

IoT-Based Automatic Monitoring and Watering System for California Lemon Tree Seedlings with Web Dashboard and Telegram Bot Integration

Muhamad Aldo Ryanda^{1*}, Ahmad Rofiq Hakim², Fara Triadi³

^{1,2,3}Computer Engineering Technology Study Program, Department of Information Technology, State Polytechnic of Samarinda
aldoryanda12@gmail.com^{1*}, rofiq_93@yahoo.com², fara_triadi@polnes.ac.id³

Abstract

Manual watering of California lemon (*Citrus limon* L.) seedlings often leads to inconsistent soil moisture, which can cause root rot from overwatering or plant stress from drought [1]. This study presents the design and implementation of an Internet of Things (IoT)-based automatic watering system integrated with a web dashboard and Telegram bot. The system employs an ESP32 microcontroller connected to a soil moisture sensor (YL-69) for detecting soil water content and an ultrasonic sensor (HC-SR04) for monitoring water tank capacity. Sensor data is transmitted to Firebase Realtime Database and displayed on a web dashboard, while a Telegram bot named "AutoLemon" provides remote notifications and manual control capabilities [2]. The development followed a Prototype methodology encompassing communication, rapid design, prototype construction, and user feedback. Field testing demonstrated that the system operates as designed: the water pump activates automatically when soil moisture drops below 40% and deactivates once moisture reaches the 50–75% range. Compared to manual watering, which consumed an average of 600 ml per day, the automated system reduced water usage to approximately 479 ml per day—a savings of roughly 20%. The integration of dual monitoring interfaces enables real-time oversight of both soil conditions and water availability, offering a practical solution for small-scale lemon cultivation.

Keywords: Internet of Things, ESP32, soil moisture sensor, ultrasonic sensor, Firebase, web dashboard, Telegram bot, California lemon, automatic watering system

1. Introduction

California lemon (*Citrus limon* L.) is a high-value horticultural crop distinguished from local lemon varieties by its higher fruit yield, greater water content, and richer vitamin composition [3]. The fruit contains flavonoids, limonene, polyphenols, vitamin C, and essential oils that exhibit antibacterial, anti-inflammatory, and antioxidant properties [4]. Its phytochemical profile includes limonene at concentrations of up to 29.52 mg/g in essential oil and flavonoids ranging from 19.5 to 27.5 mg/g in fruit extract [4]. Extraction studies using microwave-assisted methods have confirmed the presence of these bioactive compounds in lemon peel [5], while maturity classification research using reflectance-fluorescence imaging has demonstrated that California lemon can be systematically characterized for quality assessment [6]. For optimal growth, lemon seedlings require well-aerated soil rich in organic matter, adequate sunlight exposure, and a planting distance of approximately 50 cm between individual plants [7]. Soil thermal characteristics also play an important role in tropical regions, where higher thermal conductivity and heat capacity support faster physiological processes including seed germination and nutrient absorption [8]. Furthermore, overly dry soil can affect pH measurement accuracy, as fertilizer salts and minerals become more concentrated, yielding pH readings that appear lower than the true value [9].

In cultivation, maintaining proper soil moisture is critical. Soil that is too dry deprives the plant of water, while excessive moisture risks root rot [1]. In current farming practice, watering is still largely performed manually. This approach creates several problems: excessive water consumption, higher labor demands, and the inability of farmers—particularly elderly ones—to maintain consistent watering schedules [10]. Unmonitored soil moisture can swing between extremes, ultimately damaging plant health. The absence of real-time monitoring further compounds the difficulty, as farmers cannot determine precisely when watering is most needed, especially when they are away from the growing site.

An automatic plant watering system is a device designed to irrigate plants without requiring continuous manual intervention [11]. When implemented with IoT technology, such a system can detect soil moisture in real time through embedded sensors and activate a water pump automatically when moisture falls below a predefined threshold [12]. IoT itself refers to a global network infrastructure that connects

physical and virtual objects through data capture and communication capabilities, enabling machine-to-machine and human-to-machine interaction via the internet [13], [14], [15]. Integration with messaging platforms like Telegram extends this capability to remote monitoring and control, enabling farmers to manage watering from any location via their smartphones [2]. Telegram provides a bot API, data encryption, and a lightweight MTPROTO protocol, making it well-suited for IoT communication applications [16].

This study develops an IoT-based automatic monitoring and watering system for California lemon tree seedlings. The system centers on an ESP32 microcontroller interfaced with a soil moisture sensor (YL-69) and an ultrasonic sensor (HC-SR04) for water tank level detection. Data is pushed to Firebase Realtime Database and rendered on a web dashboard, while a Telegram bot handles notifications and manual pump control. The primary objectives are threefold: (1) to design and build a monitoring device that tracks soil moisture and water capacity in real time, (2) to integrate a web dashboard and Telegram bot as dual monitoring and control interfaces, and (3) to evaluate whether the automated system can reduce water consumption and labor compared to conventional manual watering.

2. Related Work

Several prior studies have explored IoT-based automatic plant watering systems, providing a foundation for the present research. Prasetyo et al. [2] developed a monitoring and control system for potato plant watering using an ESP32 and YL-69 soil moisture sensor, with Telegram integration for notifications and manual control. Their system activated the pump when soil moisture fell below 60–70% and demonstrated that the YL-69 sensor could detect soil moisture accurately across varying conditions. However, the system lacked a web-based dashboard, relying solely on Telegram for remote monitoring.

Akbar [12] implemented an IoT-based soil moisture monitoring system using an Arduino Uno and YL-69 sensor, with data displayed on an LCD and transmitted to a mobile application. The system confirmed a linear relationship between water volume and sensor voltage increase, indicating successful soil condition detection. A key limitation was its dependence on stable power and internet connectivity for real-time data access.

