determining factor. When a project must evolve rapidly, allow for frequent software updates, or operate across diverse environments, the microprocessor provides the necessary scalability and adaptability. On the other hand, if the objective is to develop a robust, stable, and application-specific solution, the microcontroller remains the most suitable choice thanks to its reliability and predictable behavior.

I.3.6 Power, Cost, and Size Constraints

Hardware constraints often play a decisive role in system design. In battery-powered or portable applications, power consumption becomes a critical parameter. Microcontrollers, with their low-power modes and optimized energy efficiency, are typically preferred to extend battery life. Microprocessors, being more power-hungry, require additional thermal management and stable power supplies, which increases system complexity.

Production cost is another key consideration, especially for mass-produced systems. Microcontrollers, by integrating most peripherals on a single chip, reduce the need for external components, resulting in a lower overall cost. Microprocessors, in contrast, entail higher expenses—not only for the component itself but also for the external memory, I/O controllers, and supporting hardware required for their operation.

Finally, size and compactness strongly influence the choice. Microcontrollers, which integrate memory, processing, and interfaces in a single compact package, are ideal for miniaturized embedded applications such as wearable or portable devices. Microprocessors, being more complex and often mounted on multi-chip boards, are better suited to systems where space is less constrained and performance is prioritized.

In summary, the final decision must balance application requirements, energy constraints, production costs, and system size to achieve the optimal design for an embedded solution.

I.4 Presentation of Some Microcontroller Boards

Development boards are essential platforms for embedded systems designers. They facilitate programming, rapid prototyping, and functional validation of applications before their final integration. They allow direct hardware manipulation, testing of algorithms, and easy interfacing with various sensors, actuators, and communication modules.

Among the most popular boards used in education, research, and industry, the following are widely adopted: Arduino, Raspberry Pi, and STM32 Discovery Board.

Each offers distinct technical features that meet specific needs in terms of computing power, flexibility, and cost — ranging from simple educational projects to advanced high-performance embedded systems.

I.4.1 Arduino

The Arduino board is an open-source prototyping platform widely known for its simplicity and flexibility. It is based on ATmega microcontrollers (8-bit AVR architecture) and provides a user-friendly development environment (Arduino IDE), suitable for both beginners and experienced engineers. Main features (Arduino Uno):

- Microcontroller: ATmega328P (AVR RISC, 8-bit)
- Clock frequency: 16 MHz
- Memory: 32 KB Flash, 2 KB SRAM, 1 KB EEPROM
- I/O pins: 14 digital (6 PWM) and 6 analog inputs
- Communication interfaces: UART, SPI, I²C
- Operating voltage: 5 V

Thanks to its wide community and rich library ecosystem, Arduino is ideal for basic robotics, IoT devices, environmental sensors, and educational prototypes.

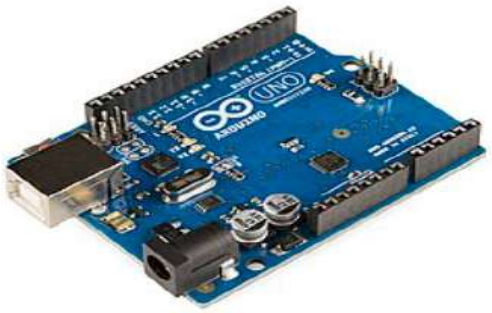

Figure I. 13 - Arduino Uno development board based on the ATmega328P microcontroller

I.4.2 Raspberry Pi

Unlike Arduino, the Raspberry Pi is not a simple microcontroller board but a single-board nano-computer capable of running a full operating system (Linux, Raspbian, Ubuntu). It is powered by an ARM processor and provides significantly higher performance, making it suitable for applications requiring intensive computation, advanced connectivity, or graphical interfaces. Main features (Raspberry Pi 4 Model B):

- Processor: ARM Cortex-A72 (quad-core, 1.5 GHz)

- RAM: 2 GB, 4 GB, or 8 GB LPDDR4
- Storage: microSD card
- Interfaces: 40 GPIO, HDMI, USB 2.0/3.0, Ethernet, Wi-Fi, Bluetooth
- Operating system: Linux-based (supports Python, C/C++, Java, etc.)

