“SmartGarden: An Economical IoT-Enabled System for Indoor Plant Surveillance and Irrigation"

Onose Alexandra

Abstract

This paper presents SmartGarden, an economical and adaptable IoT-based technology intended for real-time monitoring and automated irrigation of indoor plants. The system integrates open-source microcontrollers, ambient sensors, and a locally managed Flutter-based mobile application. In contrast to current commercial solutions that frequently depend on cloud infrastructure and proprietary ecosystems, SmartGarden prioritizes user autonomy, offline functionality, and straightforward customization. The system's novelty is in its open, scalable architecture, enabling both novice and experienced users to customize and enhance it based on particular plant care needs. The prototype was evaluated in real-world situations and shown dependable performance, affirming its potential for practical and instructional applications.

1 Introduction

Houseplants form an important organic element in indoor environments, serving its aesthetic purpose while playing an important role in improving well-being and mental comfort. They provide many advantages: energy, stability, and serenity in life. However, while many people may keep one or more of these plants in their homes, caring for them indoors can prove more challenging than expected. Most problems like under-watering, over-watering, or incorrect location come from the lack of timely and plant-specific information. According to a survey from the National Gardening Association conducted in 2020, nearly 50 percent of first-time plant owners kill at least one plant in their first year because they did not understand their needs or how to care for them.

To overcome these issues, we introduced SmartGarden: an open-source, low-cost platform that does not require horticultural knowledge and delivers moisture and nutrients without the need for subscription services. The main goal is to help the novice and the experts in the measuring and caring process of their plants—by using manual irrigation, automated irrigation, and real-time monitoring of environmental data. The system is scalable, from small potted plants to larger indoor gardens, and the adjustable settings supply it with reliability and flexibility.

SmartGarden benefits from its open and modular nature, which distinguishes it from similar solutions. It is agnostic to cloud services or proprietary ecosystems and relies entirely on on-premise hardware and software. The entire system consists of a set of wireless modules that communicate directly with the Wi-Fi through a lightweight mobile application and can monitor each plant. Users of this app can monitor temperature, humidity, and soil moisture levels; set alerts; and configure watering options. Its hardware is commonly available, cheap, and easy to assemble, which makes the system approachable for students, hobbyists, beginners, and individuals seeking to broaden their horticultural experience and their horticultural scope.

At this point, commercial smart plant care systems are widely available but are limited to single plants, or their applications are limited to polluting agricultural technologies like hydroponics. In contrast, the SmartGarden system is a middle-ground approach, a plug-and-play, reusable offering that holds the advantages of existing tools without sacrificing autonomy, cost, and education/learning. It effectively bridges the gap between entry-level consumer devices and complex industrial systems, at the same time enabling users to hone their personal and intellectual skills with horticulture.

This paper details the development and testing of the SmartGarden system, an offline, low-cost solution for indoor plant monitoring and irrigation. Unlike many commercial alternatives, SmartGarden operates entirely without cloud services, prioritizing data privacy and full user control. The platform is built using accessible, open-source hardware and software, and its modular design allows users to easily expand it—for instance, by adding AI-based diagnostics or new types of environmental sensors. These aspects together define the project's core value: a flexible, decentralized, and educational tool that combines affordability with real-world applicability and technical relevance.

The paper is organized as follows: In Section 2, the hardware architecture and circuit design are described. In Section 3, the software components (firmware and mobile application) are introduced; In Section 4, the experimental validation and results are shown. Finally, in Section 5, findings and future development directions are summarized.

2 Related Work

Smart irrigation and plant monitoring systems have been widely explored in both academic and maker communities, leveraging IoT technologies for real-time data acquisition, automation, and remote control.

Wu et al. [1] designed an intelligent flowerpot that integrates soil moisture sensing and automatic watering. While user-friendly, their solution is tied to a cloud-based infrastructure and supports a single-plant setup, limiting its scalability and privacy. Zhang et al. [3] proposed a more complex system for precision agriculture using wireless sensor networks, suitable for large-scale outdoor farming rather than household applications.

Ahmed et al. [2] presented a multi-sensor smart plant monitoring system using Arduino and Wi-Fi modules. Their setup demonstrates strong environmental monitoring capabilities but lacks modular expansion features. In contrast, SmartGarden supports flexible scaling and localized control for each plant.

Piyare [4] explored ubiquitous home control using smartphones and Android-based IoT integration, illustrating the early potential of mobile interfaces in home automation. Our project builds on similar principles but emphasizes offline-first functionality and platform independence.

Choudhari et al. [5] implemented an IoT-based smart gardening system that collects environmental data and controls irrigation. However, their solution lacks detailed modular configuration per plant and focuses more on general automation. Similarly, Zhang et al. [6] introduced a smart irrigation system using LoRa and edge computing for farms. While innovative, it introduces unnecessary complexity for small indoor setups.

Agarwal et al. [7] advanced the field by incorporating large language models into mobile apps for human-plant interaction, demonstrating potential in diagnostics and interactivity. Risheh et al. [8] proposed using transfer learning with neural networks for smart irrigation—a direction relevant for future iterations of SmartGarden, especially in AI-enhanced diagnostics, as discussed in section 2.4.5.

