IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Comprehensive Evaluation And Selection Methods For Microcontrollers In Embedded And Iot Applications

Ashutosh Mishra

System Analyst
Department of Computer Application
C.M.P. Degree College, Prayagraj, India

Abstract: This paper provides an in-depth analysis of microcontrollers (MCUs), covering their internal architecture, categories, primary components, and diverse applications. It explores widely used microcontroller families such as ARM, AVR, and PIC, offering comparisons based on features and typical use scenarios. To assist in selecting a suitable MCU for embedded and IoT-based systems, the paper introduces structured decision-making frameworks including the Kano model, specification matrices, and pin compatibility evaluations. It further outlines essential selection criteria such as memory configuration, energy efficiency, development tool support, and hardware compatibility. Common challenges like timing errors, thermal issues, and interface mismatches are also discussed, with suggestions for mitigation. The insights presented aim to guide engineers, students, and developers in making strategic microcontroller choices tailored to system demands and performance requirements.

Index Terms: Microcontroller, MCU, Embedded Systems, Internet of Things, IoT, AVR / ARM / PIC, Kano Model

I. INTRODUCTION

Microcontrollers are fundamental components in the development of modern embedded systems and Internet of Things (IoT) applications. These compact devices integrate a processor, memory, and input/output interfaces into a single chip, enabling them to manage control tasks efficiently in real-time environments. They are found in a wide array of devices, from smart home systems and wearable technology to automotive and industrial control units. Selecting the right microcontroller is a pivotal part of system design, as it impacts overall performance, cost, power consumption, and system integration. With a multitude of microcontroller families available—each offering different specifications and functionalities—the selection process requires a detailed evaluation of factors such as processing capability, memory size, peripheral availability, and compatibility with existing components. This paper delves into the various architectures, classifications, and operational principles of microcontrollers, while also examining practical strategies and tools for selecting the most appropriate microcontroller for specific applications. By addressing common design challenges and proposing structured selection methods, the work serves as a valuable guide for both newcomers and experienced developers in the field of embedded systems.

II. MICROCONTROLLER AND IT'S COMPONENTS

A microcontroller (MCU) is a compact, integrated computing device engineered to perform dedicated control tasks within embedded systems. It combines a central processing unit (CPU), memory, and input/output (I/O) peripherals onto a single chip, offering a streamlined and cost-effective solution for real-time control applications.

