Science, Engineering and Technology Vol. 7, No. 1, pp. 14-29
www.setjournal.com https://doi.org/10.54327/set2026/v7.i1.373

Automatic Plant Watering Based on Soil Moisture Sensor and PID Control

Widyasmoro Widyasmoro[1], Iswanto Bin Suwarno[2], Sulistyo Nugroho[1], Nia Maharani Raharja[3], Muhammad Hafid Alkarim[2,4]

[1]*Department of Electrical Engineering, Universitas Muhammadiyah Yogyakarta, Yogyakarta, Indonesia*
[2]*Department of Informatics, Universitas Sarjanawiyata Tamansiswa, Yogyakarta, Indonesia*
[3]*Department of Informatics, Universitas Islam Negeri Sunan Kalijaga, Yogyakarta, Sleman, Indonesia*
[4]*Department of Research and Development, Peneliti Teknologi Teknik Indonesia, Sleman, 55281, Indonesia*

Abstract

This study presents the design and simulation of an automatic plant watering system based on a soil moisture sensor and Proportional-Integral-Derivative (PID) control, implemented on an Arduino Uno in the Tinkercad circuit simulator and analysed numerically in Scilab. The objective is to provide a documented and reproducible software-in-the-loop baseline for closed-loop soil moisture regulation. The simulated plant couples a first-order direct current (DC) pump/motor model with a first-order soil water-balance model that includes irrigation input, evaporation, and drainage; the PID output is converted into an 8-bit duty-cycle command, which the Arduino firmware realises by relay ON/OFF switching and the Scilab model applies as a continuous average pump drive. With the baseline gains Kp = 5.0, Ki = 0.8, and Kd = 0.1, the response from 30% to a 60% setpoint reached 90% of the step in 4.13 s, peaked at 77.43% (an overshoot of 29.06% of the setpoint), settled within ±2% after 42.83 s, and ended 0.10 percentage points from the setpoint at 60 s. In three further scenarios (20% to 50%, 40% to 70%, and a ±1% sinusoidal sensor disturbance), the overshoot ranged from 24.21% to 35.95% and the settling time from 39.73 s to 46.95 s. Under identical model conditions, a hysteresis ON/OFF controller overshot by 38.40% and sustained a limit cycle between 58.90% and 67.48%; PID reduced the overshoot by 24.3% and the integral absolute error by 20.6% and removed the persistent offset, at the cost of a 13.2% longer rise time. A gain-sensitivity analysis showed that the overshoot is caused mainly by integral action and can be eliminated with a smaller integral gain. All results are obtained from simulation without hardware validation.

Keywords*:* automatic irrigation, soil moisture control, PID control, ON/OFF control, Arduino, Tinkercad, Scilab

1. Introduction

The global water crisis places agricultural irrigation efficiency at the top of the research agenda [1], [2]. The mismatch between conventional time-based irrigation methods and in-situ soil moisture dynamics results in two antagonistic consequences: water wastage and plant water stress [3], [4], [5]. Smart irrigation systems that integrate real-time sensor data acquisition have emerged as a solution paradigm to balance irrigation precision and crop yield optimization [6], [7]. However, the effectiveness of these systems fundamentally depends on sensor accuracy and, more crucially, on the control strategy that translates soil moisture data into adaptive actuator actions [8], [9], [10].

Despite this progress, most reported smart-irrigation systems either rely on simple threshold-based ON/OFF

Corresponding author: Iswanto Bin Suwarno (iswantobinsuwarno@gmail.com)
Received: 15 May 2026; Revised: 1 October 2026; Accepted: 7 October 2026; Published: 13 November 2026
© 2026 The Author(s). This work is licensed under a Creative Commons Attribution 4.0 International License

actuation or apply more advanced feedback strategies, such as Kalman-filter-enhanced Proportional-Integral-Derivative (PID) control or model predictive control (MPC), that have so far been validated mainly on physical hardware testbeds or in field trials. Comparatively little publicly documented guidance exists on how to design, tune, and pre-validate a PID-based closed-loop irrigation controller entirely within a virtual simulation environment before committing to hardware fabrication. This gap is particularly relevant for low-cost, Arduino-based irrigation platforms commonly used in small-scale and educational agricultural applications, where iterative hardware prototyping is costly and time-consuming, and where documented PID tuning rationale for the nonlinear, disturbance-prone dynamics of soil-moisture regulation remains scarce.

Addressing this gap, the present study investigates whether a fully virtual, software-in-the-loop (SIL) workflow, combining Tinkercad circuit simulation with Scilab numerical analysis, can support the design, tuning, and pre-validation of a PID-based automatic plant watering system with quantifiable transient performance prior to physical implementation. PID control is adopted here because, relative to conventional ON/OFF actuation, it offers continuous error correction, oscillation damping, and steady-state error elimination, while remaining computationally lightweight enough for an entry-level microcontroller such as the Arduino Uno. The specific research question addressed is whether a PID controller, tuned by trial and error from a Ziegler-Nichols starting point and evaluated entirely within a combined Tinkercad-Scilab simulation environment, can provide stable soil moisture regulation that improves on conventional ON/OFF control, and what its quantified limitations are when it is used as a documented, reproducible baseline for future comparison with more advanced control strategies.

Building on this motivation, this study makes three contributions, detailed further in the Research Objectives and Contributions section: (1) a structured, virtually verified PID-based closed-loop irrigation simulation framework with explicitly documented PID parameters and tuning rationale; (2) a quantitative characterization of the PID transient response (rise time, overshoot, settling time, final error at 60 s, and integral absolute error (IAE)) for an explicit pump–soil model under four operating scenarios; and (3) a like-for-like comparison with hysteresis ON/OFF control and a gain-sensitivity analysis, which together form a reproducible baseline against which future work can benchmark proportional-integral (PI), fuzzy, and MPC-based irrigation controllers. A detailed review of related soil-moisture sensing and irrigation-control literature, situating these contributions relative to prior work, is presented in the following Literature Review section.

2. Literature Review

2.1. Soil moisture sensing technologies

The advancement of soil moisture sensor technology has achieved high diversification, including microwave planar-based measurements for multi-parameter detection [11], capacitive sensors printed on biodegradable substrates [12], to high-precision plastic fiber optic sensors in various soil types [13]. These studies demonstrate a collective effort to address measurement variability—both stemming from inter-sensor error [14] as well as seasonal biases in satellite products through machine learning-based data fusion [15] and specific calibration for extreme conditions such as saline irrigation [16]. Despite progress in spatial optimization of sensor placement [17] and full soil profile measurement accuracy [18], the research focus is still predominantly on sensor signal acquisition and processing [19], [20], [21], [22] and has not touched much on feedback control strategies that integrate sensor data with irrigation actuation in a closed loop.

2.2. Irrigation control strategies: ON/OFF versus PID control

In the realm of implementation, the majority of existing sensor-based irrigation systems rely on simple ON/OFF control [23], [24]. This approach inherently produces soil moisture oscillations around the setpoint that induce hysteresis cycles, forcing the actuator to operate discontinuously and potentially shortening component life and disrupting rhizosphere moisture stability [11], [24]. PID control offers a continuous control mechanism capable of dampening oscillations, eliminating steady-state errors, and anticipating moisture deviations through derivative action [25], [26]. Beyond

irrigation, PID control and its enhanced variants are well established for regulating dynamic processes, including robotic trajectory tracking [27], tunnel ventilation and smoke control [28], [29], electro-hydraulic actuation [30], anaesthesia drug delivery, and grid-connected photovoltaic inverters [31], [32]. In irrigation specifically, a Kalman-filter-enhanced PID controller [33] and model-based predictive control [34] have demonstrated the effectiveness of feedback-based irrigation management, illustrating the suitability of feedback strategies for regulating soil moisture.

2.3. Research gap and motivation

Although various studies have applied feedback control in the precision irrigation domain [33], [34], [35], [36], [37], a critical research gap remains. None of these studies have formulated and validated a fully integrated PID-based closed-loop irrigation system simulation model from the sensor layer to the actuator within a single virtual development environment such as Tinkercad coupled with Scilab numerical analysis. Furthermore, systematic PID tuning procedures have been developed mainly for electromechanical systems such as direct current (DC) motors [38], whereas tuning for the slow, loss-dominated dynamics of soil moisture has not been systematically characterized in the irrigation context. This lack of documented simulation frameworks hampers the transition from control design concepts towards reproducible hardware prototypes for specific planting applications. The present work addresses this gap by providing a structured virtual prototype with documented PID parameters and tuning rationale.