Foundational insights into IoT development are presented by Bahga and Madisetti [9], offering a practical approach to system design that influenced the modularity of SmartGarden. Margolis [10] and Barnett et al. [13] provide in-depth coverage of Arduino programming, contributing to the firmware development logic in our system.

Broader perspectives on IoT are outlined in Giusto et al. [11] and Xu et al. [12], who highlight the importance of decentralization and security in industrial systems. Our project embraces these principles by prioritizing local data ownership and avoiding cloud reliance. Design best practices from McEwen and Cassimally [14] also informed the physical and software integration aspects of SmartGarden, emphasizing maintainability and modularity.

In conclusion, existing solutions often rely on cloud connectivity [1][2][5], complex hardware [3][6], or high computational resources [7][8]. In contrast, SmartGarden bridges the gap between basic maker projects and complex industrial systems, offering a scalable, user-friendly, and fully offline alternative tailored for educational and indoor use.

3 System Design

The SmartGarden system is built around a handful of simple, affordable components that are easy to find and even easier to work with. Everything is designed to do just what is needed: track plant conditions and control watering—without adding unnecessary complexity. The setup is low-power, easy to tweak, and can be expanded at any time. Whether monitoring a single plant or an entire indoor collection, the system makes it easy to connect multiple modules over the same Wi-Fi network and scale up as you go.

3.1 Hardware Components

At the heart of the SmartGarden system is a simple collection of low-cost, easy-to-find electronics, carefully chosen to manage the basic tasks of monitoring and watering indoor plants. The design is meant to be as straightforward and energy-efficient as possible while staying flexible enough to grow with the user—whether they are caring for a single plant or expanding to a full indoor garden connected over Wi-Fi.

The main "brain" of the system is the Arduino Pro Mini, running at 3.3V. This compact microcontroller coordinates all local operations, from reading sensors to deciding when the plant needs water. It connects directly to two key environmental sensors: one for air and one for soil. The first is the SHT21, mounted on a GY-21 breakout board, which tracks temperature and humidity using I^2C communication. The second is a resistive soil moisture sensor, which sends analog signals that the Arduino translates into soil moisture levels—essential for deciding when to water.

To get data from the hardware to the user (and vice versa), the system uses the ESP-01 module, powered by the ESP8266 chip. It creates a local Wi-Fi connection that lets the SmartGarden system talk to the mobile app in real time. Whether it is sending sensor readings or receiving a command to turn on the water pump, this tiny Wi-Fi module keeps everything coordinated.

Since some parts of the system need more power than others, especially the pump, the design includes a voltage step-up using an MT3608 DC-DC converter. This allows a regular USB power source or even a battery pack to power the entire system reliably, raising lower voltages to the 5V needed by more demanding components.

The pump itself is a compact unit that runs on 5 to 12 volts, more than enough to keep houseplants hydrated. It is controlled by an IRF520 MOSFET relay module, which safely switches the pump on and off based on commands from the Arduino. This keeps the watering process smooth and completely automated—or manual if the user prefers.

For quick testing or temporary setups, everything can be wired on a standard breadboard. But for something more permanent or polished, the system can be soldered onto a custom PCB. Both options work well, depending on how much flexibility or durability is needed. Diagrams of the physical layout and wiring can be seen in Fig. 1 and Fig. 2.

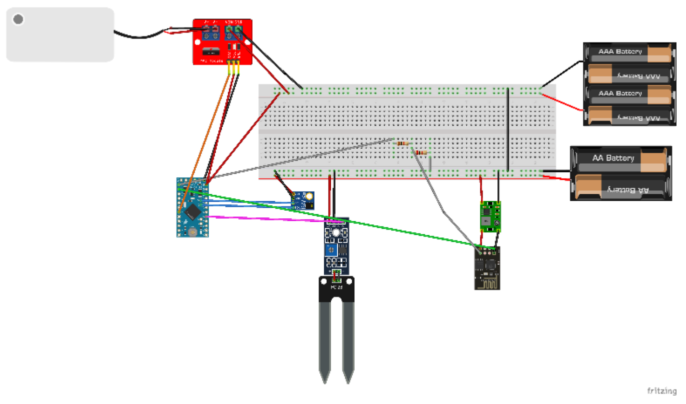

Figure 1. Physical wire configuration of the Smart Garden prototype.

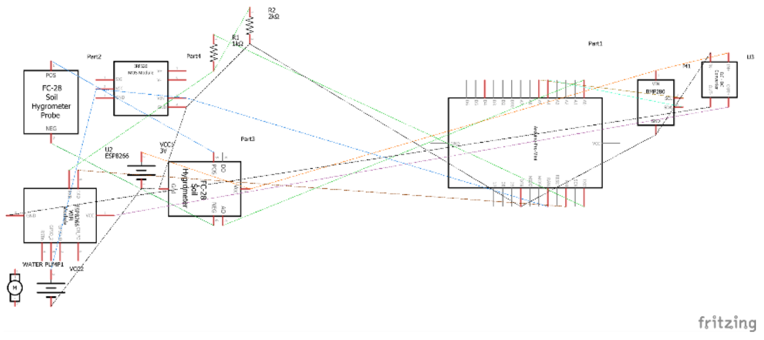

Figure 2. Diagram illustrating the electrical connections of all components.

3.2 Circuit Design

The system's wiring is uncomplicated and follows a straightforward logical architecture, as shown in Figure 3. The Arduino Pro Mini microcontroller serves as the central unit, receiving analog and digital inputs from a soil moisture sensor, a temperature sensor, and a humidity sensor. It processes the collected data and decides whether to activate the water pump through a relay module, depending on the plant's needs. The relay acts as a switch between the pump and the power source, allowing safe control via the

Arduino. Water is delivered from a tank directly to the plant through this mechanism. Communication with a mobile application is established through the ESP-01 wireless module, which provides connectivity via Wi-Fi. This allows users to monitor real-time sensor data and control the irrigation process remotely. The overall circuit design ensures energy efficiency, modularity, and compatibility with low-cost components suitable for home automation applications.

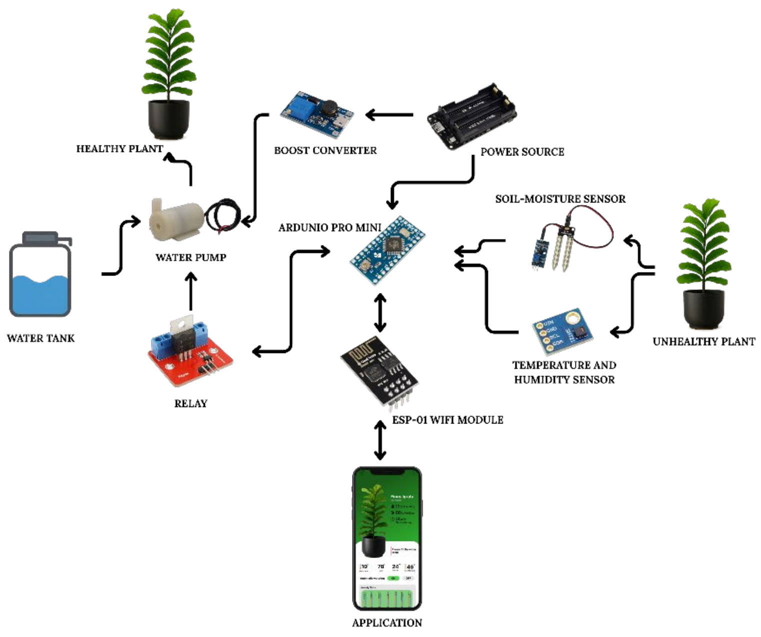

Figure 3. Logical architecture of the Smart Garden system.

3.3 Supported Configuration

The SmartGarden system is designed to manage multiple plant modules at once, with each one equipped with its own set of sensors—measuring soil moisture, air temperature, humidity, and light—along with a resolute watering controller. All these modules connect to the same Wi-Fi network, so everything communicates through a single mobile app using the ESP-01 module as the central hub.

Each plant gets its own unique ID, which the app uses to keep track of them individually. That way, you can monitor and manage several plants at the same time, even if they have different care needs. Whether you are looking after a succulent that barely needs water or a tropical plant that thrives in humidity, each one can run on his own schedule.

The system supports both manual and automatic watering modes, and it collects real-time sensory data from each plant so you can always see how things are going. If you want to expand, you can simply add more modules without changing anything in the core system—if they are on the same network, they will integrate smoothly. There are also plans to support grouped irrigation setups in the future, where several plants can share the same water source while still being watered differently.

Thanks to this flexibility, SmartGarden works just as well for someone caring for a few houseplants as it does for more ambitious indoor gardening projects.

3.4 Enhancements for Accuracy and Reliability

In addition to core system functionality, several design decisions were made to improve the accuracy, reliability, and long-term usability of the SmartGarden platform. This section highlights essential considerations such as sensor calibration and placement, which directly impact system performance.

2.4.1 Supplementary Design Considerations

The sensors in SmartGarden work right out of the box, but if you want your plants to benefit from that data overall, a little calibration goes a long way. Viewing raw sensor values is insufficient without proper calibration to ensure their interpretability and practical relevance.

Calibration begins with the soil moisture sensor, which outputs raw analog values ranging from 0 to 1023. These values represent voltage levels and do not inherently convey meaningful soil hydration levels unless the system is calibrated against known dry and saturated conditions. A simple calibration procedure involves inserting the sensor into completely dry soil and then again into fully soaked soil and taking note of those two values. From there, the system can map that range onto something much more intuitive—like 0% to 100% moisture. Following calibration, a displayed value such as 33% corresponds to a quantifiable level of soil dryness, offering users a more interpretable and actionable measurement rather than an arbitrary number.

The air sensor (SHT21, via the GY-21 module) is already solid. It gives temperature and humidity readings that are generally accurate enough for plant care, with typical variances of about ±0.3°C and ±2% RH. But if you really want to dial it in, especially if you are setting up multiple modules, you can compare it to a reliable thermometer or hygrometer. If the values are a bit off, it is easy to apply a correction in the software so everything stays consistent.

