Embedded System for Monitoring and Watering Indoor Plants with Web Interface Using ESP32

Kamelia Raynova, Veselin Gospodinov

Abstract – **This paper presents the design and implementation of an embedded system for automated monitoring and irrigation of indoor plants, using ESP32 microcontroller. The system integrates soil moisture sensor, temperature sensor, and air humidity sensor. The collected data is processed in real time and made accessible through web-based interface, allowing users to remotely monitor via Wi-Fi. The proposed system aims to enhance plant care for individuals with limited time or experience.**

Keywords – **automated, ESP32, sensor, web, Wi-Fi**

I. INTRODUCTION

In today's dynamic and high-tech society, technology plays an increasingly central role in everyday life, transforming the way we interact with the world around us. Along with technological advancements, growing environmental awareness and urbanization are increasing people's desire to bring a bit of nature into their homes and workspaces. Growing houseplants brings aesthetic pleasure, improves air quality, and contributes to a better psychological environment. Indoor plant care often requires regular monitoring and watering, which can be challenging for individuals with busy schedules or limited gardening experience. This project addresses these challenges by developing a smart, automated system that simplifies and enhances plant maintenance using modern technology. The system is built around the ESP32 platform, chosen for its low power consumption, integrated Wi-Fi and Bluetooth capabilities, and robust processing power. It incorporates multiple sensors to monitor key environmental parameters such as soil moisture, ambient temperature, and air humidity. Based on sensor readings, the system can autonomously activate a watering mechanism when the soil moisture falls below a predefined threshold, ensuring plants receive adequate hydration without manual intervention. Additionally, a web-based interface provides users with real-time access to sensor data and system status, enabling remote monitoring from any internet-connected device. The solution is designed to be cost-effective, scalable, and user-friendly, making it suitable for home applications. By automating essential plant care tasks, the system promotes healthier plant growth.

II. SURVEY OF EXISTING PAPERS

A wide range of research has explored IoT-based irrigation and agricultural automation systems, employing diverse technologies and implementation techniques. Several relevant studies have been reviewed to identify different approaches to monitoring and control. For instance, the work presented in paper [1] describes an efficient automation solution utilizing a Raspberry Pi, environmental sensors, and a camera module to gather real-time agricultural data. It further employs ThingSpeak as the IoT platform for cloud-based data storage and visualization, enabling the development of data-driven applications. In paper [2], a smart farming system is implemented using a CC3200 single-chip microcontroller to monitor environmental parameters such as temperature and humidity, while an attached camera captures field images and transmits them to the farmer via MMS over Wi-Fi. Paper [3] centers on the use of Bluetooth technology for short-range wireless communication in field monitoring applications, whereas paper [4] proposes a GSM-based greenhouse monitoring system, where remote supervision is facilitated through GPRS connectivity. Furthermore, the research in paper [5] introduces a cloud-oriented IoT architecture applicable across various precision agriculture scenarios. This architecture includes a sensing layer for data acquisition, a gateway layer that ensures connectivity to the internet, and a back-end layer responsible for data management, analysis, and automated agricultural decision-making.

Overall, the reviewed literature demonstrates a clear progression toward improved connectivity and autonomous agricultural control. However, several recurring limitations can be identified. Many systems are evaluated only in limited or controlled environments, lacking long-term field validation and comprehensive performance analysis. Additionally, most solutions exhibit only partial automation, with minimal use of advanced data analytics, optimization algorithms, or reliable fail-safe mechanisms.