To situate the present work relative to closely related irrigation control studies, Table 1 summarizes the control strategy, platform/environment, validation approach, and principal difference from the present study for representative references.

3. Research Objectives and Contributions

This research proposes the design and simulation of an automatic plant watering system based on a soil moisture sensor with PID control using Arduino on the Tinkercad platform. Our approach integrates models of the moisture sensor, relay-switched pump actuator, and digital PID controller in a single virtual prototyping architecture. This allows thorough pre-validation of the control logic and prediction of system performance metrics such as settling time, maximum overshoot, and final error - before entering the physical fabrication stage.

The adopted methodology is a software-in-the-loop (SIL) simulation approach, utilizing Tinkercad's circuit simulation to verify the wiring and the embedded PID control procedure (Algorithm 1) and Scilab to analyze the closed-loop response numerically [39], [40], [41], [42]. It should be noted that Tinkercad does not involve physical hardware; therefore, the approach is classified as SIL rather than hardware-in-the-loop (HIL). The relay module used in this study provides ON/OFF pump switching; pulse-width modulation (PWM) is approximated at the software level through duty-cycle control rather than by continuous analog PWM hardware.

The main contributions of this research are threefold. First, it provides a structured and virtually verified PID-based closed-loop irrigation system simulation framework using Tinkercad and Scilab, with explicitly documented PID parameter values ($K_p = 5.0$, $K_i = 0.8$, $K_d = 0.1$) and tuning rationale, enabling reproducibility.

Second, it provides a quantitative analysis of PID transient response characteristics (rise time, overshoot, settling time, final error at 60 s, and IAE) using an explicit two-stage plant model in which a first-order actuator (DC motor/pump) response model (Eq. (3)) drives a first-order soil water-balance model with irrigation, evaporation, and drainage terms (Eq. (4)), evaluated under four operating scenarios and against a hysteresis ON/OFF controller under identical conditions, offering a reference baseline that prior irrigation-focused PID studies have not explicitly reported in a combined simulation environment.

Third, this framework serves as a reproducible foundation for future research comparing PID against advanced strategies (fuzzy, MPC, adaptive PID) and for transitioning validated control logic to physical Arduino-based hardware prototypes [43], [44], [45], [46], [47].

Table 1. Comparison of the present study with representative related irrigation control studies.

Study	Control Strategy	Platform / Environment	Validation / Hardware Status	Difference from This Study
Hamouda et al. [23]	Threshold-based ON/OFF	Field-deployed wireless sensor network, pear orchard	Real field trial	Simple ON/OFF hardware in a real orchard; this study offers a simulated, tunable PID controller with documented transient performance. Tuning method: not applicable (threshold rule, no gain tuning). Reported metric: field-scale soil-moisture and water-use trends (see original publication for values).
Sabah Abbas et al. [24]	Automatic sensing-based ON/OFF scheduling	Physical hardware prototype	Bench/hardware testing	Physical ON/OFF device without PID or a virtual pre-validation stage; this study emphasizes a reproducible SIL prototyping workflow before hardware. Tuning method: not applicable (threshold rule, no gain tuning). Reported metric: device response time and water-use figures from bench/hardware tests (see original publication for values).
Abioye et al. [33]	Kalman filter-enhanced PID	Fibrous capillary irrigation testbed	Experimental (physical)	Adds state estimation and a physical testbed; this study is virtual-only but documents a full Tinkercad + Scilab tuning workflow. Tuning method: Kalman-filter-informed PID gains. Reported metric: irrigation water savings and moisture-tracking accuracy (see original publication for values).
Abioye et al. [34]	Model Predictive Control (MPC)	Drip irrigation model	Simulation	Uses MPC (more complex, model-dependent); this study benchmarks a simpler, lower-cost PID baseline suited to embedded Arduino platforms. Tuning method: model-based optimization (no manual gain tuning). Reported metric: water savings vs. conventional drip scheduling (see original publication for values).
Bwambale et al. [35]	Data-driven moisture modeling / scheduling	Data-driven model, field data	Field-data-driven	Focuses on moisture modeling and scheduling logic rather than an embedded feedback loop; this study implements and tunes an explicit PID controller. Tuning method: data-driven/machine-learning calibration (not a PID gain-tuning method). Reported metric: moisture-prediction accuracy (see original publication for values).
Chakravarty & Roy [36]	Quantitative Feedback Theory (QFT), multivariable robust control	Simulation (process control)	Simulation	Robust multivariable control theory, not embedded/Arduino-oriented; this study targets a low-cost, reproducible embedded implementation. Tuning method: QFT loop-shaping. Reported metric: robustness margins and disturbance-rejection performance (see original publication for values).
Ghiat et al. [37]	Data-driven MPC	Greenhouse simulation (CO_2 enrichment)	Simulation	Greenhouse-specific MPC; this study targets general soil-moisture PID control with documented trial-and-error tuning from a Ziegler–Nichols starting point. Tuning method (Ghiat et al.): model-based optimization (no manual gain tuning). Reported metric: irrigation efficiency and climate-control performance (see original publication for values).
This study	PID (trial-and-error tuning from a Ziegler–Nichols starting point)	Tinkercad (SIL) + Scilab pump–soil model	Simulation (four scenarios; ON/OFF baseline under identical conditions)	Reproducible, fully documented virtual-prototyping framework from sensor to actuator; baseline for future PI/fuzzy/MPC comparison. Tuning method: trial and error from a Ziegler–Nichols starting point (baseline Kp = 5.0, Ki = 0.8, Kd = 0.1; sensitivity in Table 3). Reported metrics (scenario S1): rise time 4.13 s, overshoot 29.06%, settling time 42.83 s, final error at 60 s 0.10 percentage points (pp); PID versus ON/OFF: overshoot −24.3%, IAE −20.6% (Tables 4–6).

4. Research Methods

4.1. Research design

This study uses a simulation-based experimental method with the aim of designing and testing an automatic plant watering system that utilizes a soil moisture sensor and PID control. The entire system is designed virtually using the Tinkercad platform, allowing testing without the need for physical hardware. The main components simulated include an Arduino Uno as the main controller, a soil moisture sensor as a source of feedback data, a relay module as a link between the control system and the actuator, and a DC motor that represents the water pump. The PID control procedure (Algorithm 1) is programmed through the Arduino IDE (Integrated Development Environment) to regulate the pump's operation

proportionally based on the measured soil moisture value. As summarised in the research flowchart in Figure 1, this study was carried out through the stages of system design, PID implementation, and test simulation, with a focus on analyzing the stability of soil moisture and the watering efficiency produced by the system.

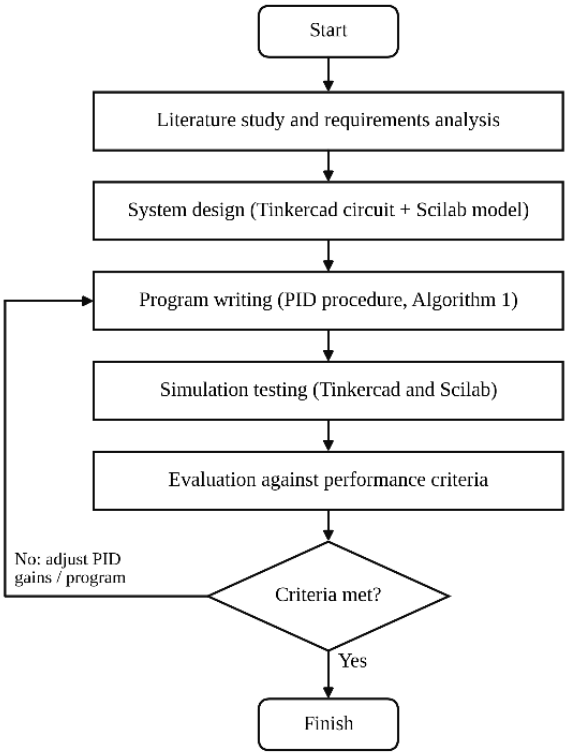


Figure 1. Research flowchart**.**

4.2. Research stages

The research phase began with a literature study to gain a deeper understanding of the automatic watering system, the characteristics of soil moisture sensors, the working principles of PID control, as well as the use of Arduino and circuit simulation on the Tinkercad platform [48]. The results of the literature study became the basis for conceptual system design, which was outlined in the form of a research flowchart and a system block diagram. Next, a circuit design was carried out, including the selection and placement of the main components, namely the Arduino Uno, soil moisture sensor, relay module, and DC motor as a water pump, a configuration commonly used in Arduino-based irrigation prototypes [49], [50]. After the design was determined, the system was built virtually using Tinkercad Circuits to enable initial testing. The next stage was programming the Arduino with the PID control procedure (Algorithm 1), which regulates the relay ON-time within each control window according to the difference between the actual soil moisture value and the predetermined setpoint value.