This board is particularly suitable for computer vision, IoT gateways, embedded servers, and intelligent robotics applications.

Figure I. 14 - Raspberry Pi 4 Model B — a single-board nano-computer

I.4.3 STM32 Discovery Board

The STM32 Discovery Board, developed by STMicroelectronics, is a professional prototyping platform designed for real-time embedded applications.
It is based on ARM Cortex-M microcontrollers, known for their low power consumption and high performance. Main features (STM32F407 Discovery):

- Microcontroller: STM32F407VGT6 (ARM Cortex-M4, 32-bit)
- Clock frequency: up to 168 MHz
- Memory: 1 MB Flash, 192 KB SRAM
- Interfaces: USART, SPI, I²C, CAN, USB OTG, Ethernet (depending on model)
- Integrated peripherals: accelerometer, 12-bit ADCs, advanced timers
- Operating voltage: 3.3 V

Thanks to its powerful processor and integrated modules, this board is ideal for industrial control systems, automation, real-time robotics, and complex data acquisition applications.

Figure I. 15 - STM32F4 Discovery Board featuring an ARM Cortex-M4 microcontroller

I.5 How to Select and Integrate a Development Board into an Embedded Project

The selection and integration of a development board is a strategic step in the design of an embedded system. This decision depends on functional requirements, technical and economic constraints, and the educational or industrial objectives of the project. A well-chosen board ensures not only performance and reliability but also ease of implementation and scalability of the prototype.

I.5.1 Project Requirements Assessment

The first step is to analyze the specific needs of the project in order to identify the most suitable board. This assessment is based on the following criteria:

- Nature of processing tasks: distinguish between simple tasks (sensor reading, actuator control) and complex ones (image processing, machine learning, IoT communication, etc.).
- Peripherals to interface: number and type of sensors, communication modules (UART, SPI, I²C, CAN, USB, Ethernet), or output devices.
- Required processing speed and power: depending on the algorithms to be executed or the system's real-time responsiveness.
- Available memory: amount of RAM for data, Flash for program storage, and EEPROM for saved parameters.

This analysis helps determine whether a low-cost, easy-to-use board (such as Arduino Uno) is sufficient, or whether a more powerful and connected platform (such as Raspberry Pi or STM32 Discovery) is required.

I.5.2 Hardware and Software Compatibility

Once the project needs are defined, it is essential to verify the board's compatibility with the project's hardware and software environment.

Hardware compatibility:

- Availability and number of GPIO pins.
- Support for required communication protocols (SPI, I²C, UART, CAN, etc.).
- Ability to connect additional modules or shields.

Software compatibility:

- Availability of development environments (IDEs) and supported programming languages.
- Access to libraries, frameworks, and example projects.
- Compatibility with embedded operating systems (Linux, FreeRTOS, etc.).
- Scalability: possibility to add new sensors or functionalities without redesigning the entire system.

I.5.3 Economic Considerations

Cost is a major factor, particularly for educational projects or systems intended for mass production. Key elements to consider include:

- Board price: Arduino boards are low-cost, while Raspberry Pi and STM32 boards are more expensive but offer higher performance.
- Required accessories: sensors, power supplies, extension boards, communication modules, etc.
- Development and maintenance cost: ease of learning, documentation availability, and software tool accessibility.
- Performance-to-cost ratio: avoid overdimensioning the hardware for simple applications.

I.5.4 Programming and Development

Development efficiency largely depends on the availability of software tools and technical support resources.

Integrated Development Environments (IDE):

- Arduino IDE: simple and intuitive, ideal for learning.
- STM32CubeIDE: professional and comprehensive, suited for real-time embedded projects.

- Linux/Python (Raspberry Pi): powerful and flexible for connected or AI-based applications.

Supported programming languages:

- Arduino → C/C++.
- Raspberry Pi → Python, C/C++, Java.
- STM32 → C/C++, Assembly, sometimes MicroPython.

Community and technical support: Open-source platforms like Arduino and Raspberry Pi benefit from vast user communities, detailed documentation, and a wealth of educational resources.

Simulation and testing: The ability to simulate algorithms before deployment (e.g., using Proteus, Tinkercad, or STM32CubeMX) facilitates project validation and debugging, improving development efficiency.