In response to these gaps, the current paper introduces a compact, low-cost IoT system designed specifically for small-scale, indoor domestic farming environments. The work introduces a complete end-to-end prototype, covering the full development flow from hardware component selection and breadboard assembly to the implementation of a detailed software architecture with SPIFFS-based file hosting. The system operates in Wi-Fi Access Point (AP) mode, enabling the ESP32 to create its own standalone wireless network and locally host a web server. Users connect directly to the device without requiring internet access, account registration, or external infrastructure, greatly simplifying deployment. The solution integrates

Received: 25.10.2025
Published: 08.11.2025
https://doi.org/10.47978/TUS.2025.75.03.002

Kamelia Raynova is with the Technical University of Sofia, Sofia, 1000, 8 Kl. Ohridski Blvd, Bulgaria (kkaneva@tu-sofia.bg)

Veselin Gospodinov is with the Technical University of Sofia, Sofia, 1000, 8 Kl. Ohridski Blvd, Bulgaria (vegospodinov@tu-sofia.bg)

sensors for monitoring five key environmental and resource-related parameters—air temperature, air humidity, soil moisture, light intensity, and water reservoir level—and performs closed-loop automation by activating the water pump only when necessary. Additional protective logic, such as preventing the pump from operating when the reservoir is empty, enhances reliability and safety. User interaction is facilitated through any standard web browser on a mobile device or computer, ensuring intuitive, platform-independent operation without the need for a dedicated application. Thus, the proposed system is presented as a fully functional and coherent framework that contributes a practical and accessible approach to smart irrigation and indoor agriculture (Fig.1).

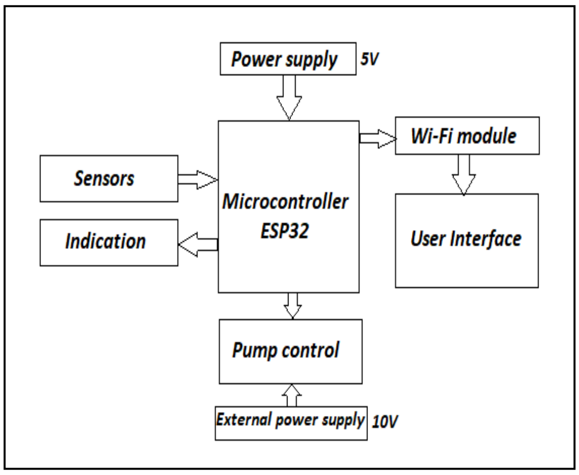


Fig. 1. Block Scheme of the System.

III. System Design

Automated plant care systems are designed to simplify and enhance the cultivation process by continuously monitoring key environmental parameters and performing automatic corrective actions, most commonly irrigation. For healthy plant growth, a well-balanced environment is essential, as even small deviations from optimal conditions can slow development or damage the plant. Among the parameters most monitored by intelligent horticultural systems, soil moisture stands out as the most critical factor. Both insufficient watering (leading to drought stress) and excessive watering (leading to root suffocation and fungal growth) can be harmful. Soil moisture can be quantified either directly as volumetric water content (%) or indirectly through electrical properties of the soil. Two major sensor technologies are used in IoT-based agriculture: resistive and capacitive soil moisture sensors.

The resistive sensor (Fig. 2) determines moisture levels by measuring the resistance between two exposed electrodes inserted into the soil. Its advantages include low cost, compact size, and ease of integration with microcontrollers. However, prolonged exposure to moisture causes electrode corrosion, resulting in deteriorating accuracy and a shorter operational lifespan. In addition, resistive measurements are significantly influenced by soil salinity, which can lead to unreliable data in fertilized or mineral-rich soils.

The capacitive soil moisture sensor (Fig. 3) represents a more advanced and reliable solution. It measures changes in dielectric permittivity, which increases with soil water content. Because its electrodes are insulated, it is resistant to corrosion and demonstrates greater long-term stability. Furthermore, it is considerably less sensitive to salinity, making it better suited for continuous precision agriculture applications. Due to these advantages, capacitive sensors are widely adopted in modern IoT systems that require durability and accuracy [6].