4.3. Needs analysis

A system requirements analysis was conducted to ensure that all components are capable of optimally supporting the automatic watering simulation process. This system requires an Arduino Uno as a control center [51], a soil moisture sensor as an input device, a DC motor as a water pump representation, a relay module as a power connector, and a light-emitting diode (LED) indicator to display system status. All components are simulated in the Tinkercad environment and controlled through a program developed using the Arduino IDE. The system is designed to work automatically with a closed-loop control principle, where soil moisture data is used as feedback to determine the watering response, so that water use efficiency and system stability can be analyzed systematically.

4.4. Components

The Arduino Uno functions as the main microcontroller that processes data from the soil moisture sensor and generates control signals to regulate the actuator. The soil moisture sensor is used to detect the water content in the growing medium and sends analog values to the Arduino as a basis for decision making. The relay module acts as an electronic switch that allows the Arduino to safely control the DC motor without overloading the control circuit. The DC motor is used as a representation of a water pump that flows water to the plants when the soil moisture is below the set threshold value. In addition, the system is also equipped with a power supply unit as a source of electricity and LEDs as visual indicators that show system conditions, such as active watering or normal (idle) operation.

4.5. System design

The automatic watering system was designed using the Tinkercad platform for electronic circuit simulation and

Scilab software for numerical analysis of the closed-loop PID response. Figure 2 shows the Tinkercad circuit, in which all components are assembled virtually according to the designed schematic. The soil moisture sensor is connected to an analog input of the Arduino, the relay module to digital pin 2, and the LED indicator (with its series resistor) to digital pin 3; the relay switches the DC motor, which is supplied by a separate 5 V source. The Arduino program (Algorithm 1) was written directly in the Tinkercad code editor to read the soil moisture value, compute the PID control output, and switch the pump through the relay. Because the Tinkercad soil moisture sensor is set manually through an on-screen slider and has no soil-water dynamics, the Tinkercad stage verifies the wiring and control logic through the Serial Monitor, whereas the quantitative closed-loop response is obtained from the Scilab model described below.

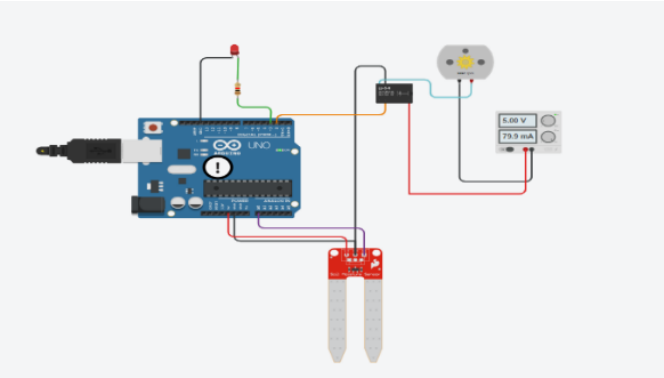


Figure 2. Automatic watering circuit in Tinkercad (Arduino Uno, soil moisture sensor, relay module, DC motor, LED indicator, and external 5 V supply).

Figure 3 shows the closed-loop block diagram of the Scilab simulation used to evaluate the PID controller before its application to the automatic plant watering system. The measured soil moisture is compared with the setpoint to generate an error signal, which the PID controller converts into a duty-cycle command; this command drives the pump/motor model, whose output supplies water to the soil water-balance model. The individual elements are defined in Eqs. (1)–(5).

In the comparison block, the error signal is obtained from the difference between the reference value and the system output, which can be mathematically expressed as:

$$e(t) = r(t) - y(t) \tag{1}$$

where $r(t) = \theta_{sp}$ is the soil moisture setpoint (%), $y(t) = \theta(t)$ is the measured soil moisture (%), and e(t) is the control error in percentage points (pp).

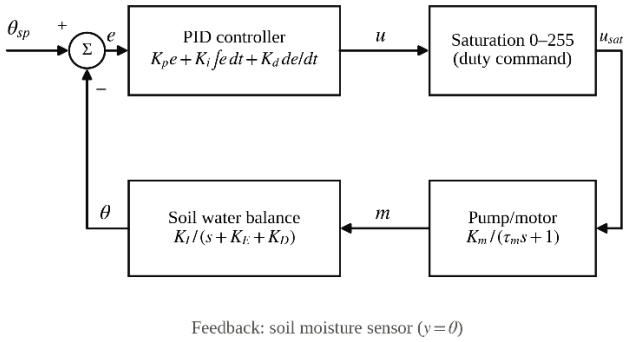


Figure 3. Closed-loop block diagram of the Scilab simulation model: PID controller, 0–255 command saturation, first-order pump/motor, and first-order soil water balance.

This error signal then becomes the input to the PID controller. PID control consists of three main components: proportional, integral, and derivative action, which respectively accelerate the system response, eliminate the steady-state error, and dampen oscillations. The general equation for PID control in the time domain can be written as:

$$u(t) = K_p\, e(t) + K_i \int e(t)\, dt + K_d \frac{de(t)}{dt} \tag{2}$$

where u(t) is the PID control signal, K_p is the proportional gain, K_i is the integral gain (s^{-1}), and K_d is the derivative gain (s). In the discrete implementation (integration step 0.01 s in Scilab and sampling interval 100 ms in the Arduino firmware), the integral term is limited to ±100 (anti-windup) and u(t) is saturated to the 8-bit range 0–255, giving the duty-cycle command $u_{sat}(t)$.

The saturated command drives a DC motor model representing the water pump actuator. The motor is modelled as a first-order dynamic system whose output m(t) represents the pump delivery (a flow-proportional drive level). In the Laplace domain, the motor model can be expressed as:

$$G_m(s) = \frac{K_m}{\tau_m s + 1} \tag{3}$$

with K_m = 1.20 as the motor gain and τ_m = 2.50 s as the motor time constant. The pump delivery m(t) is the input

of an explicit first-order soil water-balance model, in which irrigation raises the soil moisture θ(t) while evaporation and drainage remove water in proportion to the current moisture level:

$$d\theta(t)/dt = K_I m(t) - (K_E + K_D)\theta(t) \quad (4)$$

where K_I = 0.080 s^{-1} is the irrigation (infiltration) coefficient, K_E = 0.015 s^{-1} is the evaporation coefficient, and K_D = 0.005 s^{-1} is the drainage coefficient, with θ bounded to 0–100%. The measured soil moisture y(t) = θ(t) is fed back to the comparison block, closing the loop. In the Laplace domain, Eq. (4) corresponds to Θ(s)/M(s) = K_I/(s + K_E + K_D), so the open-loop plant is a cascade of two first-order lags (Figure 3). At steady state, Eq. (4) requires $K_I m = (K_E + K_D)\theta_{sp}$, which, combined with the unit-free static gain of Eq. (3), gives the steady-state command

$$u_{ss} = \frac{(K_E + K_D)\theta_{sp}}{K_I K_m} \quad (5)$$

which equals 12.5 (a 4.9% duty cycle) for θ_{sp} = 60%; this analytical value provides a consistency check on the simulated control signal. All model and controller parameters are listed in Table 2.

Table 2. Model, controller, and simulation parameters.

Symbol	Description	Value
K_p, K_i, K_d	PID gains (baseline)	5.0, 0.8 s^{-1}, 0.1 s
—	Integral (anti-windup) limit	±100
u_{sat}	Duty-cycle command range	0–255
K_m, τ_m	Pump/motor gain and time constant	1.20, 2.50 s
K_I	Irrigation (infiltration) coefficient	0.080 s^{-1}
K_E, K_D	Evaporation and drainage coefficients	0.015, 0.005 s^{-1}
Δt, T	Integration step and simulation horizon (Scilab)	0.01 s, 60 s
T_s, T_w	Firmware sampling interval and relay window	100 ms, 2 s
—	ON/OFF baseline: hysteresis band and ON command	±1 pp, 180/255

The pump and soil coefficients are nominal, illustrative values that have not been identified from measurements of a specific pump or soil; the time axis should therefore be interpreted relative to these nominal dynamics rather than as a field-scale irrigation time.