Ultimately, calibration ensures reliability and consistency in system responses, which is essential for automated decision-making. If the system says your soil is dry, it should *be* dry. If you are automating watering, you want it to kick in at the right time—not too early, not too late. Taking a few minutes to calibrate your sensors makes the entire system smarter, more dependable, and better equipped to help your plants thrive, especially if you are working with distinct species, soil types, or changing conditions throughout the year.

2.4.2 Justification for Component Selection

The components selected for the SmartGarden system were determined by principles of modularity, energy efficiency, cost-effectiveness, and interoperability. Each module was chosen for its technical compatibility and its accessibility in educational or maker-oriented contexts, where rapid prototyping and adaptation are crucial.

The wireless communication module is centered around the ESP-01, which utilizes the ESP8266 chipset. This compact and economical module features integrated Wi-Fi capabilities, facilitating effortless integration with local networks for real-time data transmission. Although it has a restricted number of GPIO pins, the ESP-01 is adept at transmitting basic sensor data and receiving control commands within small-scale IoT environments.

Environmental monitoring is chiefly performed using the SHT21 sensor, integrated through the GY-21 module. This sensor offers superior accuracy in temperature and humidity measurements compared to basic alternatives like DHT11 or DHT22. Its I^2C interface guarantees dependable data transmission and facilitates integration in multi-sensor configurations with reduced wiring complexity.

Soil moisture is assessed with an analog resistive sensor, which provides a cost-effective method for detecting hydration levels. When properly calibrated, the sensor delivers adequate precision to guide irrigation decisions and alert thresholds. While more sophisticated capacitive or digital options are available, the selected analog sensor strikes a balance between simplicity, accessibility, and satisfactory performance for general houseplant maintenance.

The IRF520 MOSFET relay module facilitates actuation, allowing for the secure and efficient management of high-power components such as the water pump via logic-level signals from the Arduino. This relay setup ensures electrical isolation between control and power circuits, thereby minimizing the potential for damage caused by voltage fluctuations.

The power infrastructure is augmented by the MT3608 DC-DC boost converter, which elevates the input voltage from low-capacity sources, such as USB ports or battery packs, to satisfy the demands of downstream peripherals, including pumps and relays. Its adjustable output and extensive input range render it exceptionally versatile for variable power conditions.

The irrigation system utilizes a 5–12V submersible pump, chosen for its compact design, silent operation, and compatibility with USB and battery systems. The pump's performance characteristics are well-suited for standard indoor plant arrangements, delivering dependable and adequate flow for periodic watering tasks.

Collectively, these elements provide a system that is both functionally resilient and easily repeatable by students, researchers, and enthusiasts, facilitating instructional applications and small-scale implementations.

2.4.3 Safety and Reliability

In the SmartGarden system, electronic components are close to water sources, so special attention must be paid to safety and long-term reliability. Moisture and electronics do not go together. Different risk factors can develop, which can be prevented through certain design strategies.

One of the key principles is to keep components physically isolated. Sensitive electronics (e.g., microcontrollers and sensors) should be located above the soil level or contained inside protective housings, limiting their exposure to splashes, humid air, and potential corrodibility. When they are properly sealed, these enclosures can enhance the safety and the mechanical integrity and life of the system.

Another paramount safety element relates to electrical isolation of the actuation circuitry. Because we are using an IRF520 relay module to control the water pump, we need a flyback diode across the pump terminals. This part reduces high-voltage surges, which occur from the inductive load of the pump when switching on and off, protecting the microcontroller and other low-voltage logic devices.

Choosing the right power supply is crucial in ensuring reliability in the system. It is recommended to use a stable 5 V power supply, with a minimal current of 1 ampere, to ensure that both control and actuation subsystems are always on. In the case of larger configurations, or for those with certain pumps/peripherals used in parallel, it could be advisable to implement a dedicated power rail for the irrigation hardware to prevent voltage dropping or instability that would affect the logic controller.

These design philosophies allow the SmartGarden system to function properly under moist environments, ensuring reliability while minimizing hardware failures.

2.4.4 Physical Containment

The first version of the SmartGarden system was built on a breadboard—quick to assemble, great for testing, and perfect for early development. But for long-term use, especially around water, a more durable and polished enclosure is strongly recommended. Having proper physical housing is not about making the system look better; it is about protecting the electronics from moisture, dust, and daily wear.

One practical solution is to use a custom 3D-printed enclosure. These can be designed to neatly fit all the core components, such as the ESP-01 Wi-Fi module, the Arduino Pro Mini, the SHT21 sensor, and the wiring that connects everything. A compact, well-fitted enclosure does not just save space; it also makes the entire system easier to manage and more visually integrated into a home or indoor garden setup.

To prevent problems caused by moisture, it is important to seal off any openings where water could sneak. Simple additions like silicone gaskets or rubber grommets around cable entry points can go a long way in keeping things safe. These kinds of water-resistant features help ensure that the system stays dependable, even in high-humidity environments or when the irrigation pump is running regularly. With appropriate enclosures and minor adjustments, SmartGarden transitions from a functional prototype to a robust, user-ready product.

2.4.5 Prospective Expansion

The SmartGarden system was designed from the beginning as a modular and extensible platform, capable of evolving with user requirements and technological progress. Future iterations of the system are anticipated to include several enhancements, focusing on increasing its autonomy and the accuracy of its environmental diagnostics.

An enhancement entails the incorporation of a water-level sensor in the irrigation reservoir. This feature would notify users of low water levels, thus averting dry-run incidents that may harm the pump or jeopardize plant hydration. Additionally, light monitoring could be implemented via a photoresistor (LDR), enabling the system to evaluate light exposure at the plant's site. Based on these measurements, the mobile application could provide recommendations for relocating the plant or activating supplementary lighting, thereby enhancing photosynthetic efficiency.

To enhance sophisticated plant care methodologies, especially in hydroponic systems or for species with particular nutritional needs, the integration of pH and electrical conductivity (EC) sensors is recommended. These sensors would allow for real-time monitoring of substrate chemistry, enabling prompt interventions such as nutrient modifications or water replenishment.

Moreover, a promising avenue for growth resides in the realm of computer vision and artificial intelligence. The mobile application could integrate a feature encouraging users to periodically capture images of their plants. These photographs could be assessed using lightweight AI models to identify visual signs of plant distress or disease, such as chlorosis, wilting, or fungal manifestations. This functionality would enable the system to provide proactive care recommendations and early alerts prior to the emergence of critical problems.

These enhancements would substantially improve the diagnostic capabilities and autonomy of the SmartGarden system, establishing it as a holistic solution for indoor plant management.

4 Software and Communication

The SmartGarden system combines embedded microcontroller logic with a cross-platform mobile application for real-time plant monitoring and intelligent irrigation management. It utilizes lightweight protocols for communication over a Wi-Fi network, ensuring efficiency, scalability, and adaptability for various plant modules.

4.1 Microcontroller Logic

The SmartGarden system's embedded firmware is allocated between two microcontroller platforms: the Arduino Pro Mini (3.3V) and the ESP-01 Wi-Fi module. These components collaboratively manage sensor data acquisition, local server interaction, and irrigation regulation. The Arduino Pro Mini interfaces with environmental sensors, notably the soil moisture probe (through analog input) and the SHT21 module (utilizing the I²C protocol). Sensor readings are obtained at specified intervals and processed locally to maintain system responsiveness and energy efficiency.

The Arduino transmits data via UART to the ESP-01 module, serving as a communication conduit between the sensor subsystem and the mobile application. The ESP-01 operates a lightweight HTTP server, functioning within a local area network independent of cloud infrastructure. This server responds to incoming requests by providing real-time environmental data in JSON format and executing irrigation commands received through the mobile interface.

The ESP-01 firmware incorporates automatic reconnection protocols to identify and rectify Wi-Fi disconnection incidents, thereby preserving system reliability. Furthermore, timing mechanisms are employed to stagger sensor readings and network transmissions, effectively reducing electromagnetic interference and congestion within the local network. Additionally, error-handling protocols are integrated to manage failed transmissions or incomplete data exchanges, ensuring consistent functionality during extended deployment periods.

The communication protocol between the Arduino and ESP-01 is succinct and efficient, emphasizing simplicity and minimal latency. Commands are conveyed as plain text messages, while sensor data is structured in JSON objects. This design approach preserves compatibility with lightweight applications and guarantees system functionality in limited networking conditions.

Segment of Arduino Pro Mini code:

```
#include <Wire.h>
#include <SHT2x.h>
SHT21 sht;
#define SOIL_MOISTURE_PIN A0
#define PUMP_PIN 8
bool pumpState = false; // Controlled by ESP
void setup() {
  Serial.begin(9600);    // Communication with ESP-01
  sht.begin();           // Start SHT21
  pinMode(PUMP_PIN, OUTPUT);
  digitalWrite(PUMP_PIN, LOW);
}
void loop() {
  //  Read sensors
  int soilValue = analogRead(SOIL_MOISTURE_PIN);
  float temperature = sht.getTemperature();
  float humidity = sht.getHumidity();
  // Send sensor data to ESP-01 in JSON format
  Serial.print("{\"temperature\":");
  Serial.print(temperature);
  Serial.print(",\"humidity\":");
  Serial.print(humidity);
  Serial.print(",\"soil\":");
  Serial.print(soilValue);
  Serial.println("}");
  // Check if ESP-01 sent a command
  if (Serial.available()) {
    String cmd = Serial.readStringUntil('\n');
    cmd.trim();
    if (cmd == "WATER ON") {
      pumpState = true;
    } else if (cmd == "WATER OFF") {
      pumpState = false;
  }
  }
  // Set pump based on received command
  digitalWrite(PUMP_PIN, pumpState ? HIGH : LOW);
  delay(5000); // Delay between readings (adjustable)
}
```

Segment of ESP-01 code:

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
const char* ssid = "Alexandra";
const char* password = "26012005";
ESP8266WebServer server(80);
String lastSensorData = "{}";
void setup() {
Serial.begin(9600); // Communicate with Arduino
WiFi.begin(ssid, password);
Serial.print("Connecting to WiFi");
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("\nWiFi connected!");
Serial.print("ESP IP: ");
Serial.println(WiFi.localIP());
// Endpoint: GET /data
server.on("/data", HTTP_GET, []() {
server.send(200, "application/json", lastSensorData);
});
// Endpoint: POST /water
server.on("/water", HTTP_POST, []() {
if (!server.hasArg("plain")) {
server.send(400, "text/plain", "Missing body");
return;
}
String command = server.arg("plain");
command.trim();
if (command == "WATER ON" || command == "WATER OFF") {
Serial.println(command); // Send to Arduino
server.send(200, "text/plain", "Command sent: " + command);
} else {
server.send(400, "text/plain", "Invalid command"); }
});
server.begin();
Serial.println("HTTP server started"); }
void loop() {
```

```
server.handleClient();
// Read sensor data from Arduino
if (Serial.available()) {
lastSensorData = Serial.readStringUntil('\n');
lastSensorData.trim();
Serial.println("[ESP] Received from Arduino: " + lastSensorData); } }
```

4.2 Mobile Application

The SmartGarden mobile application was created utilizing the Flutter framework to guarantee cross-platform compatibility and deliver an accessible interface for monitoring and managing plant modules. The application interacts with each hardware node via a local Wi-Fi network and offers intuitive visual representations of environmental data along with irrigation controls.

Upon initiating the application, users encounter a dashboard that consolidates data from all operational plant modules. Each module is depicted by a tile exhibiting real-time sensor metrics, including soil moisture percentage, ambient temperature in degrees Celsius, and relative humidity in percentage. Color-coded indicators and succinct status labels (e.g., "Needs Watering," "Healthy") facilitate a quick evaluation of plant conditions.

By selecting a designated plant module, users can obtain a comprehensive view featuring individual sensor data and control functionalities. The application accommodates both manual and automatic irrigation modes. In manual mode, users can initiate immediate watering events or regulate the irrigation duration via on-screen commands. In automatic mode, users establish soil moisture thresholds and maximum watering durations, allowing the system to function autonomously based on sensor feedback.

Notifications are generated in real time to inform users when soil moisture levels drop below critical thresholds or when environmental parameters, such as temperature or humidity, deviate from acceptable ranges. Other notification triggers encompass communication delays or data update failures, aiding in the identification of malfunctioning modules or network problems.

Each plant module is distinctly recognized within the application, enabling users to manage multiple plants autonomously and customize configurations according to the requirements of various species. This modularity guarantees that the application is scalable and adaptable to diverse use cases, ranging from individual plant monitoring to multi-node indoor gardening systems.

Flutter's cross-platform capabilities make development much more efficient while ensuring users get the same great experience whether they're on Android or iPhone. This approach also makes it easier to add exciting features down the road—like using your phone's camera to diagnose plant problems or syncing your garden data to the cloud, which would really expand what the SmartGarden system can do.

At the current stage of development, the application’s interface is represented through early-stage mockups (Figures 4 and 5), which illustrate the intended layout and user interactions of the final implementation.

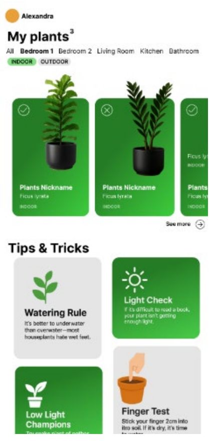
Alexandra
My plants³
All Bedroom 1 Bedroom 2 Living Room Kitchen Bathroom
INDOOR OUTDOOR
Plants Nickname
Ficus lyrata
INDOOR
Plants Nickname
Ficus lyrata
INDOOR
See more
Tips & Tricks
Watering Rule
It's better to underwater than overwater—most houseplants hate wet feet.
Light Check
If it's difficult to read a book, your plant isn't getting enough light.
Low Light Champions
Finger Test
Stick your finger 2cm into the soil. If it's dry, it's time

Figure 4. Early-stage UI mockup of the SmartGarden dashboard interface (designed in Figma).

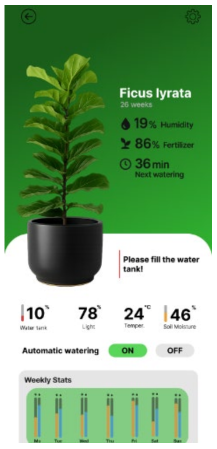

Figure 5. Early-stage UI mockup of the individual Plant Details interface(designed in Figma).

4.3 Data Communication

The communication architecture of the SmartGarden system consists of a light and decentralized model, based on a local HTTP server hosted on the ESP-01 module. This approach eliminates the need to rely on external cloud/hosting infrastructure, message broker, or third-party services, which enhances data privacy and gives the ability for the user to control the entire functionality of the system

ESP-01 module works on RESTful server, offering a small number of endpoints for data transmission and commands execution in real-time. The mobile application can retrieve sensor data from the GET /data endpoint, which returns the latest environmental values in a JSON-encoded format. Mobile interface constantly polls this endpoint so that it reflects the plant conditions accurately.

In addition to passive monitoring, users can access the POST /water endpoint to trigger control actions. This interface fetches system commands such as "WATER ON" or "WATER OFF," allowing the user to turn the water pump on or off from the application interface. We provide a highly efficient communication protocol, which maximizes raw bandwidth usage while minimizing negative content latency.

Our SmartGarden system is designed to run over the local Wi-Fi in your house, so you do not need to connect it to the internet. This provides the ability to reduce operational costs while enhancing the platform's robustness and safety, making it suitable for environments where cloud accessibility is not possible or ideal.

5 Experimental Configuration and Outcomes

For the practical assessment of the SmartGarden system, a complete prototype was built, and it was evaluated for several days with a single indoor plant: a Monstera Deliciosa. The selected species was a well-known houseplant with a moderate sensitivity to watering conditions to assess overall accuracy and responsiveness of the system.

The main objective of the assessment was to ensure the operability of the sensor modules, the fault-free operation of the irrigation control logic, and the functionality of real-time data exchange between the hardware and the mobile application in a real-life condition inside a room.

5.1 Setup Description

The experimental assessment utilized a solitary indoor plant specimen, Monstera Deliciosa, contained in a conventional flowerpot. This species was chosen for its prevalence among indoor horticulturists and its moderate sensitivity to watering frequency, rendering it an appropriate subject for evaluating the responsiveness and dependability of automated irrigation systems.

The SmartGarden system was comprehensively implemented in this configuration, with all components interconnected via a solderless breadboard to enable swift prototyping. Power was provided through a 5V USB adapter, regulated by an MT3608 step-up converter to satisfy the operational voltage demands of modules. The hardware interfaced with the mobile application through a local Wi-Fi network, facilitating real-time data collection and remote irrigation management.

Sensor instrumentation comprised a soil moisture sensor positioned adjacent to the plant's root zone to accurately measure hydration levels with significant spatial relevance. The SHT21 sensor module (GY-21) was affixed above the pot to continuously monitor ambient temperature and humidity, reflecting the microclimatic conditions directly influencing the plant. The ESP-01 module served as the wireless communication interface, establishing and sustaining a local network connection with the application, thereby ensuring uninterrupted data transmission and command execution.

Irrigation was conducted utilizing a compact submersible water pump, operable either manually through a mobile interface or automatically via sensor feedback. The pump was regulated through an IRF520 MOSFET relay module, which facilitated the requisite switching mechanism to connect the low-voltage control circuit with the pump's power supply. The complete configuration is presented in Figure 6, highlighting the arrangement of the electronic components and their spatial positioning.

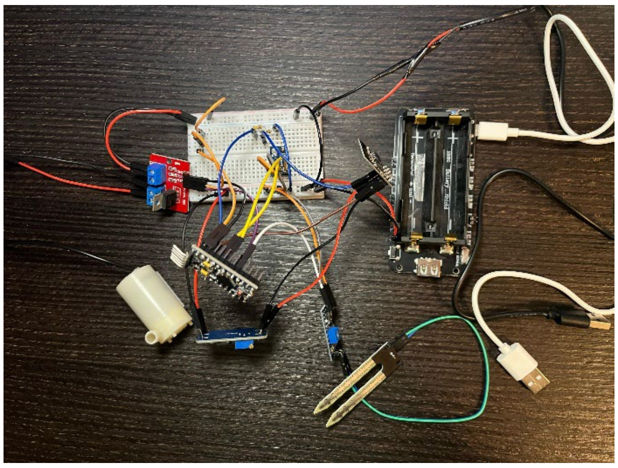

Figure 6. SmartGarden experimental setup with core components assembled on a breadboard for demonstration purposes.

5.2 Observations

During the experimental testing phase, the SmartGarden system demonstrated consistent and reliable performance under standard indoor environmental conditions. A significant observation involved the dynamics of soil moisture. After each irrigation event, moisture levels decreased at a natural rate, which varied according to plant species and soil composition. When sensor readings dropped below a user-defined threshold—established at 30% for the test plant—the system either activated automatic watering or dispatched an alert to the mobile application, contingent upon the chosen operational mode.

The SHT21 module exhibited consistent and precise readings throughout the testing period regarding environmental sensing. Variations in ambient temperature and relative humidity logically aligned with diurnal cycles and room ventilation occurrences, confirming the sensor's responsiveness and stability. These measurements yielded a dependable dataset for monitoring the plant's growing environment and guided potential future modifications to care routines.

The system's network resilience was equally significant. The ESP-01 module sustained uninterrupted Wi-Fi connectivity, with no critical failures or communication disruptions noted. The embedded firmware's automatic reconnection protocols functioned as intended during simulated disconnection scenarios, guaranteeing data continuity and maintaining user control through the mobile application.

The SmartGarden prototype demonstrated its effectiveness as a low-maintenance and responsive instrument for indoor plant care. Its capacity to combine sensor-driven automation with manual override options facilitated a flexible plant management approach. Users could oversee and irrigate multiple plants via a centralized, intuitive interface without dependence on external cloud services, validating the proposed design's feasibility for both hobbyist and research-oriented applications.

6 Conclusions

We presented SmartGarden, an open-source, modular, and affordable IoT platform designed for the monitoring and irrigation of indoor plants. By integrating low-cost microcontrollers (Arduino Pro Mini and ESP-01), high-precision environmental sensors (SHT21), and a cross-platform mobile application built using Flutter, this system provides a robust and cost-effective alternative to current smart gardening technologies.

Unlike many commercial systems that rely on proprietary cloud services, SmartGarden operates entirely offline and is fully decentralized. This promotes data privacy, long-term maintainability, and complete user control. Compared to similar solutions, such as the IoT-enabled flowerpot proposed by Wu et al. [1] or the precision irrigation controller evaluated by Zhang et al. [3], SmartGarden prioritizes simplicity, full user ownership, and hardware flexibility. Its modular architecture allows users to scale the platform as needed, accommodating a wide variety of plants and enabling deployment in both personal and educational contexts.

The project was successfully implemented and tested in real-world conditions. The system proved to be reliable, flexible, and easy to replicate, supporting both manual and automated irrigation for various plant species. Its offline functionality makes it particularly suitable for environments where internet connectivity is limited or unavailable, such as rural homes or classrooms.

SmartGarden also serves as an accessible platform for students and enthusiasts to explore embedded systems, sensor networks, wireless communication, and environmental monitoring. Its open-source nature encourages experimentation, customization, and further development.

Future enhancements may include the integration of additional sensors (e.g., pH, EC, light intensity via LDR), computer vision modules for plant health diagnostics, and optional cloud connectivity for extended functionality. These additions would significantly increase the autonomy, diagnostic precision, and usability of the system, transforming SmartGarden into a comprehensive smart horticulture platform.

The novelty of SmartGarden lies not in the individual components used, but in the practical integration of offline-first functionality, modular hardware design, and open educational accessibility.

While similar projects exist, few combine these attributes into a cohesive platform that can serve as a low-barrier entry point for IoT experimentation, especially in rural, academic, or budget-constrained environments. As such, this system contributes to bridging the gap between DIY prototyping and scalable smart infrastructure in the context of indoor horticulture.

Acknowledgement: This work was supervised by Professor *Anca Vasilescu*, from *Transilvania University of Brasov.*

References

[1] Y. Wu, C. Zhai, Y. Tian, *Design of Intelligent Flowerpot Based on Internet of Things*, Asian Journal of Applied Science and Technology, vol. 4, no. 3, pp. 1–6, 2020.

[2] M. Ahmed, A. Hannan, A. Basaruddin, M. F. Zolkipli, *Development of Smart Plant Monitoring System using IoT*, Indonesian Journal of Electrical Engineering and Computer Science, vol. 19, no. 3, pp. 1490–1498, 2020.

[3] T. Zhang, W. Zhang, J. Wang, *Wireless Sensor Network-Based Intelligent Irrigation Control System for Precision Agriculture*, Agricultural Engineering International: CIGR Journal, vol. 21, no. 1, pp. 97–107, 2019.

[4] R. Piyare, *Internet of Things: Ubiquitous Home Control and Monitoring System using Android based Smart Phone*, International Journal of Internet of Things, vol. 2, no. 1, pp. 5–11, 2013.

[5] G. R. Choudhari, P. A. Dagale, I. S. Dashetwar, R. R. Desai, A. A. Marathe, *IoT-based Smart Gardening System*, Journal of Physics: Conference Series, vol. 2601, no. 1, p. 012006, 2023.

[6] Y. Zhang, X. Wang, L. Jin, J. Ni, Y. Zhu, W. Cao, X. Jiang, *Research and Development of an IoT Smart Irrigation System for Farmland Based on LoRa and Edge Computing*, Agronomy, vol. 15, no. 2, p. 366, 2025.

[7] K. Agarwal, S. Ananthanarayanan, S. Srinivasan, A. S, *Enhancing IoT-based Plant Health Monitoring through Advanced Human-Plant Interaction using Large Language Models and Mobile Applications*, *arXiv preprint*, arXiv:2409.15910, 2024.

[8] A. Risheh, A. Jalili, E. Nazerfard, *Smart Irrigation IoT Solution using Transfer Learning for Neural Networks*, *arXiv preprint*, arXiv:2009.12747, 2020.

[9] A. Bahga, V. Madisetti, *Internet of Things: A Hands-On Approach*, Universities Press, 2014.

[10] M. Margolis, *Arduino Cookbook*, 2nd ed., O'Reilly Media, 2011.

[11] D. Giusto, A. Iera, G. Morabito, L. Atzori (Eds.), *The Internet of Things: 20th Tyrrhenian Workshop on Digital Communications*, Springer, 2010.

[12] L. D. Xu, W. He, S. Li, *Internet of Things in Industries: A Survey*, IEEE Transactions on Industrial Informatics, vol. 10, no. 4, pp. 2233–2243, 2014.

[13] R. H. Barnett, L. O. Mazurek, S. C. Cox, *Embedded C Programming and the Atmel AVR*, Delmar Cengage Learning, 2008.

[14] A. McEwen, H. Cassimally, *Designing the Internet of Things*, Wiley, 2013.

Onose Alexandra
Transilvania University of Brasov
Faculty of Mathematics and Informatics
Bd. Iuliu Maniu nr. 50
500091 Brașov Romania

E-mail:
alexandra.onose@student.unitbv.ro