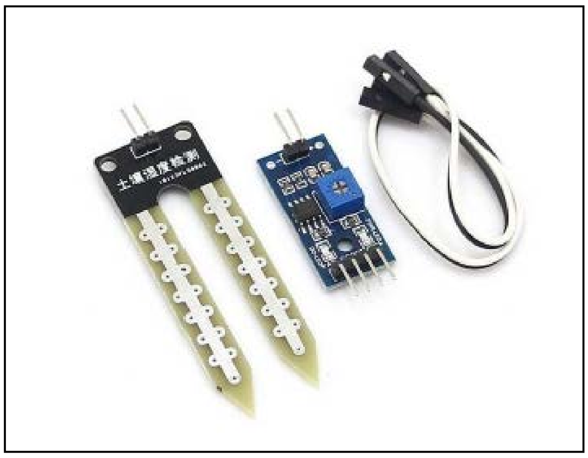


Fig. 2. Resistive Soil Moisture Sensor.

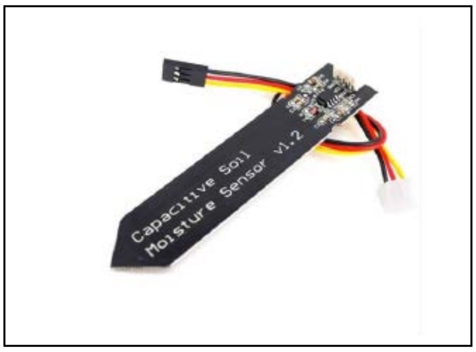


Fig. 3. Capacitive Soil Moisture Sensor.

Air temperature is another crucial factor directly affecting photosynthesis, transpiration, and respiration rates. Most indoor ornamental plants thrive in temperatures ranging from 18°C to 24°C, while prolonged exposure to values outside this range can reduce metabolic activity or cause irreversible damage. Air humidity also plays a vital role in regulating transpiration — the process by which water evaporates from the leaves. Extremely low humidity leads to dehydration and leaf-edge browning, whereas excessively high humidity (above 80–85%) promotes microbial growth and fungal disease. To monitor these important environmental conditions, a DHT11 sensor was selected (Fig. 4). It provides a simple, cost-effective, and sufficiently accurate solution for real-time indoor temperature and humidity measurement.

Light intensity serves as the primary energy source for photosynthesis. Plants differ significantly in their light requirements, ranging from high-sunlight species to those adapted for shaded environments. Therefore, monitoring illumination levels is important for ensuring proper photoperiod and maintaining optimal plant productivity. Light is commonly quantified in lux (lx), measuring human-perceived brightness, or in PPFD (Photosynthetic Photon Flux Density, μmol/m^2/s), representing photosynthetically active radiation essential for plant growth [7]. In this project, the BH1750 digital light sensor (Fig. 5) is used due to its precise lux measurement and seamless I²C communication interface, making it ideal for indoor IoT applications.

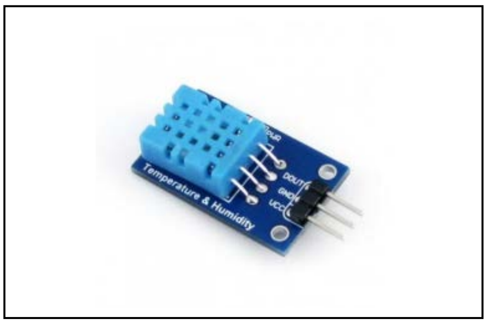


Fig. 4. DHT11 Sensor.

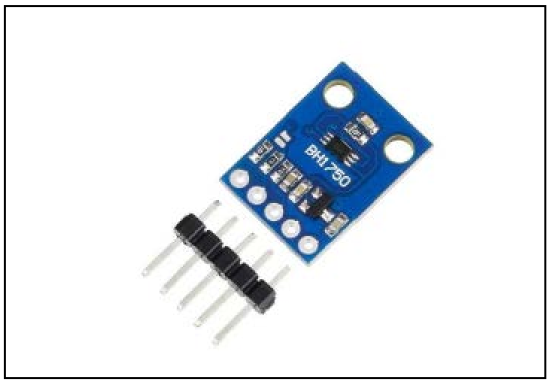


Fig. 5. BH1750 Sensor.

