E-ISSN 2599-2872; P-ISSN 2549-8150

J-PEN Borneo: Jurnal Ilmu Pertanian

journal homepage: https://jurnal.ubt.ac.id/index.php/jpen

Research Paper 8 (1) (2025) 31 - 38

IoT-Based Automatic Plant Watering System Using a Soil Moisture Sensor in Tolitoli Regency

Siti Fatima[1*], Sjarifuddin Ende[1], Masriani[1], Andre Agasi S. Laumarang[1], Vica Ivana Sallomita Zeruya[1], Abdhika Mega Cendani[1], and Ince Siti Wardatullatifah[2]

1 Agrotechnology Study Program, YPP Mujahidin Tolitoli College of Agricultural Sciences (STIP), Tolitoli, Indonesia

2 Universitas Mataram, Mataram, Indonesia

* Corresponding author. E-mail address: sitifatima1414tima89@gmail.com

Submitted/Received: 03 February 2025, Accepted: 27 March 2025

ARTICLE INFO

Keywords:
Internet of Things
Automatic watering
Soil moisture sensor
Telegram Bot

ABSTRACT

This study aimed to design and build an Internet of Things (IoT)-based automatic plant watering system capable of detecting soil moisture and regulating irrigation independently. The prototype method within the Software Development Life Cycle (SDLC) was used, comprising requirements analysis, system design, prototype development, testing, and evaluation. A NodeMCU ESP8266 served as the controller, a capacitive soil moisture sensor measured soil water content, a DHT11 monitored air temperature and relative humidity, and a relay module controlled the water pump. Testing showed that the soil moisture sensor had a mean error of 3.3% and an accuracy of 96.7%. The DHT11 achieved mean accuracies of 93.3% for temperature and 94.0% for relative humidity. The pump was activated automatically when soil moisture fell below 50% and stopped when moisture reached the programmed 50-75% range. Integration with a Telegram Bot enabled real-time remote monitoring and control. The prototype supports smart-farming applications through an efficient, water-conscious, and accessible automated irrigation system for farmers and household users.

INTRODUCTION

Water availability is a decisive factor in plant growth because both the timing and amount of irrigation influence physiological performance and crop productivity. Water deficits can cause wilting, reduce cellular expansion, and limit plant development, whereas excessive watering may create poorly aerated root-zone conditions, promote root decay, and reduce productivity [1]. Efficient water management has therefore become an important concern in modern agriculture, particularly under increasingly variable climatic conditions and limited water resources [4]. An irrigation system should deliver water when plants need it while avoiding unnecessary application.

In many small-scale agricultural and household settings, irrigation and soil-condition monitoring are still performed manually. A fixed schedule or visual assessment does not necessarily represent the actual moisture condition of the growing medium. Consequently, watering may occur too early, too late, or in an unsuitable amount. Such practices can create an imbalance in soil water content and negatively affect plant growth [12]. Manual operation also requires users to be physically present and makes continuous monitoring difficult, especially when environmental conditions change rapidly.

The Internet of Things (IoT) provides a practical route toward sensor-based monitoring and automated control. An IoT system can acquire field data, process predefined decision rules, actuate equipment, and communicate the system status to a user in real time. Soil moisture sensors are particularly relevant because they allow irrigation decisions to be linked to the measured condition of the growing medium rather than to a fixed timetable. Remote connectivity further enables users to observe and control irrigation without direct access to the equipment. These capabilities have encouraged the

adoption of IoT innovations in agriculture [3] and in greenhouse agribusiness applications [5].

Previous studies have developed automated watering and monitoring systems using Arduino or other controllers [1], soil moisture sensors and IoT communication [2], Android and Blynk interfaces [12], automatic watering and fertilization [13,15], SMS-based soil moisture monitoring [14], and fuzzy control in greenhouses [8]. These studies demonstrate the feasibility of digital irrigation, but implementation remains dependent on the local equipment configuration, sensor calibration, operating thresholds, and user interface. A practical gap therefore remains for a locally assembled prototype that combines soil moisture sensing, environmental monitoring, automatic pump control, an on-device display, alarms, and an accessible Telegram interface in one system. The present design addresses this gap in the context of Tolitoli Regency.