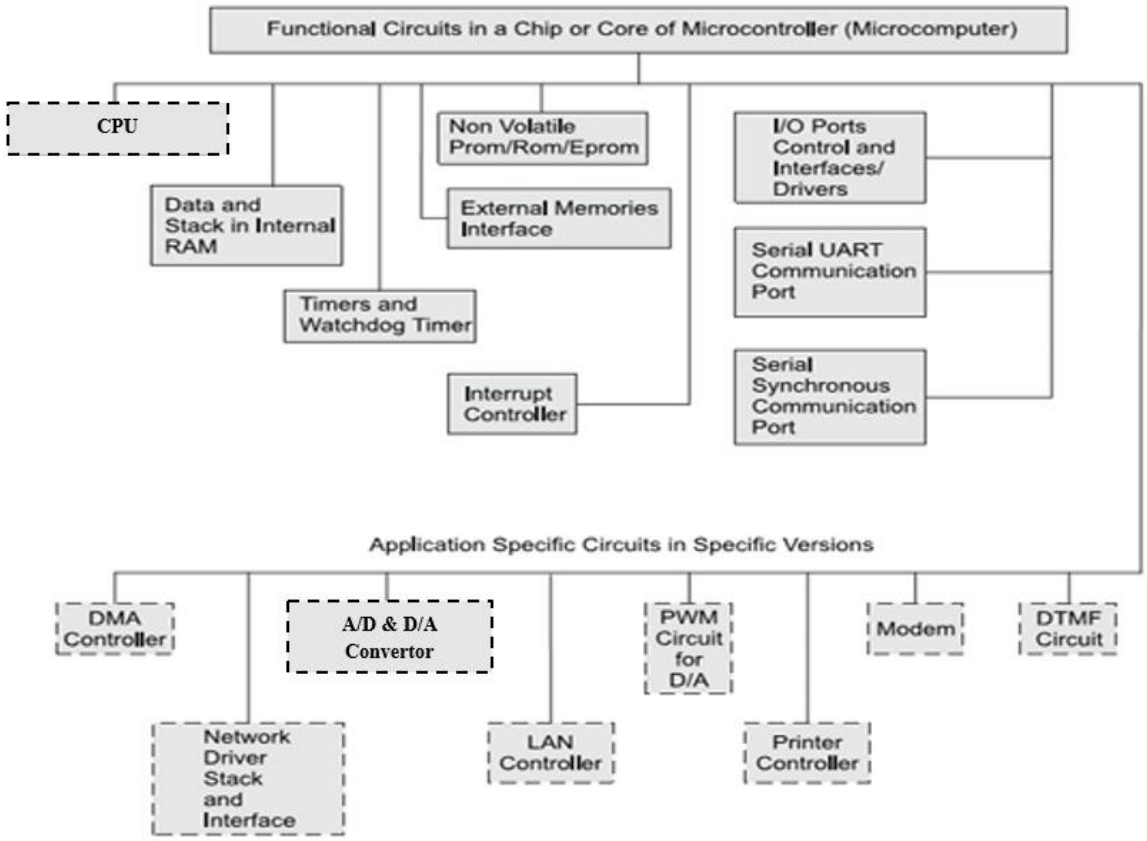

Fig 1: Devices and functional circuits (boxes with solid boundary) and the application-specific units (boxes with dotted boundary) in a specific version of a microcontroller

Key Components of a Microcontroller

Central Processing Unit (CPU): Acts as the core of the MCU, executing program instructions and handling data processing. It may use either a Reduced Instruction Set Computing (RISC) or Complex Instruction Set Computing (CISC) architecture, depending on the design. **Memory Units:** *Volatile Memory (RAM):* Temporarily stores data during execution. *Non-Volatile Memory:* Used for storing firmware or program code, typically implemented as ROM, Flash, or EEPROM. **I/O Peripherals:** Facilitate communication with external components such as sensors, actuators, displays, and switches, enabling the MCU to interact with the physical environment. **Timers and Counters:** Crucial for managing time-sensitive tasks like delays, pulse-width modulation (PWM), and event counting. **Communication Interfaces:** Enable data exchange with other devices through protocols like UART, SPI, I^2C, and CAN. **Analog/Digital Converters:** *ADC (Analog-to-Digital Converter):* Allows the microcontroller to read analog signals. *DAC (Digital-to-Analog Converter):* Converts digital outputs to analog signals for controlling devices such as speakers or analog actuators.

The integration of these components onto a single chip allows microcontrollers to function autonomously within embedded systems. Their small size, energy efficiency, and low cost make them suitable for a wide range of applications (Fig.2).