The ESP32 microcontroller was chosen as the central computational and communication unit due to its outstanding balance of processing power, connectivity, energy efficiency, and affordability, making it particularly suitable for IoT-based automation systems. It integrates both Wi-Fi (802.11 b/g/n) and Bluetooth (Classic + BLE) within a single compact chip (Fig. 6), eliminating the need for external communication modules. This significantly reduces the overall hardware cost, simplifies circuit design, and decreases power consumption — all essential requirements for small-scale smart agriculture systems that are intended to operate continuously and autonomously.

From a performance perspective, the ESP32 is equipped with a dual-core Tensilica LX6 processor running at up to 240 MHz, alongside 520 KB of SRAM, enabling true multitasking capabilities. This allows the device to concurrently handle multiple responsibilities, including real-time data acquisition from sensors, execution of control algorithms, wireless communication, and the operation of an embedded web server for user interaction. Additionally, optional deep-sleep modes greatly extend potential battery life in portable or low-power deployments, while maintaining periodic monitoring functionality.

Another major advantage of the ESP32 lies in its rich peripheral support, which contributes to highly flexible hardware integration. The controller offers numerous GPIO pins, multiple ADC channels suitable for interfacing with analog soil moisture sensors, an I²C bus for digital sensors such as the BH1750, PWM channels for controlling components like pumps, buzzers, and indicator LEDs, as well as UART interfaces for system configuration and diagnostic output. This extensive feature set minimizes the need for external supporting circuitry, allowing the device to function as a fully independent and scalable automation platform.

Equally important is the maturity of the ESP32 development environment. It is widely supported in the Arduino IDE and other popular open-source tool chains, providing access to well-maintained libraries, documentation, and examples that streamline software development. A large and active community contributes continuously to improvements, troubleshooting resources, and best-practice tutorials, making the platform beginner-friendly while still powerful enough for advanced engineering applications.

A comparison with alternative platforms further highlights the strategic advantages of the ESP32. For instance, the Arduino Uno, while widely adopted in educational projects, lacks built-in wireless connectivity and offers limited processing performance and memory capacity, making it unsuitable for real-time web hosting or multitasking operations. On the opposite end of the spectrum, the Raspberry Pi family offers significantly greater computational resources but introduces unnecessary complexity, higher power consumption, a full operating system overhead, and increased cost for a task that mainly relies on real-time monitoring and control of embedded peripherals.

Considering all the above factors, the ESP32 stands out as the optimal solution for the proposed IoT-based plant monitoring and automation system. Its unique balance of cost-effectiveness integrated wireless connectivity, computational capability, peripheral diversity, and developer ecosystem makes it a robust and future-proof foundation positioned perfectly between underpowered microcontrollers and overengineered single-board computers.

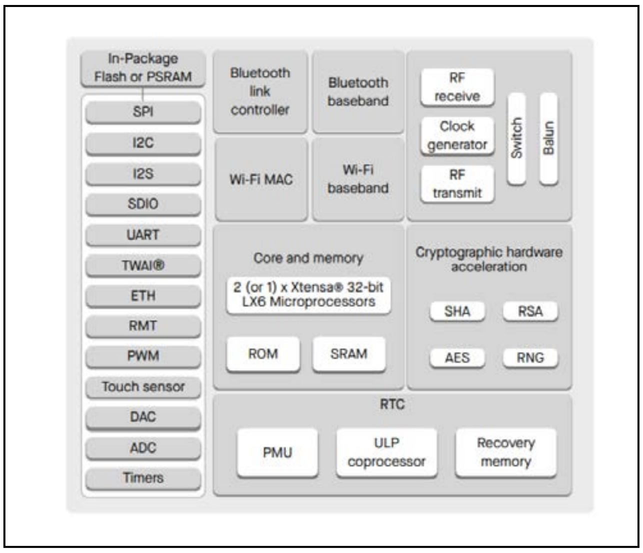


Fig. 6. Functional Block Diagram of ESP32.