Effendi et al. [10] designed an automatic watering system using a NodeMCU ESP8266 with soil moisture sensor, relay, DC water pump, and Blynk application for monitoring. The system functioned optimally, activating the pump when soil was dry and deactivating it once adequate moisture was reached. However, the accuracy of the YL-69 sensor could degrade when the soil became excessively wet.

Subagja et al. [1] tested an IoT-based automatic watering system using a NodeMCU ESP8266, soil moisture sensor, OLED display, and Firebase for web monitoring. Watering triggered automatically when moisture dropped below 50%, though data transmission to the website experienced delays of up to 133 seconds due to network instability. The system also lacked water tank capacity monitoring.

Haryanto [17] used an Arduino Uno with an FC-28 soil moisture sensor and DC pump controlled via relay, achieving automatic watering when moisture readings fell below a defined threshold. The system also displayed soil moisture data on an LCD for local monitoring. Mardiyansyah [18] developed an automatic irrigation controller using an Arduino Uno and HC-SR04 ultrasonic sensor to regulate a servo-driven irrigation gate based on water level. The system opened the gate 3.5 cm when the water distance was ≥ 10 cm, narrowed it to 2 cm at 7 cm, and closed it entirely at ≤ 5 cm. While functional, the system had no IoT or web integration for remote monitoring.

Gitakarma and Tjahyanti [14] reviewed the role of IoT and artificial intelligence in modern technology, concluding that IoT serves as the backbone of device communication while AI enables intelligent data processing. Their review highlighted that combining IoT with AI accelerates automation and improves prediction accuracy, although challenges in data management, privacy, and infrastructure persist.

Pratama [19] applied an ultrasonic sensor to an automatic trash bin with Telegram and sound alarm integration, using a prototyping method that encompassed literature study, needs analysis, design, and testing. The system successfully detected waste levels and delivered notifications to the Telegram application. Sofiumayroh and Hasfani [20] utilized the HTTP protocol for a water temperature monitoring system using an Arduino WiFi R4 and DS18B20 sensor. Their findings indicated that adding a heater accelerated water heating but increased data transmission delay to the server, with average delays ranging from approximately 1958 ms with one heater to 3767 ms with four heaters.

Agustina and Dewi [21] designed an automatic plant watering tool based on a microcontroller and IoT technology, demonstrating that IoT integration enables real-time remote monitoring of environmental parameters relevant to plant care. Pangestu [22] developed a remote monitoring system for smart agriculture, highlighting the practical value of cloud-based dashboards for agricultural data visualization and decision support. Sugandi [23] investigated temperature, pH, and moisture monitoring for smart farming applications using IoT, confirming that sensor-based environmental monitoring can improve agricultural productivity when combined with cloud-based data infrastructure.

The present study builds on these earlier efforts but introduces several distinguishing features. First, it combines two environmental sensors—soil moisture (YL-69) and ultrasonic (HC-SR04)—within a single ESP32-based system, whereas most prior works employed only one sensor type [1], [10], [12]. Second, the system provides a dual monitoring interface through both a Firebase-based web dashboard and a Telegram bot with notification and manual control capabilities, a combination not found in the reviewed literature. Third, the inclusion of water tank capacity monitoring with low-level buzzer alerts addresses a gap identified in studies such as Akbar [12] and Effendi et al. [10], where systems could fail silently when the water supply was exhausted.

3. Research Methodology

3.1. Research Setting and Object

The research was conducted at a small-scale lemon cultivation site located on Jalan Siliwangi, RT.06, Desa Purwajaya, Kecamatan Loa Janan, Kabupaten Kutai Kartanegara, East Kalimantan, Indonesia. The study focused on developing an automatic watering system for

California lemon tree seedlings with a maximum height of 50 cm, integrated with a web dashboard and Telegram bot for remote monitoring and control.

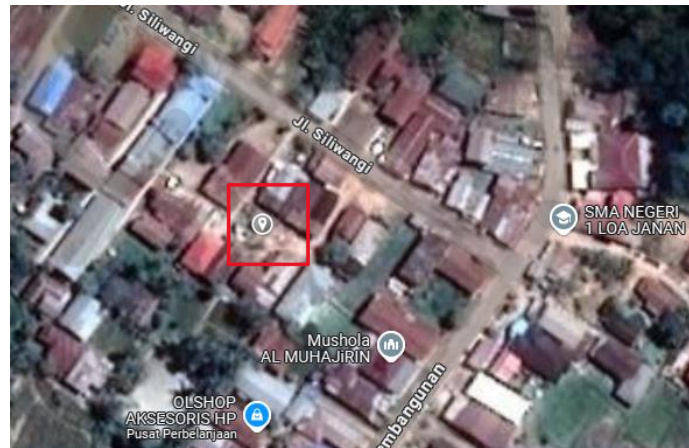


Fig 1: Map of the research location.

3.2. Systems Components

The hardware components used in this study are summarized in Table 1. The ESP32 Wroom32 serves as the main microcontroller, featuring a dual-core Xtensa LX6 processor running at 240 MHz with integrated Wi-Fi 802.11 b/g/n and Bluetooth v4.2, 520 KB of SRAM, and 4 MB of flash memory [24], [25]. The soil moisture sensor (YL-69) is a resistive-type sensor that detects soil water content through changes in electrical conductivity between its probe electrodes [26]. The ultrasonic sensor (HC-SR04) measures distance by emitting sound waves beyond the audible frequency range (20 kHz–2 MHz) and calculating the time required for the reflected signal to return [27]. The 2-channel relay module operates as an electronically controlled electromagnetic switch, using the Normally Open (NO) contact to control the water pump circuit [28]. The LM2596 step-down module reduces the 12V DC input voltage to 5V for the ESP32 and sensor modules, maintaining stable output despite input fluctuations [29].