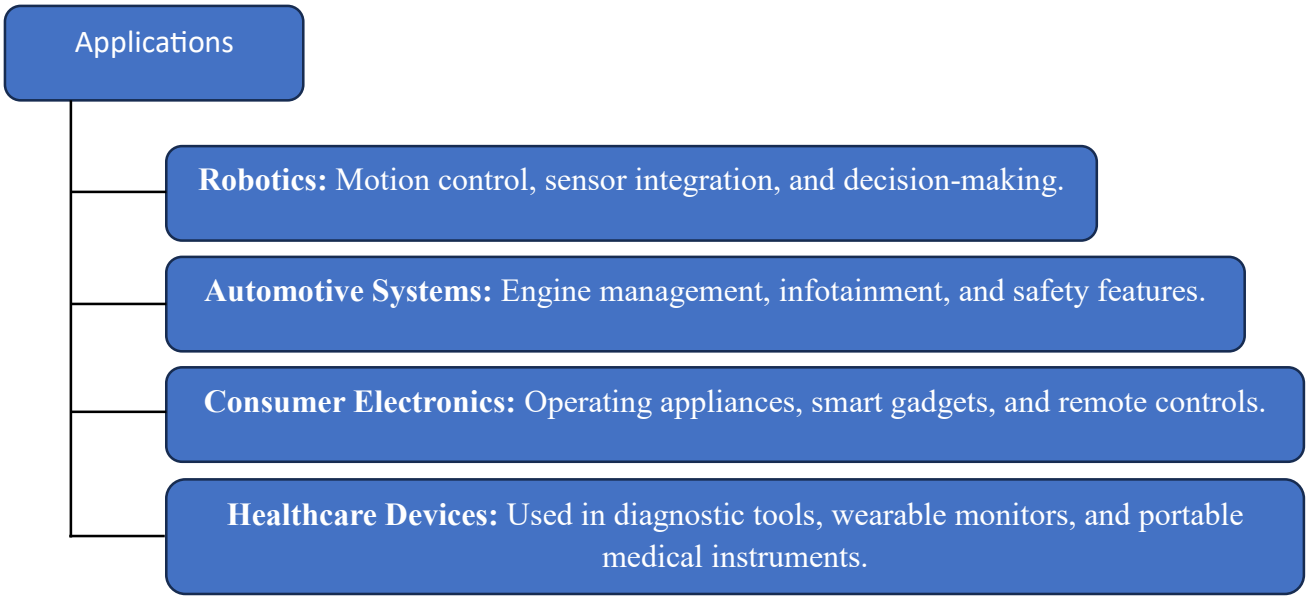

Fig.2: Various Applications of Microcontroller

By incorporating multiple control and processing functions into a compact form factor, microcontrollers have become essential to the design and functionality of intelligent electronic systems across numerous industries.

III. TYPES OF MICROCONTROLLER BOARDS

Microcontroller development boards serve as platforms for prototyping and deploying embedded applications. Each board integrates a specific MCU and offers distinct features that suit different use cases, from simple DIY projects to complex real-time systems. Below are some commonly used microcontroller boards and their key characteristics: **Arduino Uno:** Based on the ATmega328P microcontroller, this open-source board is known for its simplicity and extensive community support. It is particularly suitable for beginners, hobbyists, and educational projects due to its user-friendly programming environment and accessible hardware. **Raspberry Pi Pico:** Featuring the RP2040 chip, this board is a cost-effective option for embedded applications. It offers dual-core processing capabilities and is ideal for physical computing tasks that require performance on a budget. **ESP32:** Built around the Tensilica Xtensa LX6 processor, the ESP32 offers integrated Wi-Fi and Bluetooth, making it highly suitable for IoT applications. Its wireless communication capabilities allow it to serve in smart devices, remote monitoring systems, and home automation setups. **STM32F4 Discovery Board:** Powered by the ARM Cortex-M4 processor, this development board delivers high-speed performance and is equipped for real-time operations. It is preferred for advanced applications like digital signal processing, control systems, and robotics. **PIC16F877A:** A versatile 8-bit microcontroller board from Microchip, the PIC16F877A includes a variety of built-in peripherals. It is commonly used in industrial automation, instrumentation, and general embedded control systems. These boards vary in terms of processing power, peripheral support, memory, and connectivity options, making it important to align board selection with project-specific requirements.

IV. CLASSIFICATION OF MICROCONTROLLERS

Microcontrollers are broadly categorized based on their data bus width and internal architecture, which influences their processing capability, complexity, and suitable application domains. Understanding these classifications is essential for selecting an appropriate MCU for a given task.

Classification Based on Data Bus Width

8-bit Microcontrollers: These MCUs handle data in 8-bit chunks and are ideal for basic applications that require limited processing power and minimal memory. Due to their simplicity and low power usage, they are frequently found in home appliances, toys, and simple automation systems. **16-bit Microcontrollers:** With enhanced data handling capabilities compared to 8-bit variants, 16-bit MCUs offer a balance between performance and resource usage. They are used in automotive subsystems, smart instruments, and control applications requiring greater precision. **32-bit Microcontrollers:** Equipped with advanced features and high-speed processing, 32-bit MCUs can support complex operations, multitasking, and large memory access. These are preferred in demanding applications like multimedia devices, smart home controllers, and advanced automotive electronics.

Classification Based on Architecture and Family

AVR Microcontrollers: Developed by Atmel, AVR MCUs use RISC architecture and are recognized for their efficiency and low power consumption. They are commonly used in educational platforms, hobby electronics, and consumer products. **PIC Microcontrollers:** Manufactured by Microchip Technology, the PIC family spans 8-bit to 32-bit devices and is known for its wide peripheral support and ease of programming. They are popular in industrial control systems, embedded automation, and IoT solutions. **ARM-based Microcontrollers:** Utilizing the ARM processor architecture, these MCUs range from simple Cortex-M cores for low-power tasks to powerful Cortex-A/R series for high-performance computing. Their scalability and software support make them a leading choice for embedded and portable systems. **FPGA-based Microcontrollers:** These hybrid devices merge microcontroller features with the configurability of FPGAs (Field-Programmable Gate Arrays), allowing for custom hardware implementations. They are best suited for specialized applications like signal processing, prototyping, and high-speed data operations. Selecting the right type of microcontroller architecture depends heavily on the computational needs, power constraints, peripheral requirements, and budget of the intended application.

V. WORKING OF MICROCONTROLLERS

A microcontroller operates as a compact control unit that executes programmed instructions to perform specific tasks within an embedded system. It comprises several essential components that work together in a coordinated manner: **Central Processing Unit (CPU):** The CPU is responsible for interpreting and executing instructions stored in the program memory. It manages data operations, controls peripheral interactions, and oversees the overall logic flow. **Memory Units:** Microcontrollers include different types of memory: *Program Memory* stores the firmware or application code. *Data Memory* (such as SRAM) is used for temporary storage during program execution. *Non-Volatile Memory* (like EEPROM or Flash) retains data even when power is turned off. **Input/Output (I/O) Interfaces:** These interfaces allow the MCU to connect with external devices, such as sensors, switches, and actuators. Through these I/O ports, the microcontroller gathers input data and controls output signals. **System Clock:** A built-in oscillator or external crystal provides the timing signal that synchronizes all internal operations. The clock frequency influences the speed at which the microcontroller processes instructions and interacts with peripherals.

Operational Workflow

When powered on, the microcontroller begins executing the program stored in its memory. It may read inputs from sensors, process the data using logical or arithmetic instructions, and generate outputs accordingly—for example, turning on a motor or displaying values on an LCD. Real-time responsiveness is a critical feature of microcontrollers, making them suitable for automation and control systems. Their ability to respond quickly to external changes allows them to maintain accurate control in time-sensitive applications such as robotics, industrial automation, and embedded medical devices.

VI. APPLICATIONS OF MICROCONTROLLER

Microcontrollers are essential components in a wide range of electronic systems, thanks to their compact design, low power consumption, and real-time control capabilities. Their versatility allows them to be embedded into countless applications across various industries. Key application areas include: **Home Automation:** Used in systems like smart lighting, automated temperature control, and intelligent security setups. MCUs enable efficient control of home devices through sensors and wireless communication. **Automotive Systems:** Integrated into vehicle electronics for functions such as engine management, transmission control, airbag systems, and electric power steering. Modern vehicles rely on multiple MCUs for safe and efficient operation. **Medical Electronics:** Deployed in portable and wearable medical devices, including glucose meters, heart rate monitors, and diagnostic instruments. MCUs enable data processing, monitoring, and wireless communication with medical databases. **Consumer Electronics:** Found in everyday gadgets such as television remotes, gaming consoles, and smartwatches. They manage inputs, control outputs, and enable user interaction. **Industrial Automation:** Used in robotics, process controllers, and instrumentation systems. Microcontrollers are integral in ensuring precision, reliability, and timing accuracy in factory operations. **Internet of Things (IoT):** Serve as the processing units in IoT devices for smart homes, agriculture, cities, and wearable technologies. They manage sensing, connectivity, and cloud integration functions. **Aerospace and Defence:** Embedded in avionics systems, communication modules, and navigation units. MCUs in this domain must meet high standards of reliability and environmental resilience.

Microcontrollers continue to play a pivotal role in enhancing system intelligence and automation, bridging the gap between physical hardware and digital control across a wide spectrum of modern technologies.

VII. COMMON ISSUES WITH MCUS, AND THE PROCEDURE TO RESOLVE THEM

Despite their reliability, microcontrollers may encounter various issues during operation. Understanding these common problems and knowing how to address them is essential for developing stable and efficient embedded systems.

1. Timing Irregularities Problem: Inaccurate timing signals or poor interrupt handling can cause delays or unintended behavior in time-sensitive operations. **Solution:** Use high-precision external oscillators for tasks requiring accurate timing. Implement software-based synchronization and thorough timing analysis to ensure consistent performance. **2. Power Instability Problem:** Voltage fluctuations or poor power supply regulation may result in unpredictable behavior, data corruption, or permanent damage. **Solution:** Integrate reliable voltage regulators and power management ICs. Design systems to tolerate transient power changes and include power-saving modes where applicable. **3. Thermal Overload Problem:** Continuous high processing load or poor thermal design can cause overheating, leading to system instability or hardware degradation. **Solution:** Include heat dissipation mechanisms such as heat sinks or ventilation in your design. Optimize software routines to reduce processor workload and power consumption. **4. Electromagnetic Interference (EMI) and Signal Noise Problem:** EMI or radio frequency interference can distort signals, resulting in incorrect data interpretation or failed communication. **Solution:** Apply shielding techniques, proper grounding, and filtering (e.g., ferrite beads, capacitors) to minimize noise. Use PCB design practices that reduce EMI susceptibility. **5. Software Bugs and Inefficient Code Problem:** Faulty or poorly optimized code can lead to erratic behavior, security vulnerabilities, or resource bottlenecks. **Solution:** Use static code analysis tools, follow standardized coding practices, and conduct thorough testing and debugging throughout development. **6. Security Vulnerabilities Problem:** Unprotected microcontrollers can be targeted for unauthorized access, data theft, or system control. **Solution:** Implement security features such as encrypted communication, secure bootloaders, and hardware-level access controls from the early design stage. **7. Incompatibility with Other Components Problem:** Issues can arise when the microcontroller cannot interface properly with other system components due to mismatched voltage levels, protocols, or physical connectors. **Solution:** Ensure proper selection of components with compatible electrical and communication standards. Use level shifters or protocol converters if needed. Proactively identifying and addressing these challenges during the design and testing phases ensures greater system reliability and reduces the risk of deployment failures.

VIII. FREQUENTLY USED MICROCONTROLLERS AND THEIR FEATURES

A wide variety of microcontrollers are available in the market, each designed to cater to specific application needs. Table 1 shows an overview of some widely adopted microcontroller families, highlighting their core features and common use cases:

Table 1: Microcontrollers and their features

Frequently used MCUs	Features
Intel 8051	An iconic 8-bit microcontroller series, the 8051 is known for its simplicity and legacy use in early embedded systems. It typically includes up to 8KB of ROM and either 128 or 256 bytes of RAM. With four 8-bit bidirectional I/O ports and clock speeds reaching 12 MHz, it serves well in educational and control applications. The memory can be erased using UV light or programmed electrically, depending on the variant.
Microchip PIC16C5X/XX	This 8-bit MCU family offers low power consumption and operates at frequencies up to 40 MHz. Memory capacity ranges from 512 bytes to 2KB of ROM and 25 to 73 bytes of RAM. Notable features include an 8-bit real-time timer, a watchdog timer for fail-safes, and a power-saving sleep mode, making it suitable for battery-operated devices.
Atmel AT89CXXXX	These microcontrollers support Flash memory ranging from 1KB to 8KB and contain 256 bytes of internal RAM. Operating at clock speeds up to 20 MHz, they offer a variety of I/O pins, timers, and interrupt options. Their compactness and performance make them a popular choice for embedded project development.
Philips Corporation	Philips has developed enhanced versions of the traditional 8051 architecture, incorporating features like analog-to-digital and digital-to-analog converters, expanded I/O capabilities, and flexible memory options, including OTP and Flash.
Freescale 68HC11	An 8-bit controller designed with embedded control in mind, this microcontroller can operate up to 3 MHz. It provides ROM that can be UV or electrically erased, and RAM capacities of up to 768 bytes. Integrated analog-to-digital converters support signal acquisition tasks in instrumentation and automation.
Dallas Semiconductor DS89C4XXX, DS5000, DS80C320, DS87520	These controllers offer both Flash and UVEPROM options, with up to 64KB of program memory and 128 to 256 bytes of RAM. They feature 32 I/O pins, multiple timers, and several interrupt channels, making them suitable for legacy systems and general-purpose control.
Zilog Older: Z8, Z180 Newer: eZ8, eZ80, Z16	Zilog's early 8-bit Z8 microcontrollers are based on Harvard architecture. The newer generations, such as the eZ8 and Z16, offer pipelined execution, on-chip Flash and SRAM, and enhanced instruction sets. The Z16 series supports a combination of 8/16/32-bit processing with high addressable memory, making it fit for performance-oriented embedded systems.
Texas Instruments ARM Stellaris LM4F	Combining features of 16- and 32-bit architectures, this MCU series is designed for high-performance applications. It offers up to 256KB of Flash memory, USB 2.0 support, and speeds up to 80 MHz. Pre-installed software libraries in factory ROM and extensive peripheral integration make it ideal for communication and data processing applications.

These microcontrollers differ in architecture, performance, memory size, and peripheral support, allowing developers to choose based on the specific technical and functional needs of their projects.

IX. METHODS FOR SELECTING A MICROCONTROLLER

Choosing an appropriate microcontroller requires a structured approach due to the wide variety of available options, each differing in performance, cost, memory, and peripheral capabilities. Several methods can be used to streamline the selection process based on application requirements and system constraints.

Systematic Approach

The selection typically begins with identifying the core requirements of the intended application. This includes analyzing necessary features like processing speed, memory size, I/O availability, and communication interfaces. Because microcontrollers vary widely in these parameters, balancing performance with cost can be challenging. One helpful strategy involves categorizing requirements using the **Kano Model**: **Basic Requirements:** Fundamental functions the MCU must support (e.g., required number of I/O pins, minimum memory). **Performance Requirements:** Features that directly impact system performance, such as clock speed or ADC resolution. **Excitement Requirements:** Additional features that enhance the design but are not strictly necessary, such as built-in wireless communication. By distinguishing between these categories, designers can better prioritize and select microcontrollers that meet essential needs while considering desirable extras.

Requirements vs. Specification Matrix: This method involves listing key system requirements and comparing them against the features of various microcontrollers in a tabular format. For example:

Requirement	MCU1	MCU2	MCU3	MCU4
Req1	✓	✓	✕	✓
Req2	✓	✕	✓	✓
Req3	✓	✓	✓	✕

Fig 3: Sample requirement versus specification matrix

This visual representation helps quickly eliminate unsuitable options and highlight candidates that meet most or all critical criteria. However, while helpful for comparison, the matrix doesn't quantify how well each feature matches the need, leaving the final decision to subjective judgment.

Pin Assignment Method

This technique focuses on hardware compatibility, specifically the allocation of microcontroller pins to meet the system's interfacing requirements. Some approaches use constraint-satisfaction algorithms to automatically match system functions (like UART, PWM, or SPI) to the microcontroller's pin layout. Although effective in ensuring I/O compatibility, this method mainly addresses interface matching and may not account for other critical aspects such as processing power or memory capacity. Therefore, it is best used in later design stages when physical layout becomes a priority. Overall, combining these selection methods—functional classification, specification matching, and pin compatibility analysis—can significantly simplify the decision-making process and lead to more informed microcontroller choices.

X. CRITERIA FOR CHOOSING AN APPROPRIATE MICROCONTROLLER

Selecting the right microcontroller involves careful assessment of technical and practical parameters to ensure it meets the current system requirements and can adapt to future needs. The following criteria help guide developers in making informed decisions during the design phase.

Application Requirements

The starting point in microcontroller selection is to clearly define the intended functionality of the system. Depending on the complexity, one must decide between simple controllers (e.g., 8-bit) or more advanced ones (e.g., 32-bit). For basic control logic and cost-sensitive designs, 8-bit MCUs may suffice. On the other hand, applications demanding higher data throughput, signal processing, or multitasking often require 16- or 32-bit MCUs. Additionally, built-in peripherals like timers, ADCs, and communication ports should align closely with the system's needs to avoid external component dependencies.

Memory Considerations

Memory plays a crucial role in microcontroller performance. Three main memory types are considered and shown in Fig 4.:

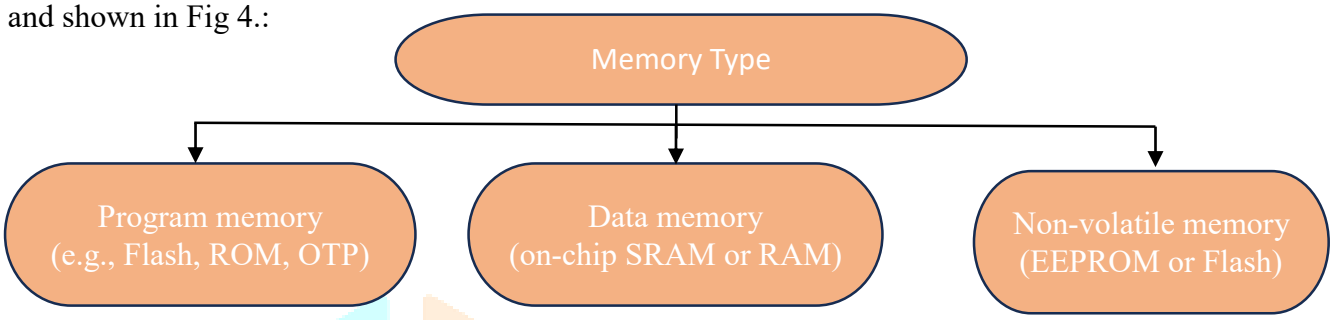

Fig 4: Various Memory Types

Choosing an MCU with sufficient onboard memory allows scalability and helps reduce reliance on external memory components. For example, Atmel's AVR MCUs provide integrated Flash, SRAM, and EEPROM, allowing flexible memory management.

Component Availability

Availability of the selected MCU in the required quantity is vital for production and scaling. Opting for widely distributed microcontrollers minimizes supply chain disruptions and ensures long-term product viability.

Physical Size and Packaging

Space constraints in compact systems necessitate selecting MCUs with an appropriate pin count and package size. Avoiding oversized packages with unused I/O pins helps reduce PCB size and cost while simplifying layout design.

Compatibility and Upgrade Path

Microcontrollers with pin-compatible or functionally similar alternatives offer greater flexibility in design revisions and upgrades. Choosing a product family that supports future scaling without major redesigns can save development time and effort.

Pre-Deployment Testing

Before committing to full-scale development, it's crucial to validate the MCU in a prototype setup. This ensures the microcontroller performs as expected under operational conditions and allows early detection of design flaws.

Power Efficiency

Power consumption directly affects battery life and thermal performance, especially in mobile or remote applications. Developers should evaluate the MCU's power profiles at different clock speeds and leverage features like sleep, idle, or deep power-down modes. Some controllers, like those in the AVR family, operate effectively at voltages as low as 1.8V, optimizing energy use.

Manufacturer Reputation

Established manufacturers provide long-term support, reliable documentation, and consistent product quality. Companies like Microchip, Texas Instruments, and STMicroelectronics have strong reputations for delivering stable and well-supported MCUs.

Technical Support and Documentation

Efficient development depends on accessible documentation and support services. Vendors that offer detailed datasheets, application notes, online forums, and responsive customer service significantly enhance the design experience.

Development Tools and Software Ecosystem

The availability of compilers, debuggers, simulators, and integrated development environments (IDEs) is a major factor in MCU selection. Tools like MPLAB for PIC microcontrollers or Keil uVision for ARM-based systems streamline development and testing. High-level language support (like C/C++) also improves portability and reduces development time.