The pumping subsystem is controlled using an L298N H-bridge motor driver (Fig. 7), a widely available and inexpensive module capable of operating DC motors at voltages higher than the logic supply, while providing the necessary current that the ESP32 cannot deliver directly. Although the driver supports bidirectional control and PWM-based speed regulation, only one direction and basic on/off switching are required in this design [8].

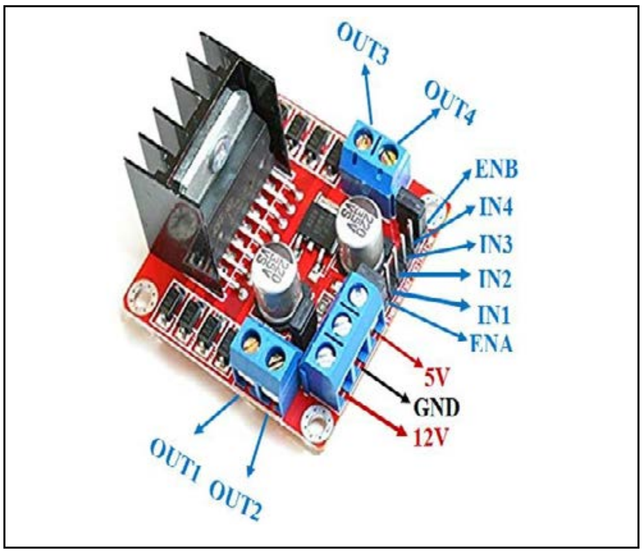


Fig. 7. L298N Motor Driver.

For audible signaling, a passive buzzer is employed as the sound-generating component of the system. Passive buzzers are extremely cost-efficient and require only a modulated signal generated through PWM or tone functions on the ESP32 to produce different auditory alerts. This enables flexible customization of sound patterns, such as short beeps for normal notifications or continuous alarms for critical states, enhancing the usability of the system by providing immediate feedback even when the user is not actively observing the interface. Since the buzzer contains no internal oscillation circuitry, it draws minimal current and has a very simple electrical interface, making it ideal for compact embedded solutions.

For visual signaling, standard light-emitting diodes (LEDs) are used, as they represent the simplest, most energy-efficient, and cost-effective form of real-time visual feedback. Multiple LEDs of different colors are assigned to indicate various operational states of the system — such as power status, irrigation activation, low water level warnings, and sensor fault alerts. They are interfaced with the ESP32's digital output pins through current-limiting resistors to ensure proper control and protect both the LED components and the microcontroller pins from excessive current draw. The combination of auditory and visual indicators ensures that users can quickly interpret the system's condition without needing to access the web interface, which is valuable for diagnostics and accessibility.

Power management is a crucial aspect of the system design, and the entire architecture is optimized to be powered directly through the ESP32's Micro-USB port, using a common 5V USB charger or a computer's USB output. The ESP32 development board includes an onboard voltage regulator that converts the 5V input into 3.3V, supplying the necessary voltage to the microcontroller and the majority of the low-power digital sensors. Components such as the BH1750 light sensor operate natively at 3.3V and are thus directly powered from this regulated output. The DHT11 temperature and humidity sensor and the resistive soil moisture sensor can operate at either 3.3V or 5V, allowing flexible integration depending on wiring and current requirements. To reduce complexity and maintain consistent voltage levels across the sensing network, all compatible sensors are powered through the 3.3V output rail.

The L298N motor driver requires two separate electrical inputs: a 5V logic supply (VCC) and a motor supply voltage (VMS). The logic supply is sourced from the ESP32 board's 5V (Vin) output, ensuring direct communication with the control signals. The water pump, being a low-voltage DC motor with an operating range of 3–5V, can also be powered through the same 5V (Vin) line, provided that the USB power source is capable of delivering sufficient current — particularly the ~1A peak current demanded during motor start-up. However, for improved system reliability and to avoid voltage dips that might cause the ESP32 to reset, a dedicated external 5V supply may be used for the pump. In that case, the ground of the external supply must be connected to the ESP32's ground to establish a common reference point for proper signal communication and stable operation.

