Application of Boolean Algebra in Automatic Plant Watering System

Marzuli Suhada M - 13522070[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522070@std.stei.itb.ac.id

***Abstract*— In the context of growing automation in agriculture, this paper examines how Boolean algebra crucially influences the design and operation of an Automatic Plant Watering System. Emphasizing logical circuitry, the study explores how Boolean expressions, grounded in algebraic principles, direct watering based on real-time soil moisture. Integrating soil moisture sensors and Boolean logic enhances system efficiency, highlighting the importance of mathematical frameworks in crafting intelligent solutions for precision agriculture.**

***Keywords*— Automatic Plant Watering System, Boolean Algebra, Logical Circuits, Soil Moisture Sensors**

I. Introduction

In the realm of gardening enthusiasts, nurturing plants not only provides aesthetic pleasure but also fosters a healthier environment. Well-maintained plants enhance visual appeal and elevate oxygen levels, creating a conducive atmosphere for a healthier lifestyle. However, the challenge lies in ensuring consistent care, especially amid our hectic lives.

The busy schedules we lead often lead to oversight, and the well-being of our plants may suffer as a consequence. In response to this predicament, the development of an Automatic Plant Watering System emerges as a technological ally to ensure the sustained health and vitality of our plant life. This project tutorial focuses on a cost-effective solution, employing the ingenious 555 Timer IC to create a DIY Automatic Plant Watering System. By seamlessly integrating technology with the nurturing process, this system alleviates the burden of daily manual watering and serves as a safeguard against potential neglect.

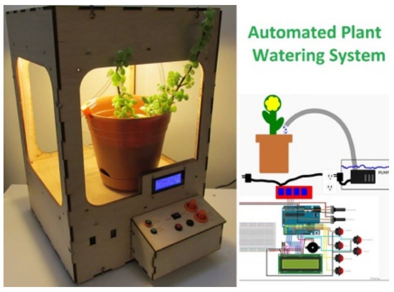


Fig. 1.1 Automatic Plant Watering System
Source: https://www.instructables.com/Automated-Plant-Watering-System/

This paper delves into the pivotal role played by Boolean algebra in the implementation of such automatic systems. Boolean logic becomes the backbone of decision-making processes within the watering system, as it interprets real-time data on soil moisture levels. The marriage of gardening and technology, exemplified by this Automatic Plant Watering System, not only addresses the common issue of forgetfulness or time constraints but also sets the stage for a more sustainable and efficient approach to plant care.

As I explore the intricacies of this DIY project, the underlying theme is the harmonious coexistence of nature and technology. The integration of Boolean algebra in this context serves as a testament to our ability to leverage innovation for the betterment of our environment, fostering a future where the care of our green companions is seamlessly woven into the fabric of our busy lives.

II. Theoretical Basis

A. Boolean Algebra

1. Definition

Boolean algebra is defined as a tuple <B, +, ·, ', 0, 1>, where B is a set defined with two binary operators, namely + and ·, and one unary operator, namely '. Subsequently, 0 and 1 in the tuple represent two distinct elements of B. The conditions for the tuple to be considered a Boolean algebra are as follows: for every a, b, c, which are members of B, the following axioms must be satisfied:

a. Identity
   i. $a + 0 = a$
   ii. $a \cdot 1 = a$
b. Commutative
   i. $a + b = b + a$
   ii. $a \cdot b = b \cdot a$
c. Distributive
   i. $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
   ii. $a + (b \cdot c) = (a + b) \cdot (a + c)$
d. Complement
   For every a member of B, there exists a unique element a' in B such that
   i. $a + a' = 1$
   ii. $a \cdot a' = 0$

With the specified axiom conditions, it can be understood that both set algebra and propositional logic algebra are instances of

Boolean algebras as they fulfil these four axioms. Upon closer inspection, the '+' operation in Boolean algebra is akin to the 'or' operation in propositional logic algebra, while the '·' operation in Boolean algebra is analogous to the 'and' operation in propositional logic algebra.

2. *Boolean Algebra 2-Value (Binary)*

Boolean Algebra 2-Value, widely regarded as the most renowned type of Boolean Algebra, operates with variables that can only assume two values: true (1) or false (0). In this specific algebraic system, denoted as Boolean Algebra 2-Value, the set B is defined as $\{0, 1\}$. The binary operators are defined using + and ·, while the unary operator is defined using '. All these operators play a crucial role in performing logical operations within this algebraic framework.