Table 1: Hardware Components

No.	Component	Specification	Function
1	ESP32 Wroom32	Dual-core Xtensa LX6, 240 MHz; Wi-Fi 802.11 b/g/n; 520 KB SRAM; 4 MB Flash	Main microcontroller
2	Soil Moisture Sensor (YL-69)	Resistive type, analog output	Soil moisture detection
3	Ultrasonic Sensor (HC-SR04)	20 kHz–2 MHz, 2 cm–400 cm range	Water tank level measurement
4	2-Channel Relay Module	5V, Normally Open (NO)	Pump switching
5	LM2596 Step-Down Module	Adjustable output, up to 3A	Voltage reduction from 12V to 5V
6	Submersible Water Pump	DC 12V, 800 L/h flow rate	Water delivery to plants
7	Power Supply	12V DC, minimum 3A	System power source
8	Buzzer	5V	Low water level alarm
9	AWG 22 Cable	20 meters	Component wiring

The software tools employed throughout the system development process are presented in Table 2. Arduino IDE was used as the primary development environment for programming and configuring the ESP32 microcontroller. It is an open-source platform that supports microcontroller programming using the C and C++ programming languages [30]. Through this software, developers can write, verify, compile, and upload source code directly to the ESP32. Arduino IDE also supports various libraries that simplify the implementation of sensor communication, Wi-Fi connectivity, data processing, and cloud integration. Firebase Realtime Database was used as the cloud-based data storage and synchronization platform. This backend-as-a-service enables the ESP32 to transmit measurement data directly to the web dashboard in real time [1]. The database stores information in a structured format and automatically synchronizes data changes across connected devices, allowing sensor readings and system status to be displayed immediately without requiring users to refresh the page manually.

The Hypertext Transfer Protocol, or HTTP, was used to facilitate data communication between the ESP32 and the cloud platform. HTTP operates through a client-server architecture over the TCP/IP protocol, where the ESP32 acts as a client that sends measurement data and requests to the server [20], [31], [35]. Its standardized communication mechanism, broad compatibility, and relatively simple implementation make it suitable for connecting embedded devices with cloud-based applications. Visual Studio Code was used as the main code editor for developing the web-based monitoring dashboard. This open-source editor, developed by Microsoft, supports multiple programming languages and provides various extensions for syntax highlighting, code completion, debugging, project management, and

version control [32], [36]. These features helped developers manage HTML, CSS, JavaScript, and other supporting files efficiently within a single development environment, thereby supporting the creation of a responsive, structured, and user-friendly monitoring dashboard.

Table 2: Software Tools

No	Software	Function
1	Arduino IDE	ESP32 firmware programming
2	Visual Studio Code	Web dashboard development
3	Firebase Realtime Database	Cloud data storage and real-time synchronization
4	GitHub Pages	Static web hosting for dashboard
5	Telegram Bot API	Remote notification and manual control
6	Fritzing	Circuit schematic design

3.3. Development Method

The system was developed using the Prototype methodology [33], [34], which involves iterative cycles of design, construction, and user feedback. This approach enables developers to build an initial model of the system quickly, obtain direct feedback from users, and refine the design before final deployment [33]. The methodology comprised four stages:

1. Communication.

Initial discussions with the end user (a local farmer) were conducted to identify the core problems with manual watering and to gather system requirements. Field observations supplemented these discussions by documenting soil conditions, sunlight exposure, and site layout.

2. Quick Design.

Based on the gathered requirements, a rapid system design was produced that included component selection, wiring layout, circuit schematic, and interface mockups for both the web dashboard and Telegram bot.

3. Prototype Construction.

The hardware was assembled and the firmware was programmed using Arduino IDE. Two timing intervals were implemented: a 5-second loop for pump control logic to ensure rapid response to soil moisture changes, and a 60-second loop for data transmission to Firebase and Telegram to conserve bandwidth and reduce server load.

4. Development, Delivery, and Feedback.

The completed prototype was tested at the cultivation site. Feedback from the end user guided further refinements to the system logic and interface design.

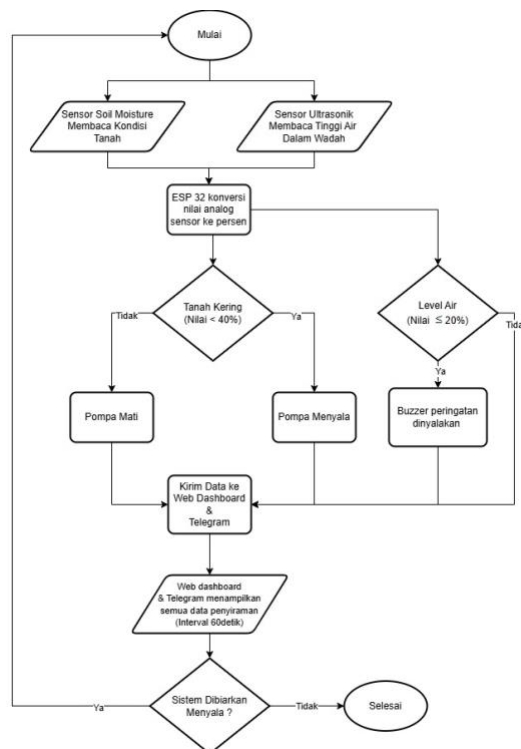


Fig 2: System Flowchart

3.4. System Architecture

The system operates on a client-server model. The ESP32 microcontroller reads data from the YL-69 soil moisture sensor and the HC-SR04 ultrasonic sensor. Based on the soil moisture reading, the microcontroller controls the water pump through the relay module. Simultaneously, sensor data and pump status are transmitted via HTTP to Firebase Realtime Database, where they are rendered on the web dashboard in real time [1], [31]. The Telegram bot receives periodic status updates and processes manual commands from the user.