The closed-loop equations were integrated with the forward-Euler method (step 0.01 s, horizon 60 s) in the Scilab scripts listed in the Data Availability Statement. Performance was quantified by the rise time t_r (time to reach 90% of the step from the initial value to the setpoint), the peak value and overshoot (peak excess above the setpoint expressed as a percentage of the setpoint), the settling time t_s (time after which θ remains within ±2% of the setpoint), the final error at 60 s, e_{60} (absolute deviation from the setpoint at t = 60 s, in pp; this term is used instead of steady-state error because not every response has fully converged within the 60 s horizon), the integral absolute error (IAE, the integral of |e(t)| over 0–60 s, in %·s), and the number of pump switching events (transitions of the simulated pump-active state, defined as a continuous command above 10/255; relay pulses within the 2 s firmware windows are not counted). These metrics allow candidate PID parameters to be compared before implementation in the Arduino firmware. Simulated results are reported with two decimal places for soil moisture, errors, overshoot, times and σ_u, and with one decimal place for IAE, command levels and relative changes.

4.6. PID parameter tuning

An initial gain set was computed with the classical Ziegler–Nichols (ZN) closed-loop rules ($K_p = 0.6K_u$, $T_i = P_u/2$, $T_d = P_u/8$) using an assumed ultimate gain K_u = 8.30 and ultimate period P_u = 12.0 s, giving K_p = 4.98, K_i = 0.83 s^{-1}, and K_d = 7.47 s. Because the linearised plant (two first-order lags without dead time) has no finite ultimate gain, these values served only as a heuristic starting point.

Candidate gain sets were then refined by trial and error, and every candidate was simulated with the same model, both without and with a ±1% sensor disturbance (Table 3). The gains K_p = 5.0, K_i = 0.8, and K_d = 0.1 (Trial 4) were adopted as the baseline implemented in the firmware; the trade-offs revealed by Table 3 are discussed in the Results and Discussion section.

Table 3. Simulated performance of the candidate PID gain sets (scenario S1: 30% to 60%). t_r = rise time; OS = overshoot; t_s = settling time; e_{60} = final error at 60 s; IAE = integral absolute error; Switch. = pump switching events. Switching events and the standard deviation of the command, σ_u ($t \geq 45$ s), are for the ±1% sensor-disturbance case; all other metrics are for the undisturbed case.

Candidate (K_p / K_i / K_d)	t_r (s)	OS (%)	t_s (s)	e_{60} (pp)	IAE (%·s)	Switch.	σ_u
ZN start (4.98 / 0.83 / 7.47)	4.98	18.72	33.16	0.02	239.0	72	14.18
Trial 1 (5.00 / 0.83 / 1.00)	4.21	27.69	42.47	0.04	403.4	34	6.12
Trial 2 (5.00 / 0.80 / 0.50)	4.18	28.09	42.56	0.07	405.6	29	5.36
Trial 3 (5.00 / 0.80 / 0.20)	4.15	28.81	42.76	0.09	413.2	23	5.35
Trial 4, baseline (5.00 / 0.80 / 0.10)	4.13	29.06	42.83	0.10	415.9	23	5.40
Trial 5 (4.80 / 0.75 / 0.10)	4.25	28.02	42.73	0.03	407.5	23	5.01
Trial 6, low K_i (2.00 / 0.10 / 0.10)	10.61	0.00	12.46	1.13	227.1	23	1.47

4.7. Program writing

The Arduino program (repository file arduino.ino) reads the soil moisture sensor, computes the PID output, and drives the relay; its control procedure is summarised in Algorithm 1.

Before being run in Tinkercad, the code was compiled and verified in the Arduino IDE to ensure that it was free of syntax and logic errors. The sensor reading (0–1023) is mapped linearly to 0–100% soil moisture between the dry and wet calibration values, and the PID output is limited to 0–255. Because the relay can only switch the pump fully ON or OFF, the command is realised as a software duty cycle: within each control window of length $T_w = 2$ s, the relay is energised for

$$t_{on} = \frac{u_{sat}}{255} T_w \quad (6)$$

An additional interlock keeps the relay OFF whenever the measured moisture is at or above the setpoint. In Scilab, microcontroller programming is not required; the same PID law, anti-windup limit, and output saturation is evaluated directly in the simulation scripts, whereas the relay windowing of Eq. (6) and the interlock are not modelled (see the limitations at the end of Section 5).

Algorithm 1. PID control procedure implemented in the Arduino firmware (sampling interval $T_s = 100$ ms).
Procedure

Input: setpoint $\theta_{sp} = 60\%$; gains K_p, K_i, K_d; window $T_w = 2$ s
Initialise: integral $I \leftarrow 0$; previous error $e_{prev} \leftarrow 0$
1: repeat every $T_s = 100$ ms
2: read analog-to-digital converter (ADC) value (0–1023) and map it to θ (0–100%)
3: $e \leftarrow \theta_{sp} - \theta$
4: $I \leftarrow \text{clamp}(I + e \cdot \Delta t, -100, 100)$ (anti-windup)
5: $D \leftarrow (e - e_{prev}) / \Delta t$
6: $u \leftarrow \text{clamp}(K_p \cdot e + K_i \cdot I + K_d \cdot D, 0, 255)$
7: $t_{on} \leftarrow (u / 255) \cdot T_w$ (Eq. (6))
8: if (elapsed time in window $< t_{on}$) and ($\theta < \theta_{sp}$) then relay ON, LED ON
9: else relay OFF, LED OFF
10: $e_{prev} \leftarrow e$; print θ, e, u to the Serial Monitor
11: until simulation is stopped

4.8. Test simulation

The simulation testing phase was carried out after the program had been verified. The verified program was entered into the Tinkercad code editor, and the simulation was run with the Start Simulation button. At this stage, the sensor slider was varied to emulate dry and wet soil, and the responses of the relay, the DC motor, and the LED indicator were observed. In Scilab, the closed-loop response was evaluated with the metrics defined above for four operating scenarios and was compared with a hysteresis ON/OFF controller under identical conditions. The ON/OFF baseline switches the pump fully ON (command 180/255) when $\theta \leq \theta_{sp} - 1$ pp and OFF when $\theta \geq \theta_{sp} + 1$ pp, otherwise holding its previous state, and uses exactly the same pump and soil models, initial condition, and integration step as the PID simulation.

4.9. Evaluation and improvement

The evaluation and refinement phase aims to identify weaknesses or problems that arise during the system

simulation process. The evaluation is conducted by comparing the system's response results against predetermined performance criteria, such as output stability, response speed, and accuracy of achieving the soil moisture setpoint. Based on the evaluation results, adjustments to the PID control parameters and improvements to the program logic are made to improve overall system performance, thereby making the designed automatic watering system more stable, efficient, and reliable under simulated conditions.

5. Results and Discussion

System testing was conducted in two complementary stages: the Tinkercad stage verified the wiring and firmware logic, and the Scilab stage quantified the closed-loop response. Four operating scenarios were simulated with the baseline gains (K_p = 5.0, K_i = 0.8, K_d = 0.1): S1, the baseline dry-to-moist transition from 30% to a 60% setpoint; S2, from 20% to 50%; S3, from 40% to 70%; and S4, the S1 transition with a ±1% sinusoidal (0.8 Hz) disturbance added to the moisture measurement to emulate sensor noise. The PID controller was then compared with the hysteresis ON/OFF controller for S1 under identical model conditions.

Testing in Tinkercad focused on verifying the functionality of the virtual hardware, including the Arduino Uno, soil moisture sensor, relay module, DC motor, and LED indicator. The Arduino receives the analog sensor value, computes the PID output according to Algorithm 1, and switches the pump through the relay. When the sensor slider was set below the setpoint, the relay and LED were energised for the computed ON-time in each window; when it was set at or above the setpoint, they remained de-energised, confirming that the firmware logic operates as designed. Scilab was then used to quantify the closed-loop behaviour of the same control law.

The system's control procedure is based on the principle of feedback control, where the soil moisture value read by the sensor is compared to a predetermined setpoint. The difference between the two values is used as an error signal, which forms the basis for calculating the PID control output. This control output then determines the condition of the actuator, namely the water pump, that is, the fraction of each control window during which the pump runs.

Figure 4 shows the soil moisture response for the baseline scenario S1, and Table 4 summarises the corresponding performance metrics.