In Boolean Algebra 2-Value, the four axioms that constitute the conditions for a Boolean algebra must be satisfied. These axioms define the fundamental properties that underpin the algebraic system, ensuring consistency and logical coherence.

3. *Boolean Expression*

Boolean expressions are formed from combinations of elements from set B using binary or unary operators. Examples of Boolean expressions are as follows:

a. 0
b. 1
c. A
d. a + b
e. a'
f. a · (b + c)

4. *Boolean Algebra Laws*

Boolean Algebra, a mathematical structure dealing with binary variables and logic operations, follows several fundamental laws that govern its operations. Here is a table explanation of some key Boolean Algebra laws:

Table 2.1 Boolean Algebra Laws

1. Identity Laws: (i) a + 0 = a (ii) a . 1 = a	2. Idempotent Laws: (i) a + a = a (ii) a . a = a
3. Complement Laws: (i) a + a’ = 1 (ii) a . a’ = 0	4. Dominance Laws: (i) a . 0 = 0 (ii) a + 1 = 1
5. Involution Law: (i) (a’)’ = a	6. Absorption Laws: (i) a + ab = a (ii) a(a + b) = a
7. Commutative Laws: (i) a + b = b + a (ii) ab = ba	8. Associative Laws: (i) a + (b + c) = (a + b) + c (ii) a . a = a
9. Distributive Laws: (i) a + (bc) = (a + b)(a + c) (ii) a(b + c) = ab + bc	10. De Morgan Laws: (i) (a + b)’ = a’b’ (ii) (ab)’ = a’ + b’
11. 0/1 Laws: (i) 0’ = 1 (ii) 1’ = 0	

Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/11-Aljabar-Boolean-(2023)-bagian1.pdf

5. *Boolean Functions*

Boolean functions refer to mathematical functions that involve variables known as literals. A literal, in this context, is a Boolean expression that holds a specific value. These functions are commonly used in computational logic, especially in the design and analysis of logical circuits.

For example, consider a few Boolean functions:

a. $f(x) = x$: This function has one literal, namely x. It means that the function depends on the single value of the variable x.
b. $g(x, y, z) = xy'z$: This function has three literals, x, y', and z. The literal y' represents the complement of y, indicating the inverse value of y.

6. *Canonical Forms*

There are two main canonical forms in Boolean algebra: the Sum of Products (SOP) form and the Product of Sums (POS) form.

a. *Sum of Products (SOP)*

In SOP form, a Boolean expression is represented as the logical sum (OR) of product terms. Each product term is a combination of variables and their complements, connected by logical AND operations. The canonical SOP form is unique for any Boolean function and provides a standardized way to express the function.

Example: If $f(x, y, z) = \sum(1, 3, 5, 7)$, the SOP form is $f(x, y, z) = x'y'z + x'yz + xy'z + xyz$.

b. *Product of Sums (POS)*

In POS form, a Boolean expression is represented as the logical product (AND) of sum terms. Each sum term is a combination of variables and their complements, connected by logical OR operations. Like SOP form, the canonical POS form is unique for any Boolean function.

Example: If $f(x, y, z) = \sum(0, 2, 4, 6)$, the POS form is $f(x, y, z) = (x + y + z)(x + y' + z)(x' + y + z)(x' + y' + z)$.

B. *Logic Circuit*

Logic circuit is a fundamental component in digital electronics that processes binary information using Boolean algebra. In its most general form, a logic circuit takes binary inputs (0s and 1s) and produces binary outputs based on logical operations. These operations include conjunction, disjunction, and negation.

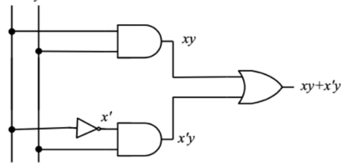


Fig. 2.1 Example of Logic Circuit
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/11-Aljabar-Boolean-(2023)-bagian1.pdf

Logic circuits are designed by interconnecting these logical operations to create a system that performs specific Boolean functions. These circuits serve as the foundation for digital systems, ranging from simple electronic devices to complex computer architectures.

C. Logic Gates

Logic gates are the elemental building blocks of digital circuits, forming the backbone of electronic systems that process and manipulate binary information. In Boolean algebra, these gates perform fundamental logical operations on binary inputs, paving the way for the design and functionality of complex electronic devices. Three primary logic gates — AND, OR, and NOT — each with distinct characteristics, contribute to the intricate landscape of digital circuitry.