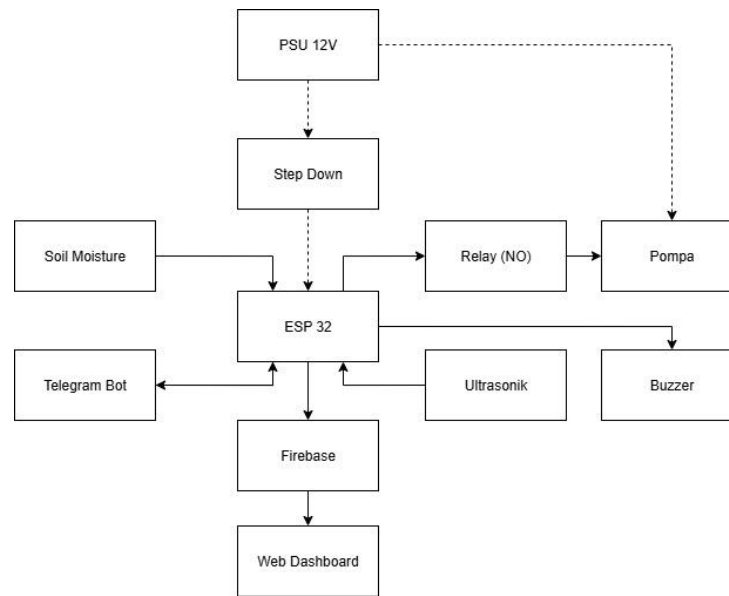


Fig 3: Block Diagram

The soil moisture sensor converts analog readings into a percentage value through calibration. The raw analog value from the sensor is mapped between predefined dry (4095) and wet (970) reference points to produce a moisture percentage. The ultrasonic sensor measures the distance from the sensor to the water surface using the formula:

$$Distance (cm) = \left(\frac{Duration \times Speed of Sound}{2} \right)$$

Table 3: Description of Distance measurement formula

No.	Symbol	Description	Unit
1	d (Distance)	Distance between the ultrasonic sensor and the water surface	cm
2	v (Speed of Sound)	Speed of sound in air (approximately 0.0343cm/μs at 20°C)	cm/μs
3	t (Duration)	Echo pulse duration measured by the sensor	μs

Which is then converted into a water level percentage based on the known tank height (30 cm).

The watering logic operates as follows: when soil moisture falls below 40%, the pump is activated automatically. The pump continues to run until the moisture reading reaches the 50–75% range, at which point it deactivates. If the water tank level drops to 20% or below, the buzzer sounds as an alert, and a notification is sent to the user via Telegram regardless of the pump state. The user can also switch between automatic and manual modes through the Telegram bot, where manual mode allows direct pump control.

4.2. Hardware Implementation

The hardware was assembled on a mounting board with all components wired to the ESP32 microcontroller. A 12V DC power supply provided energy to the system, with the LM2596 step-down module reducing voltage to 5V for the ESP32 and sensor modules. The 2-channel relay module controlled the submersible water pump, using the Normally Open (NO) contact to ensure the pump remained off by default.

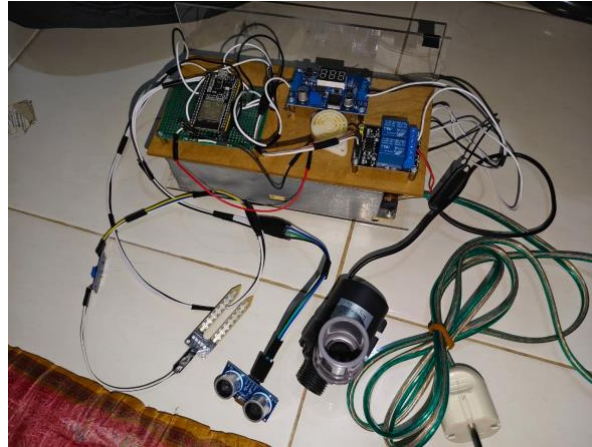


Fig 6: Assembled Hardware

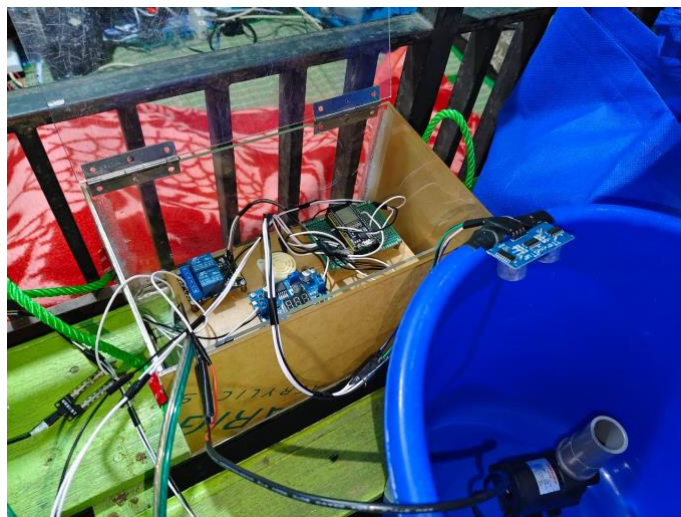


Fig 7: Device with Integrated Water Tank and Casing

The device was enclosed in a protective casing with the water tank integrated at the base. The soil moisture sensor probe was positioned in the plant pot, while the ultrasonic sensor was mounted above the water tank to measure the distance to the water surface.

4.3. Software Implementation

The firmware was programmed in C/C++ using the Arduino IDE [30]. The program begins with library imports for Wi-Fi, Firebase, and Telegram bot connectivity, followed by pin assignments and calibration parameters. During setup, the ESP32 connects to the Wi-Fi network, authenticates with Firebase, and initializes the Telegram bot with an online status message.

