Microcontroller Ecosystem Analysis and Practical Application in the Simulation of a System for an Industrial Control Room

(June 2024)

Omar A. Torres, Yuly Alvear Romo y Leidy Castaño

***Summary*— This research focuses on carrying out an exhaustive analysis of the existing official literature on the ESP32 microcontroller. The main objective is to know and understand the ESP32 microcontroller, learn about its electrical characteristics, its functionalities, the mapping of its pins, how it is programmed, and the IDE that is required for its programming will be investigated. Although it is not a chip that requires a traditional operating system like the one a personal computer has, if it has an operating system called ROT (It is a "Real Time Operating System") that allows multitasking, well, we intend to understand How the RTO works during the execution of a program. Finally, a program will be created that remotely manages equipment applied to an industrial process.**

I. Introduction

The world today experiences constant change and seemingly endless technological revolutions. A revolution is barely understood when other transformations are already underway in the field of science and technology. These revolutions intertwine and complement each other to give rise to new products, services, technologies, and continuous advances in science. The electronics revolution is linked to software development, and both converge with artificial intelligence. Currently, the growing demand in the field of the Internet of Things (IoT) reflects the automation of various aspects of our environment.

In the field of IoT and industry, microcontrollers, as programmable devices, are essential to design systems that allow, for example, the control of temperature in the home, the saving of electrical energy, the creation of general-purpose equipment for the automation industry, electromedicine and countless more applications that we omit to mention due to the narrowness of this document.

In Colombia, the understanding and practical use of microcontrollers are limited mainly to the educational field, during academic training processes, however, we can point out that the use of microcontrollers in our country remains largely unexplored, awaiting the initiative of the entrepreneurs.

This work aims to establish a first contact with electronic technology, focusing particularly on the ESP32 microcontroller. The aim is to prepare a document that can function as a learning tool regarding the electronic ecosystem of the ESP32, its operating system, and its integration with other devices. In addition, a simple practice will be carried out to program the chip to simulate the operation of a central control room that allows the management of industrial equipment and processes. The central idea behind the control is that the device works using a client-server architecture so that an operator can connect remotely and manage the equipment applied to a process.

This work becomes relevant from the perspective of the systems engineering career since it provides engineers with a broader and less generic vision of the world of systems. It implies understanding that the concept of logical '1' or '0' is not simply an abstraction, but that behind it there is a real entity, where a logical '1' represents a fully functioning engine and a logical '0' is a machine off.

The study of the microcontroller and its domain opens the doors to a world of close and accessible opportunities. This knowledge not only enriches academic training but also provides the necessary skills to explore the various possibilities offered by the world of technology and electronics.

Manuscript received Jun, 9, 2024. The materials for this project were acquired by the authors, who are students of the Faculty of Engineering, Systems Engineering Program, subject of Operating Systems, University of Antioquia.

Omar Alberto Torres, Yuly Alvear Romo and Leidy Castaño, students of the Faculty of Engineering, University of Antioquia, Medellín, Colombia.