The pump is connected to the output terminals of the L298N (OUT1/OUT2), allowing the driver to deliver the required current and manage switching operations safely. Meanwhile, the buzzer and LEDs draw only negligible current and are powered directly from the ESP32 I/O pins, ensuring that their operation does not significantly impact the overall power budget.

IV. Software Development

The Arduino IDE was selected as the primary software development platform due to its exceptional accessibility, user-friendly interface, and mature support ecosystem for the ESP32 through the additional Board Manager package. The environment provides a vast collection of pre-built libraries for sensor communication, Wi-Fi networking, file system management, motor control, and web server hosting, which significantly accelerates the development process by eliminating the need for low-level driver implementation. While more advanced environments such as PlatformIO or Espressif's native ESP-IDF framework offer powerful debugging tools, fine-grained memory control, and RTOS-level task management, the simplicity and educational value of the Arduino IDE align perfectly with the goals of this prototype — easy setup, rapid prototyping, and wide accessibility for users with varying levels of programming experience.

Arduino programming is based on the C++ language, enriched by simplified high-level abstractions provided by the Arduino Framework. This approach retains the efficiency and hardware-level control characteristic of C++, while offering ready-to-use functions and clear structure that support both procedural and object-oriented programming paradigms. Such flexibility is essential for embedded systems, where performance optimization and maintainable code organization are equally critical.

The software architecture follows a monolithic design pattern, running as a single integrated program within the standard Arduino dual-function structure: the initialization routine (setup()) and the continuously executed main loop (loop()). Although the ESP32 supports multitasking through FreeRTOS, introducing task scheduling and independent threads of execution, this prototype intentionally avoids such complexity to improve readability and reduce development overhead.

During setup (), all hardware components and services are initialized — GPIO pin direction assignment, DHT and BH1750 sensor configuration via I²C, activation of the serial debugging interface, creation of a Wi-Fi Access Point (AP) network, and the startup of the embedded web server. In the loop(), sensor data is periodically sampled, validated to prevent errors (e.g., through isnan() checks), visually and audibly signaled if thresholds are exceeded, and used to make automated decisions such as activating the water pump. A combination of built-in and external libraries enables this functionality. Global variables store real-time sensor readings, user-defined configuration values (such as irrigation thresholds), and system status flags, allowing efficient access throughout program execution.