The main loop executes two concurrent tasks. The control loop runs every 5 seconds to read the soil moisture sensor, determine the soil condition (dry, moist, wet, or transitional), and control the pump accordingly. The data transmission loop runs every 60 seconds to push the latest sensor readings—including soil moisture percentage, soil status, water level, pump status, and control mode—to Firebase. The same interval triggers a Telegram notification containing a summary of current system status.

Manual control through the Telegram bot allows the user to switch between automatic and manual modes, toggle the pump on or off, and request a status update on demand. The bot processes incoming commands and relays them to the ESP32 for execution [16].

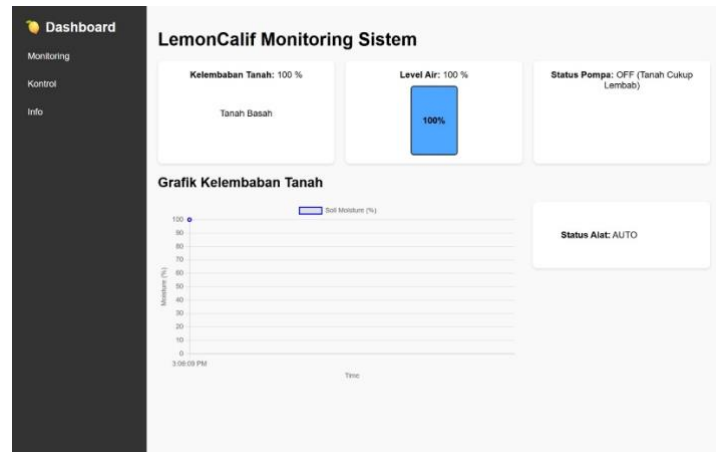


Fig 8: Web Dashboard Interface

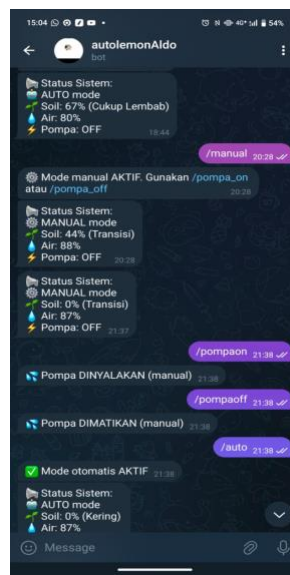


Fig 9: Telegram Bot Interface

4.4. Hardware Testing

Hardware testing verified that each component functioned correctly within the integrated system. Three key scenarios were evaluated.

Scenario 1: Moist soil condition. When the soil moisture sensor detected a reading of 70% (classified as "sufficiently moist"), the pump remained inactive. The ultrasonic sensor confirmed the water level was above 20%, and the buzzer did not sound. This confirmed that the system correctly refrained from watering when conditions were adequate.



Fig 10: System in Moist Condition

Scenario 2: Dry soil condition. When the sensor was lifted from the soil to simulate extreme dryness (reading 0%), the system immediately activated the pump. This demonstrated the system's ability to respond promptly to critical moisture loss.



Fig 11: System Activating Pump under Dry Conditions

Scenario 3: Low water level with dry soil. When the water tank level dropped below 20% while the soil remained dry, the pump did not activate despite the critical moisture level. Instead, the buzzer sounded to alert the user that the water supply was depleted. This safety mechanism prevents pump damage from dry running and ensures the user is informed to refill the tank.



Fig 12: Buzzer Alert when Water Level is Below 20%

4.5. Software Testing

Software testing validated the data communication pipeline from sensor to cloud to user interface. The web dashboard successfully displayed real-time sensor data, including soil moisture percentage, water level, and pump status, with updates refreshing at the configured interval.

The Telegram bot testing confirmed that the system could deliver timely notifications and process manual commands. When the soil moisture transitioned from 88% ("wet") to 5% ("dry") during testing, the change was reflected on both the web dashboard and the Telegram bot. Similarly, when the ultrasonic sensor detected a water level at 0–1%, the system sent a low-water notification to Telegram and prevented the pump from operating.

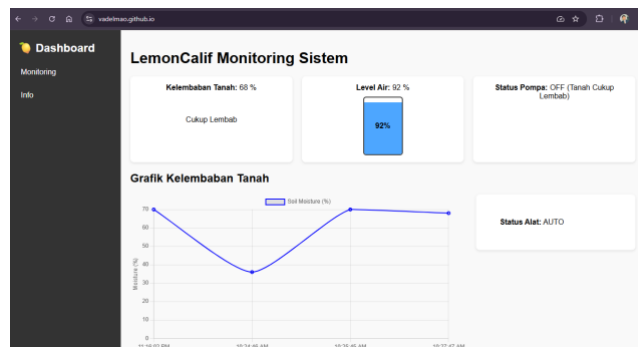


Fig 13: Web Dashboard Showing Sensor Data During Testing

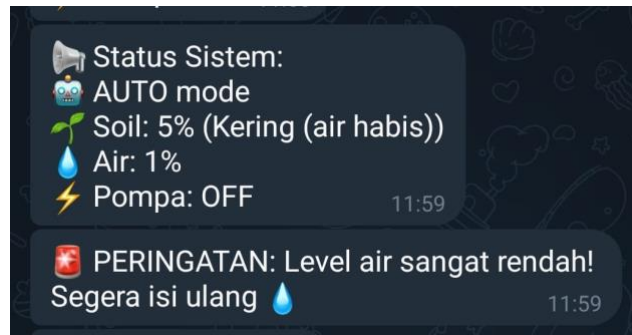


Fig 14: Telegram Notification During Water Depletion

Manual control testing demonstrated that users could switch between automatic and manual modes via Telegram commands. In manual mode, the pump toggled on or off according to user input, and the mode change was reflected on the web dashboard in real time.

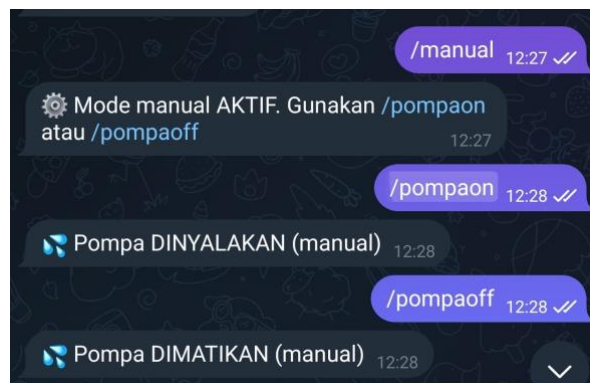


Fig 15: Manual Control via Telegram



Fig 16: Mode Status on Telegram Manual Control

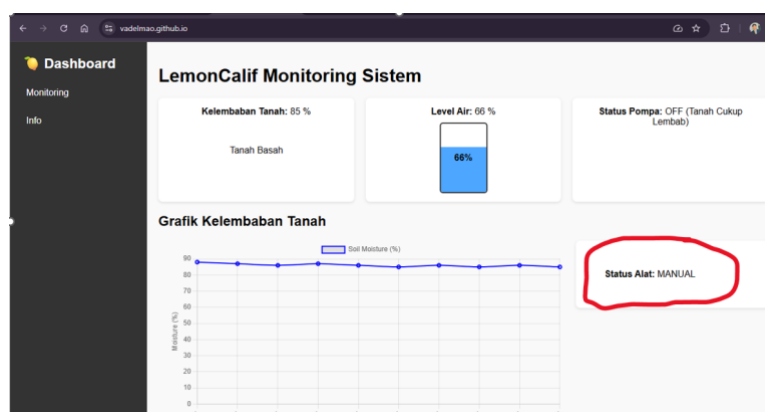


Fig 17: Mode Status on Web Dashboard

4.6. System Logic Validation

Table 4 summarizes the tested system logic under various conditions.

Table 5: Pump Activation Conditions

No.	Condition	Pump Action
1	Soil moisture < 40% (dry)	Pump activates until moisture reaches 50–75%
2	Soil moisture 50–75% (moist) or > 80% (wet)	Pump remains off
3	Soil moisture < 40% and Water level < 20%	Pump remains off
4	Manual activation via Telegram bot	Pump runs until manually deactivated by user

The system consistently followed the designed logic across all test scenarios, confirming reliable automated watering behavior.

4.7. Water Efficiency Analysis

A comparative evaluation of water consumption was conducted over three test days during a period of intermittent rain and sunny weather. Manual watering, as reported by the end user (a local farmer), consumed an average of 600 ml per day. The automated system, which activates the pump only when the soil moisture sensor detects a reading below 40%, consumed an average of approximately 479 ml per day. This represents a water savings of roughly 20% compared to the manual method.

The reduction in water usage is attributable to the sensor-driven activation logic, which prevents unnecessary watering when soil moisture is already adequate. In manual practice, farmers may water out of habit or uncertainty, leading to overwatering. The automated system eliminates this inefficiency by basing irrigation decisions on actual soil conditions rather than subjective judgment.

Day	Weather Condition	Soil Moisture (%)	Water Tank Level (%)	System Action	Water Usage (mL)
1	Normal	38	92	Pump ON	450
2	Warm	35	89	Pump ON	500
3	Hot	32	83	Pump ON	520
4	Very Hot	30	74	Pump ON	552
5	Extremely Hot	25	64	Pump ON	601
6	Extremely Hot	20	50	Pump ON	650
7	Hot & Windy	28	35	Pump ON	576
8	Cloudy	37	29	Pump ON	461
9	Heavy Rain	85	60	Pump OFF	0
Average :					478.8

4.8. Limitations

During testing and field deployment, several system limitations were identified:

1. **Sensor Degradation.**

The YL-69 is a resistive soil moisture sensor that is susceptible to corrosion from prolonged contact with water and soil [26]. The 5-second reading interval, while necessary for responsive pump control, accelerates physical wear on the sensor probes.

2. **Power Dependency.**

The system relies entirely on a 12V DC power supply with no backup battery. A power outage disables all system functions, including automatic watering and remote monitoring.

3. **Internet Dependency.**

All cloud communication—from Firebase data uploads to Telegram notifications and commands—requires a stable Wi-Fi connection. Network interruptions halt data transmission and disable remote control, leaving the system without monitoring capability [20].

4. **No Local Data Logging.**

The system does not store sensor data locally. Any data generated during a network outage is lost and cannot be recovered or synchronized once connectivity is restored.

5. Conclusion

This study successfully designed, implemented, and tested an IoT-based automatic monitoring and watering system for California lemon tree seedlings. The system integrates an ESP32 microcontroller with a YL-69 soil moisture sensor and HC-SR04 ultrasonic sensor, delivering real-time data to a Firebase-backed web dashboard and a Telegram bot named "AutoLemon."

The findings support three main conclusions. First, the system reliably performs dual monitoring of soil moisture and water tank capacity, and controls the water pump automatically based on predefined moisture thresholds. The pump activates when soil moisture drops below 40% and deactivates once the 50–75% range is reached, consistent with the optimal moisture requirements for lemon seedlings identified in the literature [7]. Second, the integration of a web dashboard and Telegram bot provides a practical dual-interface solution for remote monitoring and control [2], [22]. Users can observe soil and water conditions in real time, receive automatic notifications, and operate the pump manually from any location via their smartphones. Third, the automated system demonstrates measurable improvements in water efficiency compared to manual watering. Over three test days, the system reduced daily water consumption from an average of 600 ml (manual) to approximately 479 ml, yielding a water savings of roughly 20%. This improvement stems from the sensor-driven irrigation logic that eliminates unnecessary watering.