1. AND Gate

The output of an AND gate is true (1) only if both of its inputs are true (1). In other words, it performs the logical AND operation.

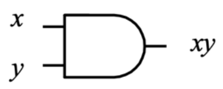


Fig. 2.2 AND Gate
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/11-Aljabar-Boolean-(2023)-bagian1.pdf

Table 2.2 Truth Table for AND Operation

AND Gate with 2 Inputs		
x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Gate

The output of an OR gate is true (1) if at least one of its inputs is true (1). It performs the logical OR operation.

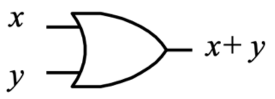


Fig. 2.3 OR Gate
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/11-Aljabar-Boolean-(2023)-bagian1.pdf

Table 2.3 Truth Table for OR Operation

AND Gate with 2 Inputs		
x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Gate

The output of a NOT gate is the opposite (complement) of its input. If the input is true (1), the output is false (0), and vice versa.

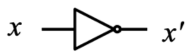


Fig. 2.4 NOT Gate
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/11-Aljabar-Boolean-(2023)-bagian1.pdf

Table 2.4 Truth Table for NOT Operation

NOT Gate	
x	x'
0	1
1	0

D. Karnaugh Map

A Karnaugh Map, commonly known as a K-map, serves as a valuable graphical tool in the realm of Boolean algebra and digital circuit design. Named after Maurice Karnaugh, an American mathematician and engineer, this method streamlines the process of simplifying algebraic expressions and optimizing logic circuits. It provides a visual representation that elucidates the relationships between different combinations of input variables and their corresponding output values.

Constructing a Karnaugh Map involves determining all possible combinations of input variables and organizing them in a grid format. Each cell in the grid represents a unique combination, containing binary values (0 or 1) indicative of the output for the respective input combination. This map becomes particularly useful for simplifying Boolean expressions and navigating truth tables efficiently.

As an example, suppose we want to minimize the following Boolean function:

$$f(x) = x'yz + xy'z' + xyz + xyz'$$

This Boolean function can be minimized using a Karnaugh Map by grouping adjacent squares with a value of 1. The grouping can be observed in the Karnaugh Map below:

Table 2.5 Karnaugh Map

yz x	00	01	11	10
0	0	0	1	0
1	1	0	1	1

Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/12-Aljabar-Boolean-(2023)-bagian2.pdf

Through this grouping, the simplified form of the minimized function using the Karnaugh Map technique is obtained as follows:

$$f(x) = yz + xz'$$

It is important to note that the result of simplifying a Boolean function using the Karnaugh Map is not always unique. This means there are several different forms of minimized functions with the same number of literals and terms.

III. Boolean Algebra Implementation in Automatic Plant Watering System

A. Concept

The Automatic Plant Watering System operates on a simple premise, automatically activating the water pump when soil moisture drops below a specific level and deactivating it when moisture is sufficient. Key components include Soil Moisture Probes, LM358 Op-Amp IC, 555 Timer IC, CL100 Transistor, Relay, and a water pump. Soil moisture, detected by the probes through soil resistance, determines the voltage sent to the LM358 Op-Amp IC. This voltage is analyzed, generating an output for the 555 Timer IC, regulating the water pump's operating time. The CL100 transistor acts as a switch, controlling relay operations that, in turn, manage the water pump's activation and deactivation.

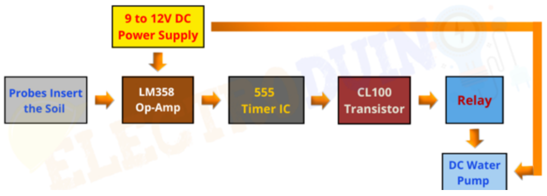


Fig. 3.1 Block Diagram of Automatic Plant Watering System using 555 Timer IC
Source: https://www.electroduino.com/automatic-plant-watering-system-using-555-timer-ic/

B. Determining Factors of Automatic Plant Watering System

The Automatic Plant Watering System incorporates several crucial factors to ensure efficient and tailored plant care. These determining factors include a manual switch for on/off control, an adjustable interval time setting, and a soil moisture sensing mechanism.