For an automated watering device to be useful outside a laboratory demonstration, its sensing, decision, actuation, and communication layers must operate as one reliable chain. A moisture sensor may respond correctly while watering still fails because the threshold is unsuitable, the relay does not switch, the pump is not supplied correctly, or the user cannot confirm the system status. Environmental readings also need to be displayed clearly so that users can interpret the operating context. Local assembly and testing are therefore necessary to determine whether the selected components produce stable readings, whether the programmed limits trigger the intended actions, and whether remote commands and notifications are understandable during practical operation. Evaluating these functions together provides a stronger basis for subsequent agronomic field trials than testing each electronic component in isolation.

Accordingly, this study aimed to design, build, and test an IoT-based automatic plant watering system using a NodeMCU ESP8266, a capacitive soil moisture sensor, a DHT11 sensor, relay-controlled pumps, and a Telegram Bot. The prototype was evaluated through component-level alpha testing, soil moisture sensor comparison, temperature and relative-humidity comparison with local BMKG weather data, pump-response observations, and verification of Telegram commands and activity logs. The principal contribution is an integrated and remotely accessible prototype that applies measured soil moisture thresholds to automatic irrigation. The study evaluates operational accuracy and functionality; it does not directly quantify crop yield or total water savings.

MATERIALS AND METHODS

Research site and time

The system was designed and assembled from May to August 2025. Prototype testing and data collection were conducted on Samratulangi Street, Tuweley Urban Village, Baolan District, Tolitoli Regency, Central Sulawesi, Indonesia. The study focused on functional performance, sensor comparison, and automated control under the environmental conditions present at the test site.

Materials, tools, and research design

The study used a prototype-development design within the Software Development Life Cycle. The principal electronic components were a NodeMCU ESP8266 Lolin controller, a capacitive V2.0 soil moisture sensor, a DHT11 temperature and relative-humidity sensor, a two-channel relay, two water pumps, a 16 x 2 I2C LCD, indicator LEDs, a passive buzzer, and supporting power and plumbing components. A laptop and smartphone were used for programming, configuration, and interaction with the Telegram Bot. The complete list is provided in Table 1.

Table 1. Tools and materials used to construct the prototype

No.	Tool or material	Quantity
1	Laptop	1
2	Smartphone	1
3	Soldering set	1 set
4	Bolts and clamps	As required
5	NodeMCU ESP8266, Lolin version	1
6	Soil moisture sensor	1
7	DHT11 sensor	1
8	Two-channel relay	1
9	Water pump	2
10	Jumper wires	As required
11	LEDs (red, yellow, and green)	As required
12	Hose	2 m
13	Passive buzzer	1
14	Baseboard	1

No.	Tool or material	Quantity
15	Enclosure box	1
16	16 x 2 I2C LCD	1
17	12 V, 5 A DC adapter	2

Prototype development

The prototype method was selected because it permits iterative development and refinement. The work began with requirements analysis, including the need to measure soil moisture, observe air temperature and relative humidity, control the pump automatically and manually, display local information, and provide remote communication. The system was then designed by defining the controller, sensors, actuators, power supply, communication flow, and watering logic. A working prototype was assembled and programmed, after which its functions were tested and refined on the basis of the observed responses. This cycle continued until the final prototype met the stated functional requirements. The procedure followed the prototype SDLC concept described by Sunandar and Herlawat [10] and Putri et al. [11] (Figure 1).