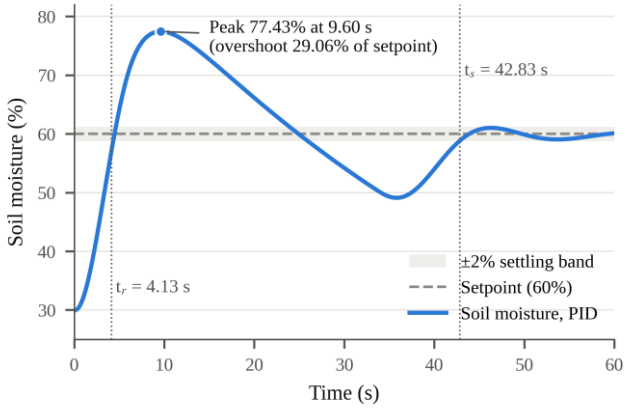


Figure 4. Soil moisture response with PID control for scenario S1 (30% to 60%); the shaded band marks ±2% of the setpoint.

Table 4. PID control performance for scenario S1 (K_p = 5.0, K_i = 0.8, K_d = 0.1).

Performance parameter	Value	Description
Soil moisture setpoint	60%	Target value
Initial soil moisture	30%	Dry initial condition
Rise time t_r	4.13 s	Time to reach 57% (90% of the 30 pp step)
Peak value	77.43% at 9.60 s	Maximum soil moisture
Overshoot	29.06%	(Peak − setpoint)/setpoint × 100
Settling time t_s	42.83 s	θ remains within 58.8–61.2%
Final moisture	60.10%	Value at t = 60 s
Final error at 60 s, e_{60}	0.10 pp	|Setpoint − value at 60 s|
IAE	415.9 %·s	Integral of |e(t)| over 0–60 s
Pump switching events	5	Transitions of the simulated pump-active state (command > 10/255)
System status	Stable, underdamped	Converges without sustained oscillation

Starting from 30%, the large initial error produces a command of 150.2/255, and the moisture rises rapidly, reaching 90% of the step (57%) after 4.13 s and crossing the setpoint at 4.49 s. Because the pump/motor lag (τ_m = 2.50 s) delays the reduction of water delivery and the accumulated integral term keeps the command high

during the rise, the moisture continues to increase to a peak of 77.43% at 9.60 s, an overshoot of 29.06% of the setpoint.

After the peak, the command remains clipped at zero (Figure 5) while evaporation and drainage slowly reduce the moisture, which undershoots to 49.10% at 35.82 s before the integral term re-activates the pump. The response then enters the ±2% band at 42.83 s and ends 0.10 pp above the setpoint at 60 s (Table 4). The response is therefore stable but underdamped: the controller removes the steady-state offset, but the transient is dominated by one large overshoot–undershoot cycle.

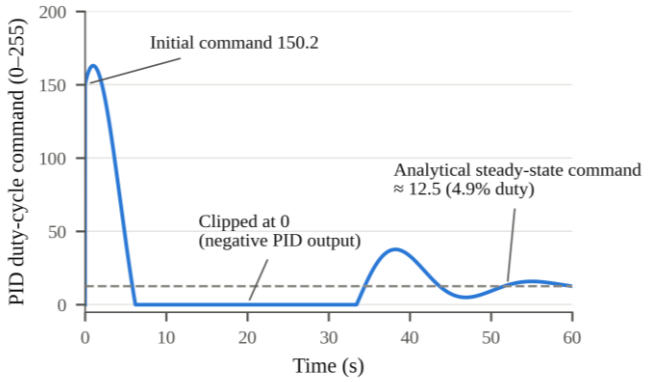


Figure 5. Continuous PID duty-cycle command $u_{sat}(t)$ computed in the Scilab simulation for scenario S1 (0–255 scale); the relay windowing of Eq. (6) is not modelled.

The continuous PID command computed in the Scilab simulation is shown in Figure 5. At the start of the simulation, the 30 pp error produces a command of 150.2/255 (a 58.9% duty cycle), which rises to a maximum of 163.0/255 at 1.00 s as the integral term accumulates. After the moisture exceeds the setpoint, the negative proportional term outweighs the accumulated integral term, and the command is clipped to zero from 6.21 s to 33.43 s; that is, the pump remains OFF while the excess water is removed by evaporation and drainage. The command then recovers and converges towards the analytical steady-state value of 12.5/255 given by Eq. (5), with a mean of 14.0/255 over the last 10 s of the simulation.

To avoid ambiguity, two levels of actuation are distinguished. In the Arduino firmware (Algorithm 1), the PID output u(t) is limited to an 8-bit duty-cycle command u_{sat} in the range 0–255 (reported here as a "PWM" value, following common Arduino terminology), and the relay module, which can only switch the DC motor circuit ON or OFF, realises this command as one ON pulse of duration t_{on} within each 2 s window (Eq. (6)), subject to the interlock; this behaviour was verified only qualitatively in Tinkercad. In the Scilab simulation, which produced Figures 4–7 and Tables 3–6, neither the relay windowing nor the interlock is modelled: $u_{sat}(t)$ is applied directly as a continuous average drive to the first-order motor model of Eq. (3), whose output feeds the soil water balance of Eq. (4). Accordingly, the command values reported here (150.2/255 at start-up and approximately 12.5/255 at steady state) and the command curves in Figures 5, 6b and 7b are this continuous simulated command, not relay pulses or a voltage or speed measured at the motor terminals, and the reported pump switching events count transitions of the simulated pump-active state rather than relay operations. Because the relay is switched ON and OFF within every 2 s window whenever $0 < u_{sat} < 255$ and $\theta < \theta_{sp}$, the number of relay operations in hardware would be considerably larger than the simulated switching counts.

The gain-sensitivity results in Table 3 clarify the origin of this behaviour. Trials 1–5, which vary K_d between 0.1 and 1.0 with only small changes in K_p and K_i, produce almost identical responses (overshoot 27.69–29.06%, settling time 42.47–42.83 s), so the small derivative gain of the baseline has little influence on the transient. The larger derivative gain of the Ziegler–Nichols starting point ($K_d = 7.47$) reduces the overshoot to 18.72% and the IAE by 42.5% (239.0 versus 415.9 %·s), but under the ±1% sensor disturbance it triples the number of pump switching events (72 versus 23) and increases the standard deviation of the command from 5.40 to 14.18, because the derivative term amplifies measurement noise. Reducing the integral gain has a much stronger effect: with $K_p = 2.0$, $K_i = 0.1$, and $K_d = 0.1$ (Trial 6), the overshoot is eliminated and the settling time falls to 12.46 s, at the cost of a slower rise (10.61 s) and a residual error of 1.13 pp at 60 s. The overshoot of the baseline controller is therefore caused mainly by integral accumulation during the rise, permitted by the relatively wide anti-windup limit (±100, i.e., up to 80/255 of command from the integral term), rather than by insufficient derivative action. The baseline gains were retained because they are the values implemented in the firmware and they tolerate sensor noise well; however,

Table 3 indicates that a lower integral gain or a tighter anti-windup limit should be preferred in the next design iteration.

Figure 6 and Table 5 summarise the responses for all four scenarios. The qualitative behaviour is the same in every case: a rise time of 4.08–4.19 s, one dominant overshoot followed by an undershoot, and settling within 39.73–46.95 s.

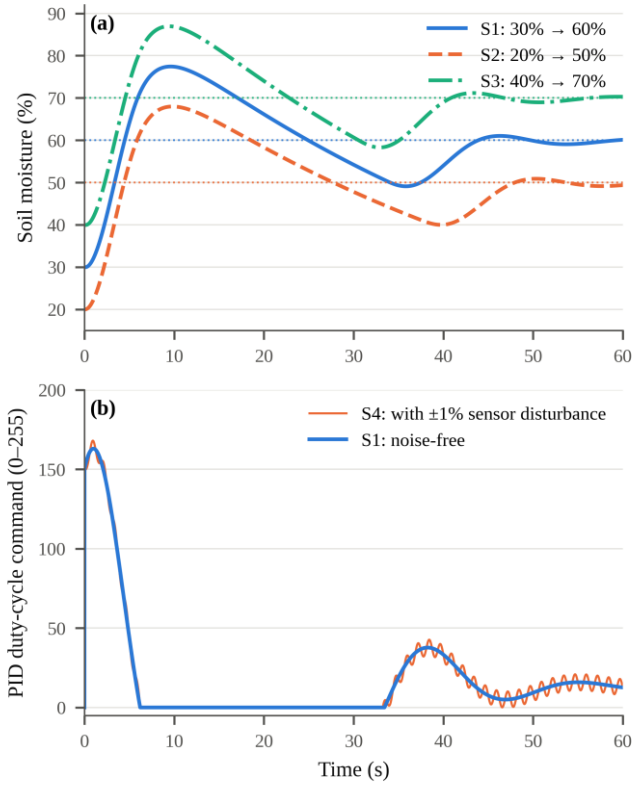


Figure 6. Multi-scenario validation of the baseline PID controller: (a) soil moisture responses for S1–S3 (dotted lines: setpoints); (b) simulated PID command for S1 without and S4 with a ±1% sensor disturbance.

Table 5. Performance of the baseline PID controller in the four operating scenarios (abbreviations as in Table 3).

Scenario	t_r (s)	OS (%)	t_s (s)	e_{60} (pp)	IAE (%·s)	Switch.
S1: 30% → 60%	4.13	29.06	42.83	0.10	415.9	5
S2: 20% → 50%	4.08	35.95	46.95	0.63	441.9	5
S3: 40% → 70%	4.19	24.21	39.73	0.28	397.2	5
S4: S1 + ±1% sensor disturbance	4.14	29.18	42.90	0.09	417.4	23

The absolute overshoot is similar in S1–S3 (16.95–17.97 pp above the respective setpoints); the differences in percentage overshoot (24.21% for S3 to 35.95% for S2) arise mainly from normalising by different setpoints. At 60 s, S2 and S3 remain 0.63 and 0.28 pp from their setpoints, indicating that their final convergence is not fully complete within the simulation horizon. The ±1% sensor disturbance in S4 changes the moisture metrics only marginally (overshoot 29.18% versus 29.06%; settling time 42.90 s versus 42.83 s), but it propagates to the command (Figure 6b) and increases the number of simulated pump switching events from 5 to 23; in hardware, this would translate into more frequent relay operation and faster relay wear. Figure 7 and Table 6 compare the PID controller with the hysteresis ON/OFF controller under identical conditions (scenario S1). The ON/OFF controller keeps the pump fully ON until the moisture reaches 61%, after which the pump/motor lag carries the moisture to a peak of 83.04% (an overshoot of 38.40%).

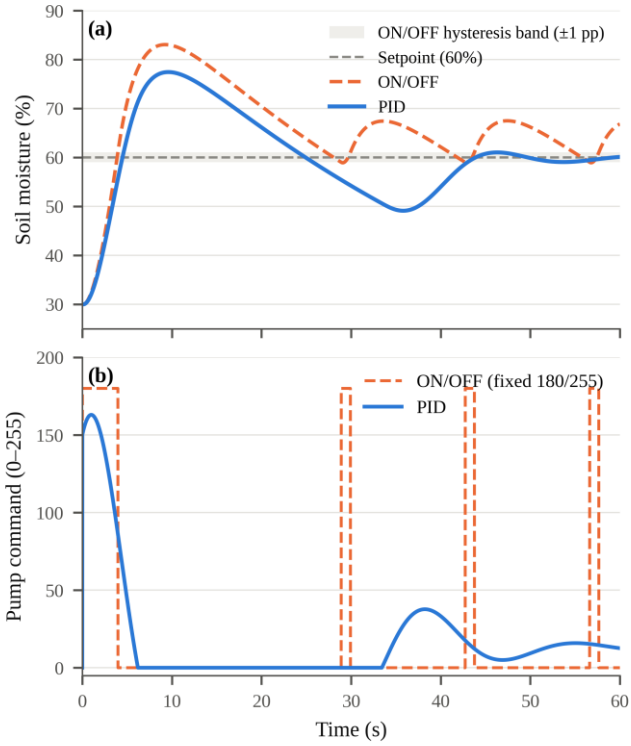


Figure 7. PID versus hysteresis ON/OFF control under identical conditions (S1: 30% to 60%): (a) soil moisture; (b) simulated pump command.

Table 6. PID versus hysteresis ON/OFF control under identical model conditions (S1).

Metric	PID	ON/OFF
Rise time t_r (s)	4.13	3.65
Peak soil moisture (%)	77.43	83.04
Overshoot (% of setpoint)	29.06	38.40
Settling time t_s, ±2% (s)	42.83	Not reached (limit cycle 58.90–67.48%)
Final moisture at 60 s (%)	60.10	66.82
Final error at 60 s, e_{60} (pp)	0.10	6.82
IAE (%·s)	415.9	523.6
Pump switching events	5	8

Thereafter, it sustains a limit cycle between 58.90% and 67.48% (mean 64.09%) and never settles within ±2% of the setpoint, ending 6.82 pp above it at 60 s. PID reduces the overshoot by 24.3% (29.06% versus 38.40%) and the IAE by 20.6% (415.9 versus 523.6 %·s), removes the sustained oscillation and the persistent offset, and requires fewer pump switching events (5 versus 8), whereas its rise time is 13.2% longer (4.13 s versus 3.65 s) because the initial PID command (150.2/255) is lower than the fixed ON/OFF command (180/255). These trends agree qualitatively with irrigation studies reporting that continuous feedback control improves moisture regulation relative to threshold switching [33], [34], although a direct numerical comparison is not possible because those studies used physical testbeds or different models. The improvement is moderate, and PID is therefore presented here as a documented, reproducible baseline rather than as an optimised controller.

Two limitations qualify these results. First, the pump and soil coefficients are nominal rather than identified, so the absolute time scales (seconds rather than the minutes to hours typical of field soils) are illustrative, and the conclusions concern the relative behaviour of the controllers under a common model. Second, the Scilab model omits the firmware interlock that keeps the relay OFF when $\theta \geq \theta_{sp}$, the 2 s relay windowing of Eq. (6), and the 1% resolution of the integer sensor mapping; these features would modify the transient in hardware and must be assessed experimentally.

6. Conclusion and Recommendations

6.1. Conclusion

This study presented the design and simulation of a PID-controlled automatic plant watering system using an Arduino Uno in Tinkercad, with numerical analysis in Scilab based on an explicit model in which a first-order pump/motor drives a first-order soil water balance. With the baseline gains ($K_p = 5.0$, $K_i = 0.8$, $K_d = 0.1$), the 30%-to-60% response rose in 4.13 s, overshot by 29.06% of the setpoint, settled within ±2% after 42.83 s, and ended 0.10 pp from the setpoint. Similar behaviour was obtained for two further setpoint changes and under a ±1% sensor disturbance (overshoot 24.21–35.95%, settling time 39.73–46.95 s), although the disturbance increased pump switching from 5 to 23 events. Compared with a hysteresis ON/OFF controller under identical conditions, PID reduced the overshoot by 24.3% and the IAE by 20.6% and removed the sustained limit cycle and the persistent offset, at the cost of a 13.2% longer rise time. The gain-sensitivity analysis showed that the overshoot stems mainly from integral action: a lower integral gain ($K_p = 2.0$, $K_i = 0.1$, $K_d = 0.1$) eliminated it but left a 1.13 pp error at 60 s. The contribution of this work is therefore a transparent, reproducible PID baseline and simulation workflow rather than an optimised or novel controller. Several limitations must be acknowledged. First, all results derive from a software-in-the-loop (SIL) simulation; no physical hardware prototype was tested. Second, the pump and soil coefficients are nominal rather than experimentally identified. Third, the soil model is a simplified first-order water-balance approximation that neglects spatial heterogeneity, nonlinear water retention, and time-varying evaporation. Fourth, ON/OFF control is the only alternative controller evaluated. Future work should (1) build and test a physical Arduino prototype; (2) identify the pump and soil parameters experimentally; (3) re-tune the controller with a lower integral gain or a tighter anti-windup limit; and (4) compare PID with PI, fuzzy, and MPC controllers under identical conditions.

6.2. Recommendation

The following directions are recommended for future research. First, a physical hardware prototype using Arduino Uno, a capacitive soil moisture sensor, and a relay-controlled pump should be constructed and tested to

validate simulation results under real environmental conditions, including sensor noise and actuator delays. Second, comparative simulations and experiments should be conducted to evaluate PID against PI control, fuzzy logic control, and MPC under the same soil model and setpoint conditions used here for the ON/OFF comparison, providing a basis for selecting the most appropriate strategy for different application contexts. Third, the four scenarios evaluated here should be extended to different soil types, evaporation rates, and step or ramp disturbances to characterize controller robustness and generalizability more completely. Fourth, the first-order soil water-balance model should be refined and calibrated to incorporate evaporation rate variability, sensor measurement uncertainty, and water retention characteristics specific to target soil types. These extensions will substantially strengthen the scientific contribution and applicability of this work.

Author Contributions

- Widyasmoro Widyasmoro: Investigation, Formal analysis.
- Iswanto Bin Suwarno: Supervision, Validation, Writing – review & editing.
- Sulistyo Nugroho: Conceptualization, Methodology, Software, Writing – original draft.
- Nia Maharani Raharja: Data curation, Visualization.
- Muhammad Hafid Alkarim: Software, Writing – review & editing.

Competing Interest Statement

The authors declare no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Data Availability Statement

The Arduino source code, Scilab simulation scripts, Tinkercad circuit model, and simulation results supporting the findings of this study are publicly available at https://github.com/hfdalkarim/Automatic-Plant-Watering-Based-on-Soil-Moisture-Sensor-and-PID-Control-.

The repository contains:

- the shared model file (pid_model.sci),
- the three Scilab scripts (PID_Main_Simulation.sce, PID_Validation_Comparison.sce, and PID_Ziegler_Nichols_Tuning.sce) with their console outputs and exported data (CSV),
- the figure-generation script,
- the Arduino firmware (arduino.ino), and
- the link to the Tinkercad circuit.

Running the Scilab scripts reproduces all values reported in Tables 3–6 and Figures 4–7.

Statement on the Ethical Use of AI Tools

Generative AI tools, including Claude (Anthropic), were used to assist with translation, language editing, and restructuring of the manuscript text during revision. The authors reviewed and edited all AI-assisted content and take full responsibility for the content of this publication.

Funding Statement

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Acknowledgments

Institutional support provided by Universitas Muhammadiyah Yogyakarta and Peneliti Teknologi Teknik Indonesia is gratefully acknowledged for facilitating the conduct and completion of this research.

References

[1] E. Bwambale, F. K. Abagale, and G. K. Anornu, "Smart irrigation monitoring and control strategies for improving water use efficiency in precision agriculture: A review," *Agric. Water Manag.*, vol. 260, p. 107324, Feb. 2022, doi: 10.1016/j.agwat.2021.107324.

[2] H. Chen, Y. Zhang, H. Yu, and S. Li, "An integrated framework for mapping agricultural water impoundments using Sentinel-2 and GEE in Northwest China," *J.*

Hydrol., vol. 672, p. 135362, Jun. 2026, doi: 10.1016/j.jhydrol.2026.135362.

[3] K. Obaideen *et al.*, "An overview of smart irrigation systems using IoT," *Energy Nexus*, vol. 7, p. 100124, Sep. 2022, doi: 10.1016/j.nexus.2022.100124.

[4] H. Mezouane, D. Hank, and R. Delli, "Agricultural Water Management under Water Scarcity in Algeria: Practices and Future Perspectives," *Open Agric. J.*, vol. 20, no. 1, Art. no. e18743315442659, Jan. 2026, doi: 10.2174/0118743315442659251120053241.

[5] H. Li, Y. Xu, H. Yu, Z. Zhou, L. Wang, and M. Li, "Coupled water-energy-carbon footprints in intensifying agriculture: Mechanisms and spatial dynamics," *Sustain. Prod. Consum.*, vol. 63, pp. 237–254, Mar. 2026, doi: 10.1016/j.spc.2026.01.010.

[6] A. Singh, A. Verdi, D. Haver, A. Sapkota, and J. C. Iradukunda, "Using a soil moisture sensor-based smart controller for autonomous irrigation management of hybrid bermudagrass with recycled water in coastal Southern California," *Agric. Water Manag.*, vol. 299, p. 108906, Jun. 2024, doi: 10.1016/j.agwat.2024.108906.

[7] J. O. Payero, U. Sekaran, and D. B. Turner, "Using a wireless sensor network and the Internet of Things (IoT) to automate a lateral move irrigation system based on real-time soil water potential data," *Smart Agricultural Technology*, vol. 14, p. 102003, Aug. 2026, doi: 10.1016/j.atech.2026.102003.

[8] V. Sharma, G. Kaur, S. S., V. Chhabra, and R. Kashyap, "Smart irrigation systems in agriculture: An overview," *Comput. Electron. Agric.*, vol. 239, p. 111008, Dec. 2025, doi: 10.1016/j.compag.2025.111008.

[9] W. T. Adugna, F. Z. Beshah, Z. W. Tesema, and P. T. Giang, "Integrated multi-sensor remote sensing and machine learning framework for monitoring soil salinity in arid and semi-arid irrigated landscapes," *Next Research*, vol. 9, p. 101699, Jul. 2026, doi: 10.1016/j.nexres.2026.101699.

[10] P. Shi, "Developing an electrochemical sensor based on carbon nanotubes for efficient monitoring of water pollution in irrigation channels," *Microchemical Journal*, vol. 222, p. 116796, Mar. 2026, doi: 10.1016/j.microc.2025.116796.

[11] S. A. Balakrishnan *et al.*, "Implementation of a compact microwave planar sensor for real-time monitoring of soil moisture and macronutrient levels," *Measurement*, vol. 275, p. 121278, May 2026, doi: 10.1016/j.measurement.2026.121278.

[12] A.-M. Zaccarin, N. Peddada, M. Machold, K. T. Turner, and R. H. Olsson, "Inductively coupled capacitive soil moisture sensors printed on a biodegradable substrate: Characterization and long-term testing," *Smart Agricultural Technology*, vol. 11, p. 101052, Aug. 2025, doi: 10.1016/j.atech.2025.101052.

[13] H. Zhang, H. Zhang, X. Ma, and M. Cao, "A plastic optical fiber humidity sensor for high-precision soil moisture monitoring: Design, validation, and unified modeling across soil types," *Eng. Geol.*, vol. 360, p. 108474, Jan. 2026, doi: 10.1016/j.enggeo.2025.108474.

[14] M. Todd, A. J. E. Gallant, A. Wang, J. Plucinski, and V. N. L. Wong, "Quantifying inter- and intra-sensor variability in low-cost soil moisture and soil temperature sensors: A comparative study," *Smart Agricultural Technology*, vol. 12, p. 101186, Dec. 2025, doi: 10.1016/j.atech.2025.101186.

[15] H. Zhang *et al.*, "Multi-sensor (since 1997) global soil moisture mapping with enhanced Spatio-temporal coverage through machine learning framework fusion," *Remote Sens. Environ.*, vol. 334, p. 115221, Mar. 2026, doi: 10.1016/j.rse.2025.115221.

[16] N. Yazdanpanah and Y. Dong, "Calibration and performance evaluation of Internet of Things-based soil moisture sensors under saline irrigation conditions," *Smart Agricultural Technology*, vol. 12, p. 101451, Dec. 2025, doi: 10.1016/j.atech.2025.101451.

[17] D. Mandal, R. Khosla, L. Longchamps, and D. Joshi, "Soil moisture sensor location-allocation using spatial association of surface moisture data," *Smart Agricultural Technology*, vol. 11, p. 100929, Aug. 2025, doi: 10.1016/j.atech.2025.100929.

[18] S. Zhang, T. Li, Z. Shen, X. Zhang, M. Shi, and L. Zhangzhong, "Novel planar electrode soil moisture sensor: achieving full-depth measurement of soil profile," *Measurement*, vol. 258, p. 119473, Jan. 2026, doi: 10.1016/j.measurement.2025.119473.

[19] L. Wang, "Design of soil moisture sensor based on fringing electric field and microcontroller unit measurement," *Progress in Engineering Science*, vol. 3, no. 2, p. 100262, Jun. 2026, doi: 10.1016/j.pes.2026.100262.

[20] Y. Zhou *et al.*, "A dual-mode response fiber paper-based sensor based on curcumin and sodium alginate for soil moisture monitoring," *Chemical Engineering Journal*, vol. 523, p. 168909, Nov. 2025, doi: 10.1016/j.cej.2025.168909.

[21] H. Zhao, R. Bhosale, and H. M. Wainwright, "A physics-guided sensor-to-model framework for real-time estimation and near-future forecasting of soil moisture," *Adv. Water Resour.*, vol. 209, p. 105221, Mar. 2026, doi: 10.1016/j.advwatres.2026.105221.

[22] J. F. Progga *et al.*, "A chip-based radio frequency sensor for soil moisture measurements: A machine learning and deep learning calibration approach," *J. Agric. Food Res.*, vol. 25, p. 102591, Feb. 2026, doi: 10.1016/j.jafr.2025.102591.

[23] F. Hamouda, À. Puig-Sirera, L. Bonzi, D. Remorini, R. Massai, and G. Rallo, "Design and validation of a soil moisture-based wireless sensors network for the smart irrigation of a pear orchard," *Agric. Water Manag.*, vol. 305, p. 109138, Dec. 2024, doi: 10.1016/j.agwat.2024.109138.

[24] N. Sabah Abbas, M. S. Salim, and N. Sabri, "ASCD: Automatic sensing and control device for crop irrigation

scheduling," *HardwareX*, vol. 18, p. e00523, Jun. 2024, doi: 10.1016/j.ohx.2024.e00523.

[25] I. Gandarilla, V. Santibáñez, J. Sandoval, and J. G. Romero, "PID passivity-based control laws for joint position regulation of a self-balancing robot," *Control Eng. Pract.*, vol. 116, p. 104927, Nov. 2021, doi: 10.1016/j.conengprac.2021.104927.

[26] C. Zhao and L. Guo, "Towards a theoretical foundation of PID control for uncertain nonlinear systems," *Automatica*, vol. 142, p. 110360, Aug. 2022, doi: 10.1016/j.automatica.2022.110360.

[27] T. Sinico, P. Neto, and G. Boschetti, "Impact of control strategy on trajectory tracking performance in open PLC-based robot controllers: A comparison of PID and model-based control," *Results in Engineering*, vol. 28, p. 107664, Dec. 2025, doi: 10.1016/j.rineng.2025.107664.

[28] L. Yu, Y. Chen, S. Chen, Y. Zhang, H. Zhang, and C. Liu, "Numerical analysis of the performance of a PID-controlled air curtain for fire-induced smoke confinement in a tunnel configuration," *Fire Saf. J.*, vol. 141, p. 103930, Dec. 2023, doi: 10.1016/j.firesaf.2023.103930.

[29] X. Guo, Z. Lin, D. Yang, Y. Yang, L. Jiang, and Y. Liu, "Experimental evaluation on PID-based adaptive longitudinal ventilation control of tunnel fire smoke," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 254, p. 105884, Nov. 2024, doi: 10.1016/j.jweia.2024.105884.

[30] M. Y. Coskun and M. İtik, "Intelligent PID control of an industrial electro-hydraulic system," *ISA Trans.*, vol. 139, pp. 484–498, Aug. 2023, doi: 10.1016/j.isatra.2023.04.005.

[31] J. M. Gonzalez-Cava *et al.*, "Robust PID control of propofol anaesthesia: Uncertainty limits performance, not PID structure," *Comput. Methods Programs Biomed.*, vol. 198, p. 105783, Jan. 2021, doi: 10.1016/j.cmpb.2020.105783.

[32] S. Malarvili and S. Mageshwari, "Nonlinear PID (N-PID) Controller for SSSP Grid Connected Inverter Control of Photovoltaic Systems," *Electric Power Systems Research*, vol. 211, p. 108175, Oct. 2022, doi: 10.1016/j.epsr.2022.108175.

[33] E. A. Abioye *et al.*, "A data-driven Kalman filter-PID controller for fibrous capillary irrigation," *Smart Agricultural Technology*, vol. 3, p. 100085, Feb. 2023, doi: 10.1016/j.atech.2022.100085.

[34] A. E. Abioye *et al.*, "Model based predictive control strategy for water saving drip irrigation," *Smart Agricultural Technology*, vol. 4, p. 100179, Aug. 2023, doi: 10.1016/j.atech.2023.100179.

[35] E. Bwambale, F. K. Abagale, and G. K. Anornu, "Data-Driven Modelling of Soil Moisture Dynamics for Smart Irrigation Scheduling," *Smart Agricultural Technology*, vol. 5, p. 100251, Oct. 2023, doi: 10.1016/j.atech.2023.100251.

[36] S. P. Chakravarty and P. Roy, "Quantitative feedback theory-based multi-variable robust control for soil quality improvement in a drip irrigated field," *J. Process Control*, vol. 123, pp. 76–86, Mar. 2023, doi: 10.1016/j.jprocont.2023.01.009.

[37] I. Ghiat, R. Govindan, and T. Al-Ansari, "Data-driven model predictive control for irrigation management in agricultural greenhouses under CO2 enrichment," *Smart Agricultural Technology*, vol. 12, p. 101071, Dec. 2025, doi: 10.1016/j.atech.2025.101071.

[38] S. Ekinci, B. Hekimoğlu, and D. Izci, "Opposition based Henry gas solubility optimization as a novel algorithm for PID control of DC motor," *Engineering Science and Technology, an International Journal*, vol. 24, no. 2, pp. 331–342, Apr. 2021, doi: 10.1016/j.jestch.2020.08.011.

[39] A. El Mezouari, A. El Fazziki, and M. Sadgal, "Smart Irrigation System," *IFAC-PapersOnLine*, vol. 55, no. 10, pp. 3298–3303, 2022, doi: 10.1016/j.ifacol.2022.10.125.

[40] S. F. Di Gennaro, D. Cini, A. Berton, and A. Matese, "Development of a low-cost smart irrigation system for sustainable water management in the Mediterranean region," *Smart Agricultural Technology*, vol. 9, p. 100629, Dec. 2024, doi: 10.1016/j.atech.2024.100629.

[41] B. Et-taibi *et al.*, "Enhancing water management in smart agriculture: A cloud and IoT-Based smart irrigation system," *Results in Engineering*, vol. 22, p. 102283, Jun. 2024, doi: 10.1016/j.rineng.2024.102283.

[42] I. Lephondo, A. Telukdarie, I. Munien, U. Onkonkwo, and A. Vermeulen, "The Outcomes of Smart Irrigation System using Machine Learning to minimize water usage within the Agriculture Sector," *Procedia Comput. Sci.*, vol. 237, pp. 525–532, 2024, doi: 10.1016/j.procs.2024.05.136.

[43] A. Morchid, R. Jebabra, H. M. Khalid, R. El Alami, H. Qjidaa, and M. Ouazzani Jamil, "IoT-based smart irrigation management system to enhance agricultural water security using embedded systems, telemetry data, and cloud computing," *Results in Engineering*, vol. 23, p. 102829, Sep. 2024, doi: 10.1016/j.rineng.2024.102829.

[44] A. Morchid, I. G. Muhammad Alblushi, H. M. Khalid, R. El Alami, S. R. Sitaramanan, and S. M. Muyeen, "High-technology agriculture system to enhance food security: A concept of smart irrigation system using Internet of Things and cloud computing," *Journal of the Saudi Society of Agricultural Sciences*, Feb. 2024, doi: 10.1016/j.jssas.2024.02.001.

[45] T. Adamo, D. Caivano, L. Colizzi, G. Dimauro, and E. Guerriero, "Optimization of irrigation and fertigation in smart agriculture: An IoT-based micro-services framework," *Smart Agricultural Technology*, vol. 11, p. 100885, Aug. 2025, doi: 10.1016/j.atech.2025.100885.

[46] S. Gupta *et al.*, "Smart agriculture using IoT for automated irrigation, water and energy efficiency," *Smart Agricultural Technology*, vol. 12, p. 101081, Dec. 2025, doi: 10.1016/j.atech.2025.101081.

[47] R. Medina, B. Canaza, A. Sullon, M. E. Ivanovich, and G. Maquera, "Intelligent automated control of abiotic water and thermal stress in greenhouses for the

optimization of pre-basic potato production," *Smart Agricultural Technology*, vol. 12, p. 101272, Dec. 2025, doi: 10.1016/j.atech.2025.101272.

[48] M. Kuzlu, V. M. Jovanovic, O. Popescu, and S. Sarp, "Development of sensing and programming activities for engineering technology pathways using a virtual Arduino simulation platform," in *2023 ASEE Annual Conference & Exposition*, Baltimore, MD, USA, Jun. 2023, doi: 10.18260/1-2--43164.

[49] D. Athina, K. Vittal, S. Revuri, K. Shireen, and K. Keerthi, "Automatic plant watering system using moisture sensor and Arduino Uno_An experimental validation," in *Microelectronics, Circuits and Systems*, Lecture Notes in Electrical Engineering, vol. 976. Singapore: Springer Nature Singapore, 2023, pp. 485–499, doi: 10.1007/978-981-99-0412-9_44.

[50] S. A. Hamoodi, A. N. Hamoodi, and G. M. Haydar, "Automated irrigation system based on soil moisture using Arduino board," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 3, pp. 870–876, Jun. 2020, doi: 10.11591/eei.v9i3.1736.

[51] M. D. Arizki, Y. Ardiyanto, and K. T. Putra, "IoT-Based Smart Irrigation System for Aglaonema Cultivation Using Soil Temperature and Humidity Monitoring via Antares Platform," in *2025 5th International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, IEEE, Aug. 2025, pp. 389–394. doi: 10.1109/ICE3IS66769.2025.11280809.