1. Switch On/Off

The system integrates a manual switch, providing users with the ability to manually control the operation of the device. This feature allows users to decide when to activate or deactivate the watering system based on specific plant care requirements.

2. Interval Time Setting

To accommodate diverse plant watering needs, the system includes an interval time setting. Users can customize the time duration between each watering cycle, offering flexibility to meet the unique moisture requirements of different plants. This ensures that plants receive water at optimal intervals, contributing to their health and growth.

3. Soil Moisture Sensing

The heart of the Automatic Plant Watering System lies in its soil moisture sensing mechanism. The system employs a sophisticated circuit comprising LM358 Op-Amp IC for soil moisture detection and a 555 timer IC for timing control.

a. *Setting Threshold Voltage:* Before testing the circuit, it is essential to set the threshold voltage at the Inverting input (Pin 2) of the LM358 Op-Amp IC. This adjustment is made using a potentiometer knob (VR1) to determine the sensor's sensitivity in dry soil conditions.

b. *Dry Soil Conditions:* When the soil is dry, the soil moisture probes exhibit high resistance and low conductivity. This results in a high voltage at the Inverting input of the LM358 IC. Consequently, the IC produces a Low output, triggering the 555 timer IC. The timer, in turn, activates the relay, switching on the water pump and initiating the watering process.

c. *Moist Soil Conditions:* Conversely, when the soil is moist, low resistance and high conductivity enable voltage to pass through the soil. This leads to a low voltage at the Inverting input of the LM358 IC. The IC responds by producing a High output, resetting the 555 timer IC. The timer then generates a Low output, deactivating the relay. This, in turn, turns off the water pump, preventing overwatering.

C. Logic Circuit Input as a Representation of the Factors

Based on those three factors, the writers illustrate the system's operation by variables x, y, and z.

1. Switch On/Off

The variable x mirrors the manual switch's functionality. In this context, if x is active (1), it signifies that the system is in an ON state, and the watering mechanism is engaged. Conversely, if x is inactive (0), the system is in an OFF state, and the watering process is halted.

2. Interval Time Setting

Similar to the timer in the previous analysis, the variable y represents an adjustable interval time setting. An active state (1) for y indicates that the system is currently within the specified time interval, allowing for flexible watering schedules. Conversely, when y is inactive (0), the system does not adhere to a specific time interval, responding to soil conditions immediately.

3. Soil Moisture Sensing

In this scenario, the variable z reflects the conditions of soil moisture, the writers simplifying it into a binary state: whether the soil is dry or moist. If z is equal to 1 or ON, then the soil is in a moist condition, or the water level is more than sufficient. If z is equal to 0 or OFF, then the soil is in a dry condition or needs water.

D. Truth Table

From the representation created in the previous section, we can design a truth table. The columns in this truth table contain each bit representing the input (current conditions) and output (activation or deactivation of the watering system) based on the combinations of the three inputs. Meanwhile, the rows in the table represent the combinations or possibilities that can occur. The total number of possible combinations is $2^3 = 8$. All these possibilities can be clearly seen in the table below.

Table 3.1 Truth Table for the system

Logic Circuit for Automatic Plant Watering System			
Input			Output
x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Source: Author

E. Karnaugh Map and Logic Circuit

After designing the Truth Table, the next step involves creating Karnaugh Maps for each bit of the input with three variables, namely bits x, y, and z. The purpose of these Karnaugh Maps is to simplify each Boolean function we have derived from the truth table, aiming to minimize the use of logic gates in the subsequent circuit.

Each cell in the Karnaugh Map can be filled with 0, 1, or X based on the previously created truth table. However, in this case, the writer does not use the variable X or the "don't care" condition. After filling each cell with the appropriate values, the simplification process for the Boolean functions takes place.

Table 3.2 Karnaugh Map for the System

x \ yz	00	01	11	10
0	0	0	0	1
1	1	1	1	1

Source: Author

The simplification involves combining cells with a value of 1 that are adjacent to each other, following the minimization rules of the Karnaugh Map. Therefore, the result of simplifying the Boolean functions is in the Sum of Product (SOP) form.

$$f(x,y,z) = x'yz' + xy'z' + xy'z + xyz + xyz'$$
$$f(x,y,z) = xy(z + z') + xy'(z + z') + yz'(x + x')$$
$$f(x,y,z) = xy + xy' + yz'$$
$$f(x,y,z) = x(y + y') + yz'$$
$$f(x,y,z) = x + yz'$$

From the function, it can be understood that the automatic plant watering system integrates three key factors as inputs in a logic circuit, where various logic gates determine the generation of the output. Initially, a NOT gate is applied to the soil moisture sensing input because the criteria for initiating plant watering are when the soil moisture sensor detects dryness or insufficient moisture, necessitating the inversion of the soil moisture sensing input. Subsequently, the inverted soil moisture sensing result is combined with the interval time setting input in an AND gate. This combination reflects the condition for activating the watering system when the interval time is enabled, and the soil moisture sensor registers low moisture levels. The output of this AND gate becomes the input for an OR gate along with the switch input, ultimately determining whether the automatic plant watering system will be activated or deactivated. The logic circuit assumes an OFF state for all determining factors or when the switch is turned off before the specified interval time is reached.

Here is the logic circuit designed for the automatic plant watering system with all determining factors in the initial state.

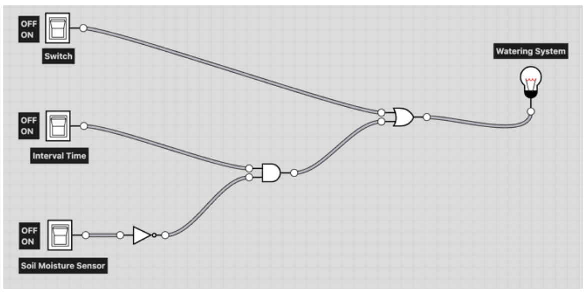
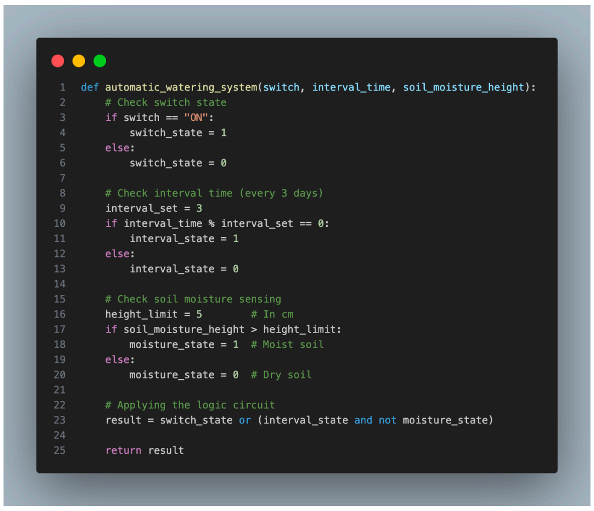


Fig. 3.2 Logic Circuit of Automatic Plant Watering System
Source: Author

F. Implementation the System in Simple Code

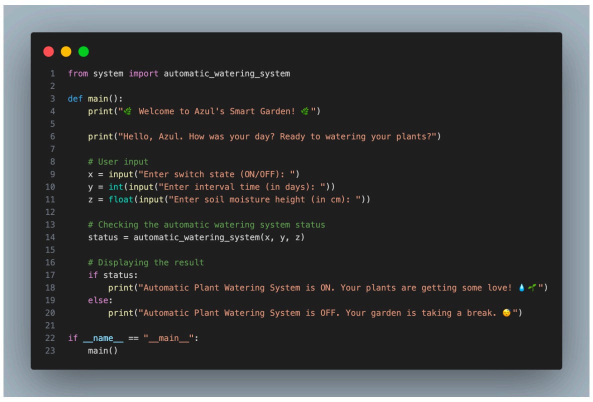
The *automatic_watering_system* function is designed to simulate the logic of an automatic plant watering system based on three input parameters: *switch, interval_time,* and *soil_moisture_height*.



```python
def automatic_watering_system(switch, interval_time, soil_moisture_height):
    # Check switch state
    if switch == "ON":
        switch_state = 1
    else:
        switch_state = 0

    # Check interval time (every 3 days)
    interval_set = 3
    if interval_time % interval_set == 0:
        interval_state = 1
    else:
        interval_state = 0

    # Check soil moisture sensing
    height_limit = 5         # In cm
    if soil_moisture_height > height_limit:
        moisture_state = 1  # Moist soil
    else:
        moisture_state = 0  # Dry soil

    # Applying the logic circuit
    result = switch_state or (interval_state and not moisture_state)

    return result
```

Fig. 3.3 Implementation of Boolean Algebra Function in Automatic Plant Watering System in Python
Source: Author

Firstly, it evaluates the state of the manual switch. If the switch is in the "ON" position, the *switch_state* is set to 1, indicating an active system, and 0 otherwise. Next, it checks whether the specified interval has passed. If the condition is met, the *interval_state* is set to 1; otherwise, it is set to 0. Additionally, the function examines the soil moisture level by comparing it to a predefined height limit. If the soil moisture surpasses this limit, *moisture_state* is set to 1, indicating moist soil; otherwise, it is set to 0 for dry soil. Finally, the function computes the overall result by applying a logical circuit. The watering system is deemed active *(result = 1)* if the manual switch is ON or if both the interval condition is satisfied, and the soil moisture is not high. The logical expression used is *switch_state or (interval_state and not moisture_state).*



```python
from system import automatic_watering_system

def main():
    print("🌿 Welcome to Azul's Smart Garden! 🌿")

    print("Hello, Azul. How was your day? Ready to watering your plants?")

    # User input
    x = input("Enter switch state (ON/OFF): ")
    y = int(input("Enter interval time (in days): "))
    z = float(input("Enter soil moisture height (in cm): "))

    # Checking the automatic watering system status
    status = automatic_watering_system(x, y, z)

    # Displaying the result
    if status:
        print("Automatic Plant Watering System is ON. Your plants are getting some love! 💧🌱")
    else:
        print("Automatic Plant Watering System is OFF. Your garden is taking a break. 😴")

if __name__ == "__main__":
    main()
```

Fig. 3.4 Implementation of the main program in Python
Source: Author

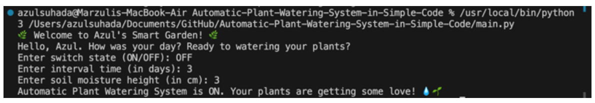
The *main* function serves as the central component of the program. The program awaits user input for three crucial factors. Subsequently, the function invokes the *automatic_watering_system* from the imported *system* module, passing the gathered inputs. The automatic watering system's status is then determined, reflecting whether the defined conditions for system activation are met. If the system is active, then plant watering will take place. Conversely, if the system is inactive, it indicates a resting period for the plants or watering stops.

IV. Testing

In this testing, the author assumes that soil moisture in plants is determined by the water height present. If the water height is above 5 cm, the soil is considered moist; conversely, if the water height is less than or equal to 5 cm, the soil is deemed dry. Furthermore, the author also assumes that the watering interval is set every 3 days. The author also uses the output of a plant watering system, which is considered "ON" when the final output is 1 or true, and "OFF" when the final output is 0 or false. As there are three determining factors, there are a total of 8 possible cases for this logic circuit. However, the author will discuss three potential cases that may occur in this automatic plant watering system.

1. Switch OFF, Day-3, Dry Condition

In this case, the switch is turned off, and it is already the 3rd day since this plant has been cared for. It is also known that the water height is at 3 cm, indicating that the soil is currently dry. This might occur because it has been quite some time since the last time the plant was watered, which was on the first day. In this scenario, the system is activated because the plant requires water, and automatic watering takes place.



```
● azulsuhada@Marzulis-MacBook-Air Automatic-Plant-Watering-System-in-Simple-Code % /usr/local/bin/python
  3 /Users/azulsuhada/Documents/GitHub/Automatic-Plant-Watering-System-in-Simple-Code/main.py
  🌿 Welcome to Azul's Smart Garden! 🌿
  Hello, Azul. How was your day? Ready to watering your plants?
  Enter switch state (ON/OFF): OFF
  Enter interval time (in days): 3
  Enter soil moisture height (in cm): 3
  Automatic Plant Watering System is ON. Your plants are getting some love! 💧🌱
```

Fig. 4.1 Testing if switch of, day-3, dry condition
Source: Author

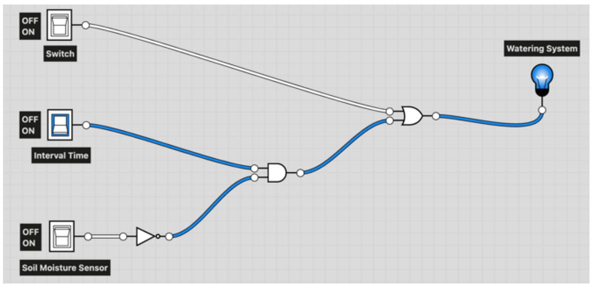
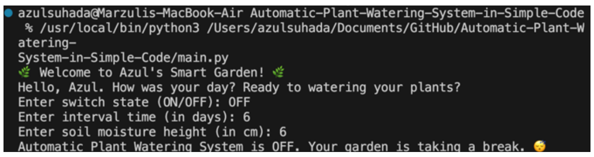


Fig. 4.2 Logic circuit if switch of, day-3, dry condition
Source: Author

2. Switch OFF, Day-6, Moist Condition

In this case, the switch is turned off, and it is already the 6th day since this plant has been cared for. It is known that the water height is at 6 cm, indicating that the soil is currently moist. Although the time interval has been reached, the watering system does not activate or turns off automatically because continuous watering could lead to the plant having excess water, resulting in unhealthy conditions and the risk of wilting.



```
● azulsuhada@Marzulis-MacBook-Air Automatic-Plant-Watering-System-in-Simple-Code
   % /usr/local/bin/python3 /Users/azulsuhada/Documents/GitHub/Automatic-Plant-W
  atering-
  System-in-Simple-Code/main.py
  🌿 Welcome to Azul's Smart Garden! 🌿
  Hello, Azul. How was your day? Ready to watering your plants?
  Enter switch state (ON/OFF): OFF
  Enter interval time (in days): 6
  Enter soil moisture height (in cm): 6
  Automatic Plant Watering System is OFF. Your garden is taking a break. 😴
```

Fig. 4.3 Testing if switch off, day-6, moist condition
Source: Author

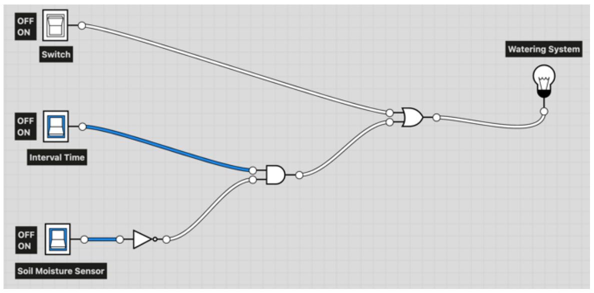
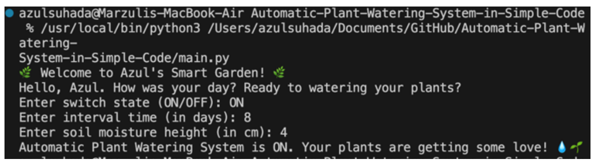


Fig. 4.4 Logic circuit if switch off, day-6, moist condition
Source: Author

3. Switch ON, Day-8, Dry Condition

In this case, the switch is turned on, and it is already the 8th day since this plant has been cared for. It is known that the water height is at 4 cm, indicating that the soil is currently dry. In this case, the watering system is activated because the switch is turned on, allowing watering to take place. When the switch is on, the conditions of the time interval or soil moisture sensor do not affect the system because the true value of the switch can already make the watering system operational. This case occurs when the plant owner intentionally wants to perform watering manually.



```
● azulsuhada@Marzulis-MacBook-Air Automatic-Plant-Watering-System-in-Simple-Code
   % /usr/local/bin/python3 /Users/azulsuhada/Documents/GitHub/Automatic-Plant-W
  atering-
  System-in-Simple-Code/main.py
  🌿 Welcome to Azul's Smart Garden! 🌿
  Hello, Azul. How was your day? Ready to watering your plants?
  Enter switch state (ON/OFF): ON
  Enter interval time (in days): 8
  Enter soil moisture height (in cm): 4
  Automatic Plant Watering System is ON. Your plants are getting some love! 💧🌱
```

Fig. 4.5 Testing if switch on, day-8, dry condition
Source: Author

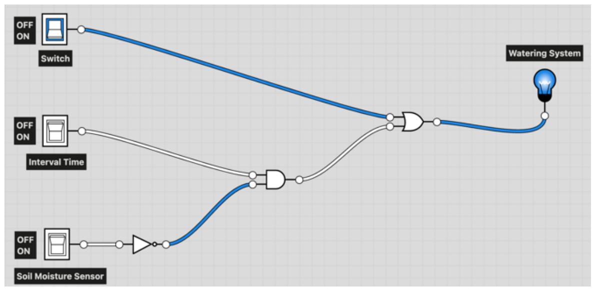


Fig. 4.6 Logic circuit if switch on, day-8, dry condition
Source: Author

V. Conclusion

In this paper, the author explores the significant role of Boolean Algebra in everyday life, particularly in the design and functionality of the Automatic Plant Watering System, emphasizing the logical circuitry of its operation. By leveraging Boolean expressions rooted in algebraic principles, this system achieves precise watering based on real-time soil moisture conditions. The integration of soil moisture sensors and Boolean logic not only enhances system efficiency but also underscores the importance of mathematical frameworks in creating intelligent solutions for precision agriculture.

The logic circuit presented in this paper can also simulate various possible inputs for the three determining factors that activate watering in the automatic plant watering system. The system will be turned off when the switch is off, either the time interval has not been reached, or the soil is detected as moist, or when the time interval condition is met but the soil is detected as moist. Besides these conditions, the watering system in the automatic plant watering system will be activated.

From this paper, the key takeaway is that the seamless integration of technology and gardening through Boolean logic can pave the way for a smarter, more efficient, and sustainable approach to plant care in our busy lives. In the future, the author hopes to have a better understanding of how Boolean algebra is applied to other systems.

VI. Appendix

Here are some attachments:

1. GitHub source code used to create this paper: https://github.com/zultopia/Automatic-Plant-Watering-System-in-Simple-Code.git
2. The author employs the tools available at https://logic.ly/demo to simulate the logic circuit in the design of this automatic plant watering system. Here is the result of all possible cases that can occur from the system: https://drive.google.com/drive/folders/1L7tWVpfiFblNRBhrh8sP-aw2z-6-aWFw?usp=sharing

VII. Acknowledgment

The author expresses gratitude to Allah SWT for His blessings and grace, allowing the completion of this paper within the stipulated timeframe. The author would like to thank to Dr. Fariska Zakhralativa Ruskanda, S.T., M.T., as the lecturer for the Discrete Mathematics course in the first semester of 2022/2023, Class 02, for guidance and teachings throughout the semester. The author would like to thank my parents for continuous support during the educational journey and last for all my friends who have assisted and supported in the completion of this paper.

References

[1] Electroduino. 2023. "Automatic Plant Watering System using 555 Timer IC." https://www.electroduino.com/automatic-plant-watering-system-using-555-timer-ic/#Working_Principle_of_Automatic_Plant_Watering_System_using_555_Timer_IC, accessed December 1, 2023.

[2] Munir, Rinaldi. 2023. "Aljabar Boolean (Bagian 1)." https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/11-Aljabar-Boolean-(2023)-bagian1.pdf, accessed December 2, 2023.

[3] Munir, Rinaldi. 2023. "Aljabar Boolean (Bagian 2)." https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/12-Aljabar-Boolean-(2023)-bagian2.pdf , accessed December 2, 2023.

[4] Panda, Sneha. 2021. "D Flip Flop: Circuit, Truth Table, Working, Critical Differences". https://lambdageeks.com/d-flip-flop-circuit-working-truthtable-differences/, accessed December 4, 2023.

[5] E-thaksalawa. 2021. "Logic Gates with Boolean Functions". https://www.e-thaksalawa.moe.gov.lk/moodle/pluginfile.php/15815/mod_resource/content/1/SG10_ICT_Chapter4.pdf, accessed December 8, 2023.

[6] Khan, E. 2018. "Automatic Plant Watering & Irrigation System – Circuit, Code & Project Report". https://www.electricaltechnology.org/2018/08/automatic-plant-watering-irrigation-system.html, accessed December 9, 2023.

[7] Inggi, Rahmat and Rizal. 2020. "Perancangan Alat Pengontrol Ketinggian Air Dan Penyiraman Tanaman Secara Otomatis Berbasis Arduino Pada Media Tanam Hidroponik". https://media.neliti.com/media/publications/328107-perancangan-alat-pengontrol-ketinggian-a-d94097e5.pdf, accessed December 10, 2023.

[8] Punitha.K, Shivaraj Sudarshan Gowda, and Devarajnayaka R. 2017. "Automated Plant Watering System". https://www.ijert.org/research/automated-plant-watering-system-IJERTCONV5IS18012.pdf, accessed December 10, 2023.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 5 Desember 2023

[illegible]

Marzuli Suhada M, 13522070