Cost Considerations

Cost remains a deciding factor, especially for large-scale production. Designers must weigh the cost of the MCU and any external components it requires. Choosing an MCU with integrated features such as USB, CAN, or Ethernet may help lower the total bill of materials by eliminating additional chips. By evaluating these factors holistically, developers can select microcontrollers that not only satisfy present design goals but also provide room for enhancement and scaling in future iterations.

XI. CONCLUSION

Microcontrollers are at the core of modern embedded and IoT systems, providing efficient, real-time control across diverse applications such as healthcare, automotive, consumer electronics, and industrial automation. This paper has explored the internal structure, classifications, and operational principles of microcontrollers, while also presenting a comparative overview of commonly used development boards. Given the wide range of microcontrollers available, selecting the most suitable one is a complex yet essential task. Structured methods—such as the Kano model, feature comparison matrices, and pin assignment techniques—offer a systematic way to evaluate and narrow down choices. In addition to technical fit, factors such as availability, development tools, scalability, power efficiency, and cost must be carefully considered. Furthermore, the paper has highlighted common operational challenges—including timing errors, power instability, and noise interference—and outlined strategies to mitigate these issues, contributing to more robust system design.

In summary, effective microcontroller selection involves balancing performance requirements with practical constraints. By applying thoughtful evaluation techniques and leveraging suitable tools, developers can ensure their designs achieve optimal functionality, reliability, and long-term viability.

REFERENCES

1. Parai, M. K., Das, B., & Das, G. (2013). An overview of microcontroller unit: From proper selection to specific application. International Journal of Soft Computing and Engineering (IJSCE), 2(6), 2231–2307.

2. K., S., & Parameswari, P. (2022). Microcontroller selection for automotive embedded systems [Master's thesis, DIVA Portal]. https://www.diva-portal.org/smash/get/diva2:1712830/FULLTEXT01.pdf

3. Particle. (n.d.). MCU vs SoC vs microprocessor for IoT. Particle. https://www.particle.io/iot-guides-and-resources/mcu-vs-soc-vs-microprocessor-for-iot/

4. Shiksha. (n.d.). Microcontroller: Types, functions, uses, challenges and solutions. https://www.shiksha.com/online-courses/articles/microcontroller-types-functions-uses-challenges-and-solutions-blogId-155711

5. University of Kashmir. (n.d.). Embedded systems hardware architecture: Unit 1. https://cs.uok.edu.in/Files/79755f07-9550-4aeb-bd6f-5d802d56b46d/Custom/EmbeddedSysHardArch_Unit1.pdf

6. McGraw Hill Education. (n.d.). Embedded systems. https://www.mheducation.co.in/embedded-systems-9789353168025-india

7. Arm Community. (2013, October 14). 10 steps to selecting a microcontroller. Arm Developer. https://community.arm.com/arm-community-blogs/b/embedded-and-microcontrollers-blog/posts/10-steps-to-selecting-a-microcontroller

8. Microcontroller Tips. (2019, July 8). Key factors to consider when choosing a microcontroller. https://www.microcontrollertips.com/key-factors-consider-choosing-microcontroller/

9. Wikipedia contributors. (2025, April 27). Microcontroller. Wikipedia. https://en.wikipedia.org/wiki/Microcontroller

10. Wikipedia contributors. (2025, March 10). STM32. Wikipedia. https://en.wikipedia.org/wiki/STM32

11. Patil, A. B., & Patil, S. S. (2017). Microcontroller selection in embedded systems. ResearchGate. https://www.researchgate.net/publication/321081699_Microcontroller_Selection_in_Embedded_Systems

12. Baran, M., & Nilsson, M. (2022). Microcontroller selection for automotive embedded systems [Master's thesis, Halmstad University]. DIVA Portal. https://www.diva-portal.org/smash/get/diva2%3A1712830/FULLTEXT01.pdf

13. Rego, P. (2012, September 5). Leveraging the Kano model for optimal results. UX Magazine. https://uxmag.com/articles/leveraging-the-kano-model-for-optimal-results

14. Embedded Online Conference. (2022). Selecting the right microcontroller for your embedded application [PDF]. https://s3.amazonaws.com/embeddedonlineconference/eoc/sessions_slides/Selecting_the_Right_Microcontroller_for_your_Embedded_Application_EST.pdf