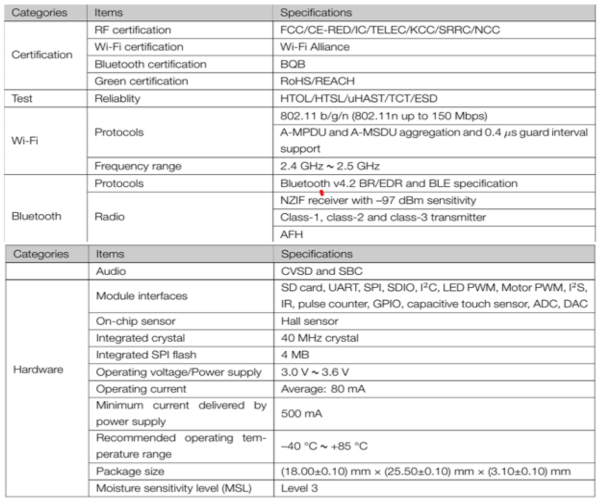
II. Theoretical Framework

A. *Microcontroller vs Microprocessor*

The differences between a microcontroller and a microprocessor focus on their design, functions, and applications. A microcontroller is an embedded device that combines a central processor, memory, and input/output peripherals on a single chip, designed to execute specific tasks in embedded systems. On the other hand, a microprocessor is an integrated circuit that includes only the central processing unit (CPU) and depends on external components such as memory and peripherals. While microcontrollers are used in a wide range of embedded devices and automated systems, microprocessors are common in personal computers and computer systems that require general processing capabilities. Additionally, microcontrollers tend to be more energy efficient, making them suitable for battery-powered devices or systems with energy efficiency requirements, while microprocessors typically consume more power and are used in systems with access to a constant power source, such as the electrical grid.

B. *ESP32 Microcontroller*

The ESP32, from an electronics perspective, is presented as a comprehensive microcontroller with an efficient and powerful design. By integrating a dual-core processor, memory, Wi-Fi, and Bluetooth modules, as well as various peripherals, this microcontroller simplifies circuit design and provides solid performance at 240 MHz The ESP32's wireless connectivity makes it easy to create computers that communicate with each other and wireless data transfer. On the other hand, its energy efficiency makes it ideal for battery-powered applications. With its versatility in peripherals, such as SPI, I2C, and UART ports, and its compatibility with Arduino, the ESP32 has become an attractive option for IoT projects. Additionally, it offers electronic designers a versatile tool backed by an active developer community and extensive online support.



Categories	Items	Specifications
Certification	RF certification	FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC
	Wi-Fi certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green certification	RoHS/REACH
Test	Reliablity	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with –97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
Categories	Items	Specifications
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I^2C, LED PWM, Motor PWM, I^2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC
	On-chip sensor	Hall sensor
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	4 MB
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating temperature range	–40 °C ~ +85 °C
	Package size	(18.00±0.10) mm × (25.50±0.10) mm × (3.10±0.10) mm
	Moisture sensitivity level (MSL)	Level 3

Fig.1 ESP32 technical specifications

C. *ESP32 architecture*

The ESP32-WROOM-32 is a powerful and versatile Wi-Fi+BT+BLE MCU module designed for a wide variety of applications, from low-power sensor networks to more demanding tasks such as voice coding, music streaming, and decoding. of MP3. At the core of this module is the ESP32-D0WDQ6 chip, designed to be scalable and adaptable. It has two CPU cores that can be controlled independently. The CPU clock frequency is adjustable from 80 MHz to 240 MHz

The ESP32 integrates a complete set of peripherals, including capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I²S and I²C. It offers built-in 802.11 b/g/n Wi-Fi connectivity and incorporates classic Bluetooth and Bluetooth Low Energy (BLE), making it versatile for various IoT applications. Additionally, it provides a wide variety of peripherals, including GPIO, UART, SPI, I²C, and PWM, as well as analog-to-digital converters (ADC) for measuring or generating analog signals. Security features are included. The operation and management of the CPU is done by the real-time operating system (RTOS), such as FreeRTOS for multitasking and efficient resource management. Designed to be efficient in terms of power consumption, the ESP32 is suitable for battery-powered devices.
Programming can be done in languages such as C or C++ using the Arduino integrated development environment (IDE) or the Espressif IDF (IoT Development Framework).

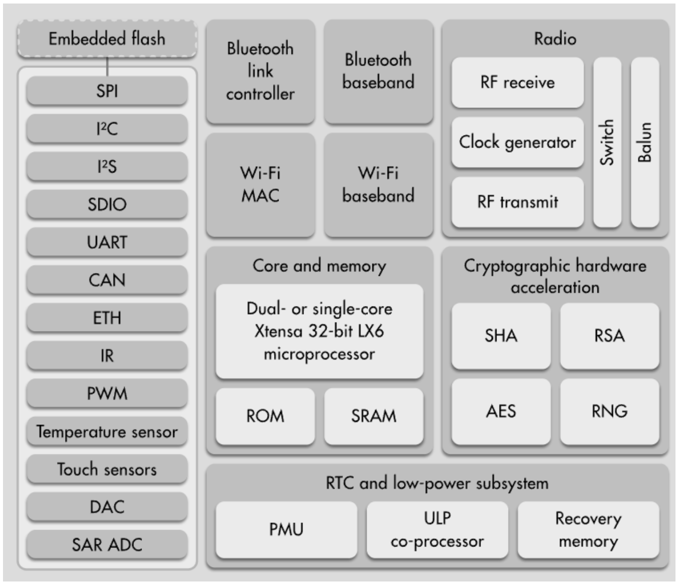


Fig. 2 ESP32 architecture

D. *ESP32 pin mapping*

The ESP32 uses pin mapping and multiplexing techniques to achieve multifunctionality of its GPIO (General Purpose Input/Output) pins. This means that the same pin can assume different roles or functions depending on the specific configuration and needs of the program. Through the pin configuration system, specific functions are assigned to the GPIO pins, such as analog input, digital output, I²C interface, SPI interface, UART, among others. This process is done by adjusting multiplexing registers (MUX) associated with each pin. In summary, thanks to these mapping and multiplexing techniques, a single pin on the ESP32 can be dynamically adapted to perform different tasks, providing flexibility and efficiency in resource management in pin-limited environments.

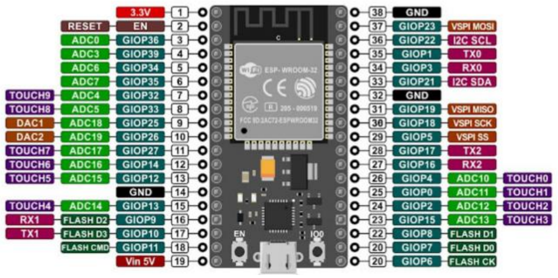


Fig. 3 Pin mapping on the ESP32

As for the input and output pins of the ESP32, most of the pins can be used as both digital inputs and outputs. However, it is notable that group pins 34 to 39 are used as input pins only. GPIO pins 6 to 11 are reserved for SPI flash connection and should not be used as GPIO. In addition, the ESP32 has 10 capacitive touch sensors that can be used as touch inputs, corresponding to GPIO pins 4, 0, 2, 15, 13, 12, 14, 27, 33 and 32.

The ESP32 also offers analog-to-digital and digital-to-analog converters on several of its pins, allowing for the digitization and conversion of analog signals.

On the other hand, the ESP32 integrates a Wi-Fi module that provides wireless connectivity capabilities, as well as support for the FreeRTOS real-time operating system (RTOS), which offers efficient and predictable task management in embedded environments.

E. *ESP32 programming*

To program the ESP32, several development tools make it easy to write, compile, and upload code to the microcontroller. Some of the most common tools to program the ESP32 are:

Arduino IDE – A development tool widely used for programming microcontrollers, including the ESP32. You can program the ESP32 using the official ESP32 for Arduino platform, which provides an easy-to-use interface and a large user community.

PlatformIO: is a development platform that integrates with different IDEs, such as Visual Studio Code and Atom. It offers support for a variety of boards, including the ESP32, and provides a more advanced interface with more powerful development features.

Espressif IDF (IoT Development Framework): is the official development framework from Espressif, the manufacturer of the ESP32, it is more advanced than Arduino solutions and provides direct access to the specific functions and features of the ESP32. It is suitable for developers looking for greater control over the code they are developing and the hardware.

MicroPython – is a Python implementation designed for microcontrollers, including the ESP32. It allows the ESP32 to be programmed in Python, which can be beneficial for those familiar with this language.

NodeMCU/Lua: for those who prefer to use Lua, NodeMCU is a platform that allows the ESP32 to be programmed using a scripting language.

Each tool has its advantages and disadvantages, and the choice largely depends on your preferences and development requirements. The Arduino IDE is popular for beginners due to its simplicity, while the Espressif IDF provides greater control and access to advanced features for more experienced developers. The choice may also depend on the user community, the documentation available, and the specific features needed by the project.

F. *ESP32 as server*

The ESP32, thanks to its connectivity capacity through the TCP/IP protocol, can function as a web server. This means it can receive requests and send responses over the network, making it a flexible tool for the Internet of Things (IoT) and remote-control applications.

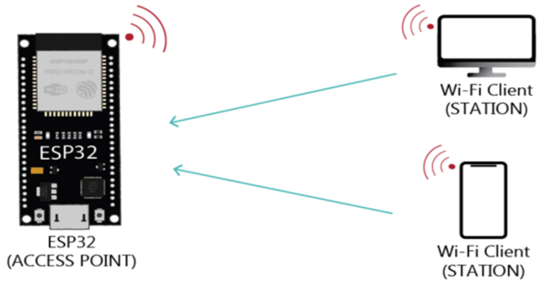


Fig. 4 ESP32 connectivity

III. Methodology

The methodology followed for the analysis and simulation of an industrial control room with the ESP32 included several key stages. First, a thorough review of available documentation, including datasheets and manuals, was carried out to understand the architecture and functionalities of the ESP32. Then, the development environment was configured using the Arduino IDE 2.3.2, which allowed the development of the program for the ESP32. Subsequently, the hardware was assembled using a breadboard, where the electronic components were placed, and the circuits controlled by the ESP32 were tested. The final stage involved programming and/or developing the HTML code for the graphical interface plus the logic to control the machines in an industrial control room, written so that the device operates as a server.

The next step was the development of the necessary tests and adjustments to ensure the proper functioning of the system.

Next, to save the program in the ESP32 memory, the need to put the device in programming mode to load the new firmware was highlighted. Once in programming mode, the Arduino IDE was used to write, compile, and upload the program. Finally, exhaustive tests of the system were carried out, such as the connection, of clients to the device to access the Virtual control room application, verifying that both the start and stop of the desired equipment could be achieved, and tests were carried out at different distances from the device.

A. *Initial Acquisition of the ESP32*

When you first purchase an ESP32 module, it typically arrives in its factory-default state, ready to be programmed to your specific needs. The following describes how it usually arrives and the need to put it into programming mode.

Factory State: When you receive a new ESP32, it usually comes with basic factory firmware installed. This firmware can vary by manufacturer but is usually simple enough to allow the programming of the device.

Ready to Program: The ESP32 does not contain specific user code at the beginning, which means that it is ready to be programmed with the firmware you want to load into it. This provides flexibility to adapt the ESP32 to specific needs.

Programming Mode: To load new firmware into the ESP32, it is generally necessary to put the device into programming mode. This mode allows communication with the programming tool and loading of the new code into the ESP32.

Specific Pin Connection: to activate the programming mode, certain pins of the ESP32 are connected or disconnected depending on the specific variant of the module. These pins typically include GPIO0 and GPIO2, among others. The exact pin combination may vary by manufacturer and model.

Programming Interface: When you put the ESP32 in programming mode, it enters a state that allows communication with the programming tool through interfaces such as UART or USB. This step is essential to upload the user firmware to the ESP32.

The need to put the ESP32 into programming mode lies in the ability to allow new firmware to be loaded in a safe and controlled manner. By connecting or disconnecting certain pins, the ESP32 enters a specific state that enables the programming interface and facilitates the user's code-loading process. This model is a fundamental feature that ensures the flexibility and customizability of the ESP32 for different applications.

B. *ESP32 W-ROOM-32 programming mode*

The procedure to put the ESP32 microcontroller in the programming state is as follows:

1. Open the Arduino IDE.
2. Go to the tools tab in the menu bar and select the ESP32 card reference, which in our case was the ESP32 Dev Module.
3. In tools, select the USB port. In our case, COM port 6 was selected.
4. Compile the program to run.
5. With the card connected to the COMS port 6 of the computer, and energized, start loading the program. The IDEE starts the compilation while generating messages in the Arduino console, just when the message that the connection is being established appears. communication, press the boot button that comes on the ESP32 card, hold it down for a moment, and release it, the program is thus loaded with the user's program.

IV. Implementation

The implementation consisted of the development of a comprehensive system that emulates a control center in an industrial environment, using the ESP32 as the main management device and a basic web interface for interaction with the system through a local server.

The equipment subject to control includes: "Lighting", "Oven FC01", "Band CV03", "Fan Fa32 Bottom of the Oven", "Pump PP321", "Emergency Pump PP333", "Fire Alarm" and "CCC Emergency ".

Security logic was developed to ensure the safe operation of the system. For example, turning on the oven is only possible if the oven bottom fan (FA32) is activated and at least one of the water pumps (PP321 or PP333) is running. If any of this equipment goes off, the melting furnace (FC01) must stop automatically to prevent risk situations.

Los equipos sujetos a control comprenden: "Alumbrado", "Horno FC01", "Banda CV03", "Ventilador Fa32 Fondo del Horno", "Bomba PP321", "Bomba de Emergencia PP333", "Alarma Contra incendio" y "Emergencia CCC".

V. Results

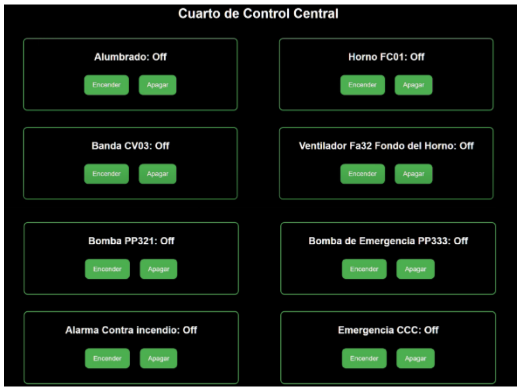


Fig. 5 Graphical interface showing the name of the machine with its on and off buttons and a text that confirms the status of the machine.

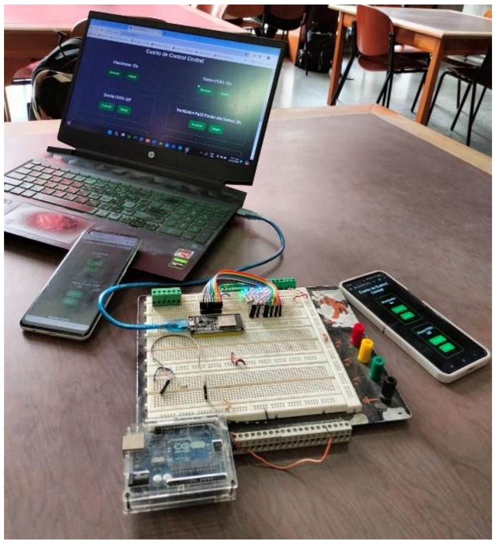


Fig. 6 Assembly in operation with 3 devices connected to the same Wi-Fi network.

The results obtained show that the ESP32 performs effectively as a web server thanks to its connectivity capacity through the TCP/IP protocol. In the proposed exercise, the ESP32 was programmed to connect to a private Wi-Fi network when powered on. Once connected to the network, the microcontroller is available for clients to access using the assigned IP address.

It is important to note that programming the ESP32 as a web server is relatively simple, largely due to the availability of the "<wifi.h>" library. This greatly simplifies the configuration process and allows you to quickly take advantage of the ESP32's server capabilities in practical applications.

VI. Conclusions

During the development of this work, the proposed objectives were successfully achieved. Literature related to the ESP32 microcontroller, provided by manufacturers, was extensively explored, allowing us to delve into the broad hardware and software ecosystem of this device.

In the process, we gained detailed knowledge about communication protocols, analyzed the electrical characteristics of the chip, and thoroughly understood pin mapping. This mapping allowed us to assign various functions to the same pin, either as input, output, PWM output, or communication functions.

Additionally, we dive into the fundamental elements of chip programming, establishing a valuable connection between this knowledge and our training as systems engineers.

It is an outstanding achievement to successfully implement a system that simulates the operation of a control room in an industrial company. The designed application allows an operator to connect to monitor and control a process.

This simulation approach finds its relevance in the field of the Internet of Things (IoT), where interconnection and remote management are essential.

The achievement of this work was possible thanks to the support and commitment of the team, underscoring the importance of collaboration in projects of this nature.

In the horizon of potential advances in this project, several directions are suggested that could enrich and optimize the simulation of the industrial control room based on the ESP32 microcontroller. Considering the academic and research context, the following areas of exploration could be of interest:

A. Research in sensors and monitoring:

Explore the integration of additional sensors to expand the monitoring capability of the simulated industrial environment. This could include research on the proper selection of sensors and their communication protocols.

B. Development of specific communication protocols:

Research and develop specific communication protocols for the efficient management of a wider range of industrial devices. This could involve adapting existing standards or creating new communication approaches.

C. Optimization of graphical interfaces:

Explore improvements to graphical user interfaces to increase efficiency and usability. This could include research in data visualization, process graphing, and user-centered design.

D. Integration of emerging technologies:

Investigate the feasibility of incorporating emerging technologies, such as Machine Learning or Artificial Intelligence, to improve the predictive and adaptive capacity of the system. Evaluate how these technologies could contribute to operational efficiency and decision-making.

E. Development of advanced security measures:

Research and develop advanced security measures, such as multi-factor authentication, strong encryption, and continuous monitoring systems. These aspects are crucial in industrial environments where safety is a top priority.

F. Research in energy optimization strategies:

Explore energy optimization strategies to efficiently manage the energy consumption of the controlled devices. Investigate possible alternative energy sources and their implementation in the system.

G. Integration with cloud platforms:

Investigate the feasibility and benefits of integration with cloud platforms. This would allow data to be stored centrally, facilitating long-term trend analysis and remote access to critical information.

H. *Development of Complementary Mobile Applications:*

Research and develop complementary mobile applications that extend monitoring and control capabilities to mobile devices. Evaluate the implementation of additional functionality to improve operational flexibility.

These perspectives offer a framework for the continued research and development of this project, providing opportunities for significant contributions in academia and technology.

References

[1] Carrasco, D. (2021, abril 25). WLAN en ESP32: Primeros pasos con el Wifi. ElectroSoftCloud. https://www.electrosoftcloud.com/wlan-en-esp32-primeros-pasos-con-el-wifi/

[2] Barry, R., & The FreeRTOS Team. (n.d.). Mastering the FreeRTOS™ Real Time Kernel: A Hands-On Tutorial Guide. FreeRTOS. https://freertos.org/Documentation/Mastering-the-FreeRTOS-Real-Time-Kernel.v1.0.pdf

[3] Pinillos, J. M. (2020, mayo 7). Básicos ESP32: Mapeo de pines y sensores internos. Tecnotízate. https://tecnotizate.es/esp32-mapeo-de-pines-y-sensores-internos/

[4] Punto Flotante S.A. (2022). Manual básico NodeMCU ESP32 DevKit V1 Arduino IDE. https://www.puntoflotante.net/MANUAL-BASICO-NODEMCU-ESP32-ARDUINO.pdf

[5] Espressif Systems. (n.d.). ESP-WROOM-32 Datasheet. Retrieved from https://www.alldatasheet.com/view.jsp?Searchword=Esp-wroom-32%20datasheet&gad_source=1&gclid=CjwKCAiAopuvBhBCEiwAm8jaMUrA9d3ZAeRBJZGn3q_aJnhFT3ny3V6c93gDRD1OtHYEOarCMxLaWRoCNPYQAvD_BwE

[6] Electronics. (2021, julio). Especificaciones del módulo ESP32. Vasanza. https://vasanza.blogspot.com/2021/07/especificaciones-del-modulo-esp32.html

[7] Espressif Systems. (s.f.). ESP32. https://www.espressif.com/en/products/socs/esp32