The system does, however, have limitations that warrant further development. Replacing the resistive soil moisture sensor with a capacitive type would improve durability. Adding a local data logging mechanism, such as a MicroSD card, would preserve sensor data during network outages. Implementing a backup power source would ensure uninterrupted operation during electricity failures. Future work should also explore the integration of additional sensors—for soil pH, ambient temperature, or light intensity—to enable more comprehensive plant health monitoring [23].

6. References

- [1] F. E. Subagja et al., "Pengujian sistem penyiraman tanaman otomatis berbasis IoT," *Infotronik: Jurnal Teknologi Informasi dan Elektronika*, vol. 8, no. 2, pp. 91–97, 2023, doi: 10.32897/infotronik.2023.8.2.3015.
- [2] J. Prasetyo, M. I. Ashari, and M. Ardita, "Rancang bangun sistem monitoring dan kontrol penyiraman tanaman kentang otomatis berbasis IoT dengan integrasi bot Telegram," *Magnetika: Jurnal Mahasiswa Teknik Elektro*, vol. 8, no. 2, pp. 204–212, 2024.
- [3] Jatengprov.id, "Petani jeruk lemon California," 2020. [Online]. Available: <https://jatengprov.go.id/beritadaerah/hasil-lebih-menguntungkan-petani-cabai-beralih-tanam-jeruk-lemon-california/>
- [4] S. Shiyani et al., "Narrative review: Profil fitokimia dan potensi farmakologi Citrus limon," *Jurnal 'Aisyiyah Medika*, vol. 7, no. 2, 2022, doi: 10.36729/jam.v7i2.851.
- [5] N. O. Cindiya, N. Islami, and Muyassaroh, "Ekstraksi kulit jeruk lemon (Citrus limon) dengan variasi perlakuan bahan dan daya menggunakan metode microwave assisted extraction (MAE)," *Jurnal ATMOSPHERE*, vol. 4, no. 2, pp. 21–26, 2023, doi: 10.36040/atmosphere.v4i2.8308.
- [6] Y. K. Firdaus et al., "Maturity parameters characterization and classification of Lemon (Citrus limon (cv. California)) based on reflectance-fluorescence imaging and machine learning model," *International Journal of Agricultural Technology*, vol. 21, no. 2, pp. 507–522, 2025, doi: 10.63369/ijat.2025.21.2.507-522.
- [7] S. Safiah et al., "Lama perendaman dan konsentrasi yang berbeda terhadap pertumbuhan stek pucuk jeruk lemon (Citrus limon) menggunakan ZPT alami air kelapa," *JURNAL ILMIAH EDUNOMIKA*, vol. 7, no. 2, 2023, doi: 10.29040/jie.v7i2.9993.
- [8] C. J. Gea and E. Mendrofa, "Pengaruh sifat termal tanah terhadap pertumbuhan tanaman hortikultura diberbagai zona iklim," *Jurnal Ilmu Pertanian dan Perikanan*, vol. 1, no. 1, pp. 153–158, 2024, doi: 10.70134/penarik.v2i2.84.
- [9] O. J. Camberato, "Keep in mind soil test K and pH are affected by low soil moisture," *Pest&Crop Newsletter*, 2022. [Online]. Available: <https://extension.entm.purdue.edu/newsletters/pestandcrop/article/keep-in-mind-soil-test-k-and-ph-are-affected-by-low-soil-moisture/>
- [10] N. Effendi, W. Ramadhani, and F. Farida, "Perancangan sistem penyiraman tanaman otomatis menggunakan sensor kelembapan tanah berbasis IoT," *Jurnal CoSciTech (Computer Science and Information Technology)*, vol. 3, no. 2, pp. 91–98, 2022, doi: 10.37859/coscitech.v3i2.3923.
- [11] M. Ridwan, "Perancangan sistem penyiraman tanaman otomatis," *Jurnal Sains dan Ilmu Terapan*, vol. 4, no. 2, pp. 30–33, 2021, doi: 10.59061/jsit.v4i2.92.
- [12] Akbar, "Implementasi Internet of Things untuk monitoring kelembapan tanah menggunakan mikrokontroler," *Jurnal Kecerdasan Buatan dan Teknologi Informasi*, vol. 2, no. 2, pp. 91–97, 2023, doi: 10.69916/jkbt.v2i2.32.
- [13] T. P. Utomo, "Potensi implementasi Internet of Things (IoT) untuk perpustakaan," *Buletin Perpustakaan*, vol. 2, no. 1, pp. 1–18, 2019.
- [14] M. S. Gitakarma and L. P. A. S. Tjahyanti, "Peranan Internet of Things dan kecerdasan buatan dalam teknologi saat ini," *KOMTEKS*, vol. 1, no. 1, 2022. [Online]. Available: <https://ejournal.unipas.ac.id/index.php/Komteks/article/view/1060>
- [15] M. Rashid and U. I. Wani, "Role of fog computing platform in analytics of Internet of Things—issues, challenges and opportunities," in *Fog, Edge, and Pervasive Computing in Intelligent IoT Driven Applications*, John Wiley & Sons, Ltd, 2020, pp. 209–220, doi: 10.1002/9781119670087.ch12.
- [16] Khuriati et al., "Pembelajaran IoT menggunakan aplikasi Telegram dan Blynk bagi siswa jurusan IPA SMA Negeri I Nguter," *BERKALA FISIKA*, vol. 26, no. 2, pp. 66–70, 2023.
- [17] G. P. Haryanto, "Control motor pompa air daur ulang STP berbasis Arduino dengan sensor kelembapan tanah," *JURNAL MEDIA INFOTAMA*, vol. 20, no. 2, pp. 706–712, 2024, doi: 10.37676/jmi.v20i2.6636.
- [18] Z. Mardiyansyah, "Sistem pengontrol irigasi otomatis menggunakan mikrokontroler Arduino Uno," 2019. [Online]. Available: <https://www.academia.edu/90411652/>
- [19] R. Pratama, "Penerapan sensor ultrasonik pada kotak sampah otomatis menggunakan Telegram dan alarm suara," *Jurnal Teknik dan Sistem Komputer*, 2023. [Online]. Available: <https://www.academia.edu/112853819/>
- [20] H. Sofiumayroh and H. Hasfani, "Pemanfaatan protokol HTTP pada sistem monitoring suhu air menggunakan website berbasis Internet of Things (IoT)," *Coding: Jurnal Komputer dan Aplikasi*, vol. 12, no. 2, pp. 102–111, 2024, doi: 10.26418/coding.v12i2.84050.
- [21] E. B. Agustina and A. T. Dewi, "Perancangan alat penyiram tanaman otomatis berbasis mikrokontroler dan Internet of Things (IoT)," *Lontar Physics Today*, vol. 2, no. 3, pp. 98–106, 2023, doi: 10.26877/lpt.v2i3.18031.
- [22] U. L. A. Pangestu, "Perancangan sistem monitoring jarak jauh pada smart agriculture system," 2023. [Online]. Available: <https://core.ac.uk/reader/555452800>
- [23] B. Sugandi, "Monitoring suhu, pH dan kelembapan untuk aplikasi smart farming berbasis Internet of Things," *Electrician: Jurnal Rekayasa dan Teknologi Elektro*, vol. 19, no. 3, pp. 394–403, 2025, doi: 10.23960/elc.v19n3.2920.
- [24] M. Muliadi, A. Imran, and M. Rasul, "Pengembangan tempat sampah pintar menggunakan ESP32," *Jurnal Media Elektrik*, vol. 17, no. 2, pp. 73–79, 2024, doi: 10.59562/metrik.v17i2.5398.
- [25] E. W. Pratama and A. Kiswanto, "Electrical analysis using ESP-32 module in realtime," *JEECS (Journal of Electrical Engineering and Computer Sciences)*, vol. 7, no. 2, pp. 1273–1284, 2022, doi: 10.54732/jeeecs.v7i2.21.