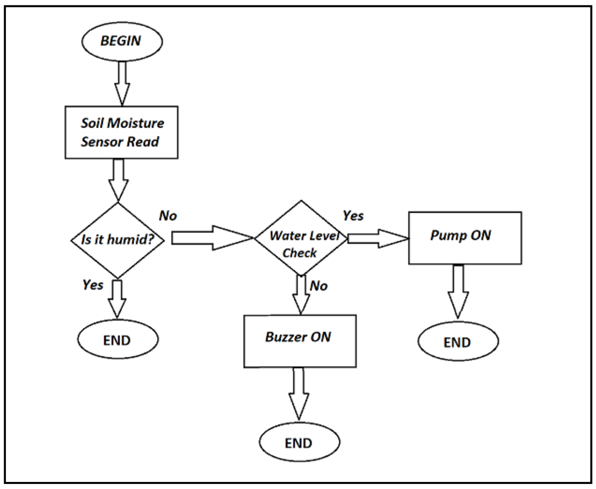


Fig. 8. Algorithm Block Scheme

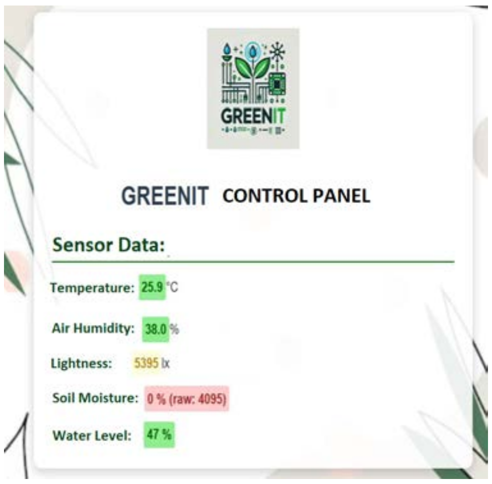
The communication architecture of the system is based on a Wi-Fi Access Point (AP) configuration. In this mode, the ESP32 autonomously establishes its own wireless network by broadcasting a predefined SSID and password. Any device such as a smartphone, laptop, or tablet can connect directly to this network without requiring existing Wi-Fi infrastructure, an active router, or an internet connection. This ensures complete autonomy and local accessibility — a key advantage for indoor gardening systems where internet availability may be limited. The primary trade-off of this approach is that when connected to the ESP32 network, the user's device temporarily sacrifices access to external internet services. However, this limitation is considered acceptable given the system's target use case, where simplicity, privacy, and fully offline operation are prioritized.

Once connected, the user interacts with the device via an embedded web server hosted directly on the ESP32. The server, operating on the standard HTTP port 80, is implemented using the Arduino WiFiServer library, ensuring compatibility with practically all modern browsers. The communication between the webpage and the system logic relies on dynamic manipulation of the Document Object Model (DOM) using JavaScript. When the index.html file is loaded in a browser, the HTML elements are parsed and converted into the DOM tree structure. JavaScript execution is triggered only after the DOMContentLoaded event fires, guaranteeing that all interface elements are fully defined before the script attempts to interact with them. At that point, the script retrieves specific elements — such as the temperature or soil moisture display fields — using document.getElementById() and updates their content in real time based on live sensor readings received from the ESP32. These updates are rendered immediately in the browser, creating a smooth and continuously refreshed interface without requiring manual page reloads.

The user interface is intentionally designed to be simple, intuitive, and universally accessible. It is delivered as a lightweight web page stored in and served from ESP32's internal SPIFFS file system, eliminating the need for external hosting platforms or dedicated mobile applications. The visual structure is defined in index.html, where responsiveness is ensured using the viewport meta tag, providing correct scaling on various screen formats, especially mobile devices. Content is arranged within a central <div class="container"> layout to maintain clear organization and readability.

Sensor data visualization forms the core of the interface. Each monitored parameter — air temperature, relative humidity, light intensity, soil moisture, and water reservoir level — is represented by a labeled <p> element containing a <span> tag with a unique identifier (e.g., <span id="temperature">). These identifiers enable direct JavaScript control over display updates. Styling is achieved through an external style sheet (style.css), which emphasizes a clean and functional aesthetic. Visual cues are used to enhance user awareness: elements dynamically change background or text color when values exceed predefined thresholds, allowing users to quickly determine whether a parameter is within a healthy range immediately.

Once development and testing are complete, the software is compiled and uploaded to the ESP32 via the Arduino IDE. During this process, the correct development board (ESP32 Dev Module) and serial communication port are selected to ensure proper deployment. This rapid upload-and-test workflow enables efficient iteration throughout the prototype development cycle.


GREENIT CONTROL PANEL

Sensor Data:

Temperature: 25.9 °C

Air Humidity: 38.0 %

Lightness: 5395 lx

Soil Moisture: 0 % (raw: 4095)

Water Level: 47 %

Fig. 9. User Interface.