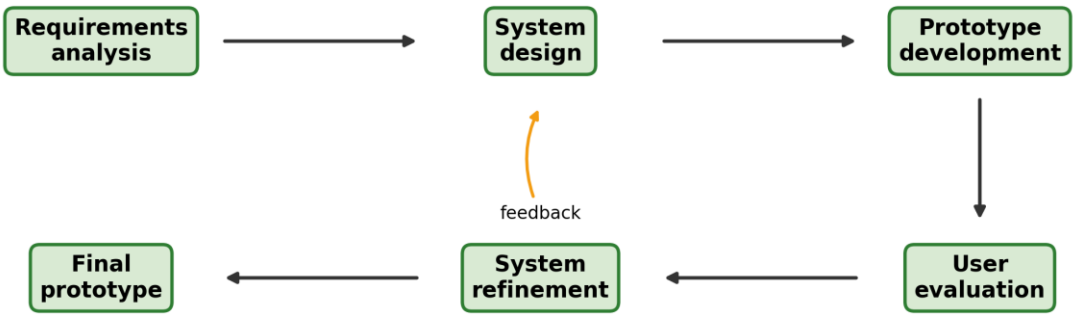


Figure 1. Prototype Software Development Life Cycle used in the study

Source: adapted from Sunandar and Herlawat [10].

System operation

The automatic watering system was designed to irrigate without continuous manual intervention. The soil moisture sensor detected the moisture condition of the growing medium and transmitted the reading to the NodeMCU ESP8266. The controller processed this value, updated the LCD, and applied the programmed rule to the relay and pump. Soil conditions were classified as dry, moist or normal, and wet. When the soil was dry and the moisture reading was below 50%, the controller activated the pump. Irrigation was stopped when the reading entered the programmed 50-75% range. The user could also operate the pump manually through the Telegram interface.

The capacitive soil moisture sensor responded to changes in water content and produced a signal that the controller converted into a percentage value. This percentage was used as the decision variable for pump activation. The DHT11 simultaneously supplied air-temperature and relative-humidity data. Telegram commands enabled the user to start manual or automatic operation, view garden status, check pump status, and review recent watering activity. Notifications were sent when the NodeMCU connected to the network and whenever relevant watering actions occurred.

Observed parameters and data analysis

Soil moisture sensor performance was evaluated using 11 containers, each holding 100 g of oven-dried soil. One container served as the 0 mL water control, and the other containers received water additions from 10 to 100 mL in 10 mL increments. The sensor was inserted into each prepared sample, and three readings (U1, U2, and U3) were recorded. The mean sensor reading was compared with the manually assigned moisture reference based on the water addition. Relative error was calculated as |manual reference - sensor mean| / sensor mean x 100%, following the calculation used in the original dataset; accuracy was calculated as 100% - error. For example, the 10% reference

and 9% sensor mean produced an error of |10 - 9| / 9 x 100% = 11.1%.

DHT11 performance was assessed by recording temperature and relative humidity at 14:37, 15:35, 20:22, and 20:32 on one test day. The readings were compared descriptively with corresponding weather data from the Indonesian Agency for Meteorology, Climatology, and Geophysics (BMKG). Error and accuracy percentages were summarized for the four observations. No inferential statistical test was applied.

Pump response was evaluated from the moisture reading before and after watering, start and finish times, watering duration, and manual or automatic status. The pump was expected to activate automatically below the 50% threshold and to stop after the target range was reached. Telegram functionality was verified by observing bot responses, commands, notifications, garden-status information, pump status, and stored watering logs in real time.

RESULTS

Alpha testing

Alpha testing was conducted internally before user-level testing to verify that each component performed its intended basic function. As shown in Table 2, the NodeMCU and baseboard powered and connected to Wi-Fi; the LCD displayed startup information, soil moisture, and pump status; both sensors returned readings; the buzzer responded to dry-soil conditions; the relay operated the pumps; and the Telegram Bot exchanged commands and notifications with the NodeMCU. All seven test items were successful.

Table 2. Alpha-testing results for the prototype components

No.	Input/component	Expected output	Result
1	NodeMCU ESP8266 and baseboard	Indicator lights turn on and the controller connects to Wi-Fi	Successful
2	16 x 2 I2C LCD	Displays startup text, soil moisture, and pump status	Successful
3	Soil moisture sensor	Detects soil moisture	Successful
4	DHT11 sensor	Measures air temperature and relative humidity	Successful
5	Buzzer	Sounds when the soil is dry	Successful
6	Relay and pump	Relay switches the pump in response to the moisture rule	Successful
7	Telegram Bot	Sends commands to the NodeMCU and displays notifications	Successful

Soil moisture sensor performance

The sensor test compared three capacitive V2.0 readings with manual reference values for progressively wetted soil (Table 3). The 0% control was reported as unreadable for calibration purposes. At the 10% reference, U1, U2, and U3 were 7%, 9%, and 11%, respectively, giving a mean of 9% and the highest calculated error of 11.1%. From 20% to 100%, mean readings ranged from 19.3% to 97.0%. The smallest error was 0.6% at the 60% reference, while the largest error after the 10% treatment was 6.2% at the 40% reference.

Table 3. Soil moisture sensor readings and relative error

No.	Soil (g)	Water (mL)	Manual (%)	U1 (%)	U2 (%)	U3 (%)	Sensor mean (%)	Error (%)	Reading
1	100	0	0	0	0	0	—	—	Unreadable
2	100	10	10	7	9	11	9.0	11.1	Readable
3	100	20	20	19	18	21	19.3	3.4	Readable
4	100	30	30	28	30	31	29.7	1.1	Readable
5	100	40	40	37	39	37	37.7	6.2	Readable
6	100	50	50	49	47	51	49.0	2.0	Readable
7	100	60	60	59	62	58	59.7	0.6	Readable
8	100	70	70	68	71	69	69.3	1.0	Readable
9	100	80	80	77	79	76	77.3	3.4	Readable
10	100	90	90	88	91	89	89.3	0.7	Readable
11	100	100	100	96	98	97	97.0	3.1	Readable

M/manual = manually assigned comparison value; U1-U3 = sensor replicates. Mean sensor reading = 53.7%; mean error = 3.3%; overall accuracy = 96.7%. Decimal points are standardized from the original comma notation.

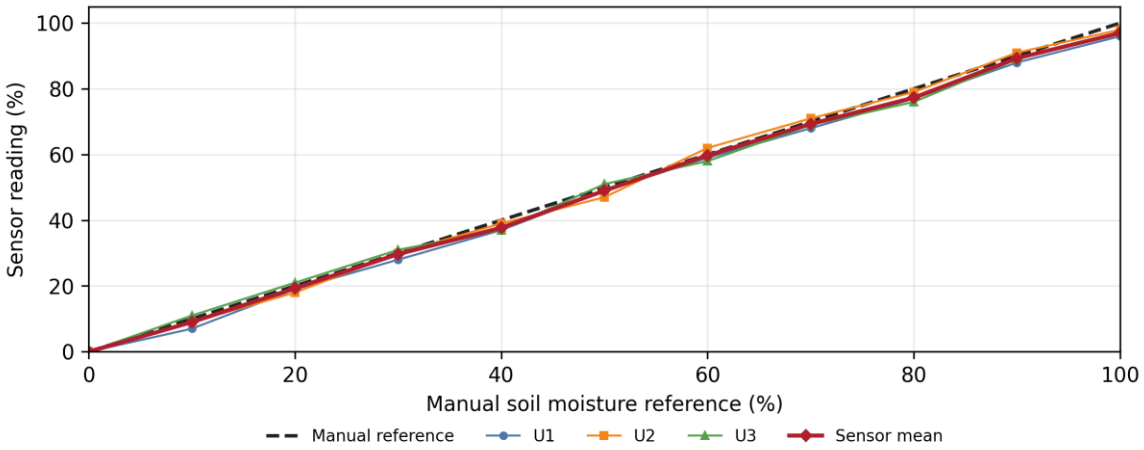


Figure 2. Comparison of manual soil moisture references and three sensor replicates

Across the readable treatments, the mean error was 3.3%, corresponding to an overall accuracy of 96.7%. Sensor response increased approximately in proportion to the amount of water added, from the low-moisture treatments to the near-saturated treatment (Figure 2). The transition from dry to moist conditions was clearly represented across the 30-70 mL water additions. The sensor therefore distinguished dry, moist, and wet conditions within the tested range, although performance was weakest below approximately 10% moisture.

Air temperature and relative humidity

DHT11 readings were compared with BMKG weather data at four times during one day. Temperature errors ranged from 6.3% to 7.1%, with individual accuracies of 92.9-93.8% and a mean accuracy of 93.3% (Table 4). Relative-humidity errors ranged from 3.8% to 9.0%, with accuracies of 91.0-96.2% and a mean accuracy of 94.0% (Table 5).

Table 4. DHT11 air-temperature readings compared with BMKG data

No.	Time	DHT11 temperature (°C)	BMKG temperature (°C)	Error (%)	Accuracy (%)
1	14:37	32	30	6.3	93.8
2	15:35	31	29	6.5	93.5
3	20:22	29	31	6.9	93.1
4	20:32	28	26	7.1	92.9
Mean	—	—	—	6.7	93.3

Table 5. DHT11 relative-humidity readings compared with BMKG data

No.	Time	DHT11 relative humidity (%)	BMKG relative humidity (%)	Error (%)	Accuracy (%)
1	14:37	78	71	9.0	91.0
2	15:35	86	82	4.7	95.3
3	20:22	79	76	3.8	96.2
4	20:32	76	71	6.6	93.4
Mean	—	—	—	6.0	94.0

Pump operation and watering history

Eight watering events were recorded: five automatic events and three manual events (Table 6). Automatic watering began at initial soil moisture values from 3% to 45%. For the first three automatic events, moisture increased from 9% to 89%, from 3% to 86%, and from 20% to 88%, respectively. The three manual tests began at 5%, 5%, and 6% and ended at 62%, 97%, and 85%. Watering duration ranged from 16 to 66 s. Event 2 increased

moisture from 3% to 86% in 61 s, whereas event 7 increased it from 45% to 79% in 16 s.

Table 6. Watering history and pump status

No.	Start	Finish	Duration	Before (%)	After (%)	Manual	Automatic
1	08:00:46	08:01:22	36 s	9	89	OFF	ON
2	14:28:25	14:29:26	61 s	3	86	OFF	ON
3	14:30:20	14:30:46	26 s	20	88	OFF	ON
4	15:39:57	15:40:18	21 s	5	62	ON	OFF
5	15:47:12	15:48:19	66 s	5	97	ON	OFF
6	16:28:54	16:29:14	20 s	6	85	ON	OFF
7	17:28:39	17:28:55	16 s	45	79	OFF	ON
8	18:47:48	15:48:19	31 s	5	71	OFF	ON

The finish time for event 8 is reproduced from the source record and should be verified because it precedes the recorded start time.

Telegram notifications and activity logs

The Telegram interface provided five principal interaction paths (Table 7). The /start command opened the navigation menu; /manual activated manual watering and returned a confirmation; /automatic enabled sensor-based watering and generated a log when moisture was below 50%; /status garden displayed temperature, relative humidity, pump status, and recent watering history; and an automatic welcome message confirmed successful NodeMCU and bot connectivity.

Table 7. Telegram Bot features in the automatic watering system

No.	Menu/command	Submenu or option	Function	Log/notification
1	/start	Manual; Automatic; Garden Status; Hide Menu	Displays the main navigation menu	No specific notification or activity log
2	/manual	Watering; Back	Activates the pump manually	Bot displays: 'Manual watering active'
3	/automatic	Watering; Back	Waters automatically using sensor data	Bot sends: 'Automatic watering active, moisture <50%'
4	/status garden	Temperature and humidity; Pump status; Watering history	Monitors garden status in real time	Displays environmental data, pump ON/OFF status, and the latest watering log
5	Automatic welcome	None	Sends a message when NodeMCU connects	'NodeMCU connected! Welcome to the IoT smart-garden system'

The automated watering log recorded soil moisture, two paired temperature values, two paired relative-humidity values, and pump status (Table 8). At soil moisture values of 5% and 18%, the pump was ON. At 96% and 98%, the pump remained OFF. Thus, the observed pump states followed the programmed moisture threshold in all four logged trials.

Table 8. Automatic-watering activity log

No.	Soil moisture (%)	Temperature (°C)	Relative humidity (%)	Pump
1	5	32/30	78/71	ON
2	18	31/29	86/82	ON
3	96	29/31	79/76	OFF
4	98	28/26	76/71	OFF

Paired temperature and relative-humidity values are reproduced as reported in the source dataset.

DISCUSSION

The alpha-test results show that the controller, sensors, output devices, and communication interface functioned as an integrated system. Successful component operation is important because the watering decision depends on a complete chain: the sensor must detect the soil condition, the NodeMCU must process the reading, the relay must switch the pump, and the interface must communicate the outcome. The results therefore verify functional integration rather than isolated sensor performance alone. This integrated architecture is consistent with the direction of earlier automatic watering systems that combined a controller, soil moisture input, and remote monitoring [1,2,7,12-15].

The soil moisture sensor provided its most important quantitative performance result: a mean error of 3.3% and an overall accuracy of 96.7% across the reported readable treatments. The approximately linear rise in sensor response as water addition increased indicates that the device was sensitive to changes over a broad moisture range. The error was greatest at the 10% reference and the 0% condition was unsuitable for the reported calibration, showing that the sensor was less dependable at the very dry end of the range. Above 20%, most readings approached the manual reference and the 60% treatment produced only 0.6% error. This behavior supports use of the sensor for threshold-based watering, while also indicating that field calibration should emphasize the dry range where pump activation occurs.

The DHT11 comparisons produced mean accuracies of 93.3% for temperature and 94.0% for relative humidity. These values indicate adequate agreement for contextual garden monitoring in this prototype, but the sensor did not control the pump directly; the irrigation decision was based on soil moisture. The comparison was limited to four measurements on one day and to BMKG weather data rather than a colocated laboratory reference instrument. Consequently, the reported DHT11 accuracy should be interpreted as a descriptive site comparison, not as a comprehensive calibration across seasons or microclimates.

Pump behavior followed the programmed control logic. Low initial moisture values triggered automatic watering, and post-watering readings increased substantially. Shorter durations were observed when the initial soil was less dry, whereas some very dry conditions required longer operation. Pump pressure could also be adjusted, so duration cannot be attributed to soil moisture alone. The manual mode likewise increased moisture and demonstrated that user intervention remained available. These findings establish responsiveness and operational control, although the study did not measure discharge volume, water-use efficiency, plant growth, or yield. Claims regarding water savings should therefore be treated as expected practical benefits that require direct verification in subsequent trials.

Telegram integration increased the usability of the prototype by allowing remote selection of manual and automatic modes, access to environmental and pump status, and review of watering history. Immediate messages also improved transparency by confirming whether the controller was connected and whether a watering action had been initiated. The use of a familiar messaging application reduces the need for a separate dedicated mobile application. However, remote functionality remains dependent on stable internet connectivity, correct bot configuration, and secure access to the control commands.

Overall, the prototype met the intended functional objective and offers a feasible foundation for small-scale smart farming in Tolitoli. Future work should repeat calibration with a gravimetric soil-water reference, increase the number of soil types and moisture replicates, test multiple crops and growth stages, record water discharge and energy use, evaluate long-term sensor drift, and assess system performance across wet and dry seasons. The time inconsistency in the eighth watering record and the paired values in the activity log should also be checked against the original device logs before publication. These steps would strengthen data traceability and allow the agronomic and resource-efficiency benefits of the system to be quantified.

CONCLUSION

An IoT-based automatic plant watering system was successfully designed and operated using a NodeMCU ESP8266, a capacitive soil moisture sensor, a DHT11 sensor, relay-controlled pumps, an LCD, and a Telegram Bot. The soil moisture sensor achieved a mean error of 3.3% and an accuracy of 96.7%, while the DHT11 achieved mean accuracies of 93.3% for temperature and 94.0% for relative humidity in the four-point comparison. The pump activated automatically below 50% soil moisture and stopped after the programmed target range was reached. Telegram integration enabled real-time remote monitoring, notifications, and manual or automatic control. The prototype therefore met the study objective and provides a practical platform for further field validation of water use, reliability, and crop response.

REFERENCES

[1] Al-Hafiz, N. W., & Erlinda. (2020). Design of an automatic plant watering system using Arduino. Jurnal Teknologi dan Open Source, 3(2), 245-260.

[2] Effendi, N., Ramadhani, W., Farida, F., & Dimas, M. (2022). Design of an IoT-based automatic plant watering system using a soil moisture sensor. Jurnal Computer Science and Information Technology (CoSciTech), 3(2), 91-98.

[3] Harsanto, B. (2020). Internet of Things innovation in the agricultural sector: A scientometric analysis approach. Informatika Pertanian, 29(2), 111-122.

[4] Nanda, A. P., Jeprianto, & Mahdi, M. I. (2024). Automatic soil-moisture-sensor-based plant watering system for increasing agricultural productivity. Technologia: Jurnal Ilmiah, 15(4), 764-774.

[5] Nasution, N., & Hasan, M. A. (2020). IoT in agribusiness: A case study of lettuce in a greenhouse. IT Journal Research and Development (ITJRD), 4(2), 86-93.

[6] Rahmat, S., Ikhsanudin, M., Diani, R., Kusuma, Y. F., Suadah, Ningrum, P. A., Afrianti, Prasetia, I., Sari, N. I., Faina, & Annisa, N. (2021). Processing agricultural products to improve the economy of farmers in Bintan Regency. Jurnal Pengabdian dan Pemberdayaan Masyarakat Kepulauan Riau (JPPM Kepri), 1(2), 155-167. https://doi.org/10.35961/jppmkepri.v1i2.265

[7] Ratnawati, & Silma. (2017). Internet-of-Things-based plant watering control system using Propeller. Jurnal Inspiraton, 7(2), 143-154.

[8] Saragih, K. A., & Kurniawan, R. (2025). IoT-based automatic watering system using Sugeno fuzzy logic for soil moisture control in a greenhouse. Jurnal Algoritma, 22(1), 808-819. https://doi.org/10.33364/algoritma/v.23-1.2327

[9] Siddiq, M. H., Susanti, E., & Gunoto, P. (2023). Design of an Internet-of-Things-based vehicle speed measuring device. Sigma Teknika, 6(1), 117-126.

[10] Sunandar, & Herlawat. (2020). Prototype of an e-mail-based attendance machine at SMP Assyairiyah Attahiriyah Jakarta. Journal of Studens Research in Computer Scince, 1(2), 179-190.

[11] Putri, M. P., Barovih, G., Azdy, R. A., Yuniansyah, Saputra, A., Sriyeni, Y., Rini, A., & Admojo, F. T. (2022). Algorithms and data structures (1st ed.). Widina Bhakti Persada Bandung (Widina Media Utama Group).

[12] Ulinuha, A., & Riza, A. G. (2021). Android-based automatic plant monitoring and watering system using the Blynk application. Abdi Teknoyasa, 26-31. http://journals2.ums.ac.id/index.php/abditeknoyasa/

[13] Wiranto, Musyrifah, & Mansyur, M. F. (2024). Internet-of-Things-based automatic watering and fertilization system with Android application interface monitoring. Jurnal Sistem Informasi dan Informatika (Simika), 7(1), 28-37.

[14] Yahwe, C. P., Isnawaty, & Aksara, L. F. (2016). Development of an SMS-based prototype system for monitoring soil moisture based on plant watering results: A case study of chili and tomato plants. SemanTIK, 2(1), 97-110.

[15] Yanti, D. S., Azis, M. F., & Sidehabi, S. W. (2021). Prototype for automatic plant watering and fertilization with an Internet-of-Things-based monitoring system. Proceedings of the National Seminar on Industrial Technology (SNTI), 8(1), 18-22.