- [26] H. Husdi, "Monitoring kelembaban tanah pertanian menggunakan soil moisture sensor FC-28 dan Arduino Uno," *ILKOM Jurnal Ilmiah*, vol. 10, no. 2, pp. 237–243, 2018, doi: 10.33096/ilkom.v10i2.315.237-243.
- [27] G. M. Vuyyuru and M. R. Shirke, "Performing basic tasks on computer using hand gestures & ultrasonic sensors," *International Journal of Engineering Research & Technology*, vol. 10, no. 5, 2021, doi: 10.17577/IJERTV10IS050279.
- [28] D. A. Jakaria and M. R. Fauzi, "Aplikasi smartphone dengan perintah suara untuk mengendalikan saklar listrik menggunakan Arduino," *JURNAL TEKNIK INFORMATIKA (JUTEKIN)*, vol. 8, no. 1, 2020, doi: 10.51530/jutekin.v8i1.462.
- [29] Hendrayudi, Jepri, and Salamudin, "Rancang bangun sistem keamanan kendaraan sepeda motor menggunakan sidik jari berbasis Arduino Uno," *JIK: Jurnal Informatika dan Komputer*, vol. 13, no. 1, pp. 27–33, 2022.
- [30] D. W. Hermanto et al., "Sistem monitoring dan pengukuran pembangkit listrik surya dan angin berbasis Internet of Things (IoT)," *JURNAL TEKNIK ELEKTRO*, vol. 11, no. 3, pp. 371–378, 2022, doi: 10.26740/jte.v11n3.p371-378.
- [31] Z. Didi and I. E. Azami, "Monitoring of submersible pumps using ESP32 microcontroller and photovoltaic panels," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 3, pp. 1470–1477, 2023, doi: 10.11591/ijeecs.v30.i3.pp1470-1477.
- [32] R. S. Martin and Y. Dewanto, "Prototipe kunci pintu otomatis menggunakan sensor kamera berbasis Raspberry," *JURNAL TEKNOLOGI INDUSTRI*, vol. 12, no. 1, 2023, doi: 10.35968/jti.v12i1.1044.
- [33] S. Hasyati et al., "Perancangan sistem informasi pemesanan interior pada PT Cipta Kreasi Buana berbasis web menggunakan metode prototype," *Jurnal Riset Teknik Komputer*, vol. 2, no. 1, pp. 10–20, 2025, doi: 10.69714/0astw770.
- [34] Z. A. Muhtadi and L. Junaedi, "Implementasi metode prototype dalam membangun sistem informasi penjualan online pada toko herbal pahlawan," *Journal of Advances in Information and Industrial Technology*, vol. 3, no. 1, pp. 31–41, 2021, doi: 10.52435/jaiit.v3i1.88.
- [35] M. Maulana, "HTTP adalah: Pengertian, fungsi & cara kerja," *ITBOX by Course-Net*, Mar. 16, 2023. [Online]. Available: <https://itbox.id/blog/http-adalah>
- [36] N. Wilyanto et al., "Pembuatan website menggunakan Visual Studio Code di SMA Xaverius 3 Palembang," *FORDICATE*, vol. 3, no. 1, pp. 1–8, 2023, doi: 10.35957/fordicate.v3i1.5057.