V. CONCLUSION

The developed prototype successfully demonstrates the feasibility and practical benefits of an affordable, locally managed smart plant care system that integrates environmental monitoring with automated control actions. By combining multiple key parameters — including soil moisture, air temperature, humidity, light intensity, and water reservoir status — the system provides a holistic overview of plant health, enabling timely and efficient irrigation decisions. The implementation of the ESP32 as the central processing and communication unit, particularly with its built-in Wi-Fi module operating in Access Point mode, ensures that users can easily interact with the system directly through their personal devices without reliance on external network infrastructure, subscription-based platforms, or cloud connectivity. This independence from the internet enhances accessibility, reduces complexity, and increases reliability, particularly in indoor home-based settings where simplicity and ease of setup are highly valued. The selected sensor suite has proven to be well-matched to the fundamental requirements of domestic plant maintenance. Real-time visualization and automated decision-making support users in maintaining optimal growing conditions, even in the absence of gardening expertise. Despite the current interface being relatively simple, the browser-based local web interface remains intuitive, responsive, and fully adequate for real world use. The modular nature of the system design — both in hardware and software — provides a strong foundation for future development and encourages iterative improvements. Practical enhancements such as refined data visualization, adjustable automation thresholds, scheduled irrigation cycles, or integration of historical trend analysis could further increase usability and intelligence. Additionally, the system’s flexibility creates numerous opportunities for expansion into more advanced features and broader applications. Examples include harvesting and processing data analytics through external cloud services (when desired), extending environmental sensing with CO_2 or pH measurements, enabling remote access through home network integration, or incorporating renewable energy through solar power for increased sustainability and portability.

Furthermore, the architecture could be adapted to support larger-scale agricultural deployments or multi-zone household greenhouse environments. Overall, the project successfully addresses a real and growing need for smart home horticulture solutions. It clearly demonstrates how modern embedded systems and IoT technologies can be leveraged to deliver intelligent, cost-effective, and user-friendly automation tools that promote plant health and reduce the burden of routine care. By bringing digital control into everyday gardening activities, the prototype contributes to fostering environmental awareness, enhancing convenience, and strengthening the connection between individuals and nature within their own homes. It therefore represents not only a functional achievement, but a valuable platform for continued innovation in smart agriculture.

Acknowledgment

The authors would like to thank the Research and Development Sector at the Technical University of Sofia for the financial support, project 251ИХЪБ0002-09.

References

[1] J. Boobalan, V. Jacintha, J. Nagarajan, K. Thangayogesh and S. Tamilarasu, "An IOT Based Agriculture Monitoring System," 2018 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2018, pp. 0594-0598, doi: 10.1109/ICCSP.2018.8524490.

[2] S. R. Prathibha, A. Hongal and M. P. Jyothi, "IOT Based Monitoring System in Smart Agriculture," 2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT), Bangalore, India, 2017, pp. 81-84, doi: 10.1109/ICRAECT.2017.52.

[3] R. K. Jha, S. Kumar, K. Joshi and R. Pandey, "Field monitoring using IoT in agriculture," 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Kerala, India, 2017, pp. 1417-1420, doi: 10.1109/ICICICT1.2017.8342777.

[4] Ji-chun Zhao, Jun-feng Zhang, Yu Feng and Jian-xin Guo, "The study and application of the IOT technology in agriculture," 2010 3rd International Conference on Computer Science and Information Technology, Chengdu, 2010, pp. 462-465, doi: 10.1109/ICCSIT.2010.5565120.

[5] A. Khattab, A. Abdelgawad and K. Yelmarthi, "Design and implementation of a cloud-based IoT scheme for precision agriculture," 2016 28th International Conference on Microelectronics (ICM), Giza, Egypt, 2016, pp. 201-204, doi: 10.1109/ICM.2016.7847850.

[6] Hrisko, Joshua, Capacitive Soil Moisture Sensor Theory, Calibration, and Testing., July 2020, DOI: 10.13140/RG.2.2.36214.83522

[7] ROHM Semiconducor, “Digital 16bit Serial Output Type Ambient Light Sensor IC”, https://www.mouser.com/datasheet/2/348/bh1750fvi-e-186247.pdf?srsltid=AfmBOoqoFzOqVP_SKaAKVzwnixx-f_K-lCkhkot4CBRHRG4s1YJGcT_u

[8] Handson Technology, “L298N Dual H-Bridge Motor Driver”, https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf