IoT-Based Chili Plant Watering Automation Using NodeMCU ESP8266 and Blynk when the Pump is Running

Nuril Mustofa, Sunardi
Department of Electrical Engineering, Universitas Ahmad Dahlan, Yogyakarta, Indonesia

ARTICLE INFORMATION

Article History:

Submitted 13 June 2022
Revised 10 December 2022
Accepted 13 January 2023

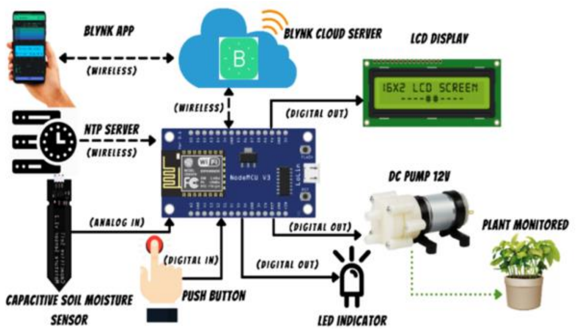
Keywords:

IoT;
NodeMCU;
Blynk;
Automation;
Sprinkling

Corresponding Author:

Sunardi,
Department of Electrical Engineering,
Universitas Ahmad Dahlan,
Yogyakarta, Indonesia
Email: sunardi@mti.uad.ac.id

ABSTRACT



Automatic plant watering system can help users in caring for plants. Along with the development of technology, it is possible to monitor and control using the Internet of Things (IoT) from anywhere and anytime as long as the device is connected to the internet. The system designed in this study performs watering on chili plant automatically and in real time monitored through the Blynk application on a smartphone. Automation is carried out based on the moisture value parameter obtained from the capacitive soil moisture sensor as input and the NodeMCU ESP8266 as the controller. The output of the system is water sprinkling that comes out through a 12V DC water pump as an actuator and the Blynk application as a monitor and controller via IoT. Automation and monitoring through smartphones using the Blynk application in this study have been successfully carried out. Watering can be done regularly according to predetermined time intervals automatically and the amount of water given to plants according to their needs. At a humidity that is less than 60% and the schedule is appropriate, the pump will run for 4 seconds with a water discharge of 116.32 ml which has been adjusted to the volume of soil and water needs of chili plant.

Document Citation:

N. Mustofa and Sunardi, “IoT-Based Chili Plant Watering Automation Using NodeMCU ESP8266 and Blynk when the Pump is Running,” *Buletin Ilmiah Sarjana Teknik Elektro*, vol. 5, no. 1, pp. 1-11, 2023, DOI: 10.12928/biste.v5i1.6164.

1. INTRODUCTION

Water is essential for all life, be it for humans, animals, plants, it also applies to microorganisms [1]. Water is also used for industrial, agricultural, fire-fighting, reforestation, transportation, and other purposes [2]. For plants, water is the main ingredient of the plant body. The water content in plants varies between 70-90%, depending on age, species, tissues, and environment [3]. Watering plants is closely related to the condition of moisture content in the soil. The need for sufficient water is one of the most important things. Lack of moisture content or excess moisture content can result in plants not being able to grow well [4]. Regular and proportional watering consistently, especially if it is on a wide scale and done manually, is certainly not easy and requires more effort to get maximum crop yields [5].

Humans often make mistakes in everyday life, one of which is forgetting or not remembering what should be done. It can also happen due to busy daily activities or multitasking which can reduce the ability to concentrate for long periods of time and increase the time it takes to complete certain jobs, but also increase the risk of making mistakes so that things like watering plants are often forgotten [6]. To reduce the occurrence of errors, a system that can work automatically to carry out watering plants [7] is needed. The use of sensors that can determine the moisture content in the soil is believed to be one of the attractive solutions according to the needs of plants. The condition of the soil moisture content is obtained from the soil moisture sensor [8].

Internet of Things, also known as the abbreviation IoT, is a concept that aims to expand the benefits of continuously connected internet connectivity. With the development of internet infrastructure, the world is heading for the next round where not only smartphones or computers can be connected to the internet. But various kinds of real objects will be connected to the internet. For example it can be production machines, cars, electronic equipment, wearables, and includes any tangible object that is all connected to local and global networks using embedded sensors and/or actuators [9].

Controlling through IoT is very helpful for humans because they can carry out activities from anywhere and anytime as long as the device is connected to the internet so that plants are expected to meet the water needs of plants. In addition to control, IoT can also monitor or monitor in real time so that users can know directly the field conditions and the status of the devices used. This advancement in IoT technology can facilitate a variety of jobs including watering plants [10]. This control or monitoring can be done through websites or applications on smartphones that are almost owned by everyone where in 2019 there were at least 3.2 billion gadget users globally, up to 5.6% from the previous year. While the number of active devices used reached 3.8 billion units. In 2022, the number of smartphone users is predicted to reach 3.9 billion users [11].

Automatic watering systems can ease the burden of providing water when plants need it, automation can be used or utilized to help with routine work because it can run continuously without knowing the time [12]. Automation can be done based on a capacitive soil moisture sensor as input, NodeMCU ESP8266 microcontroller as controller, water pump as output, and Blynk application as monitor and controller via IoT. Sistem automatic control is a form of system control in the absence of human intervention in its mechanism of action [13]. With these devices, automation and monitoring of watering water on plants can be done easily.

2. METHODS

The design of this system is carried out with two stages of design, namely hardware design and software design. Hardware design required block diagrams and circuit diagrams. The design of the software contains a flow chart of the method used.

2.1. Hardware Design

The hardware design consists of a control system, the smallest NodeMCU system, which acts as a sensor controller and performs data processing. The design of the system is presented in the form of a block diagram that will help to design an automation tool of the plant watering system using a soil moisture sensor. The block diagram of the plant watering system automation tool can be seen in Figure 1.

The system is divided into three parts, namely input, process, and output. In the input section, there are four hardware components and virtually/software. In the process section there is only one, namely the N odeMCU ESP 8266 microcontroller. In the output section, it consists of four hardware and software components. NTP Server is a software input component that is entered through a program with the NTPClient.h library and with the functions contained in it. NTP Server is used as a real time input on systems that takes real time. A capacitive soil moisture sensor is an input part that is used to measure soil moisture in this system. The push button is used as a manual control for manually watering plants. The blynk application is an application platform used as a real time monitoring and controller that can be done via a smartphone with an internet connection. LCD (Liquid Crystal Display) is used to display the necessary data such as moisture value, ADC value, soil condition, and pump condition. The pump is used as a system output that plays a role in draining water from the reservoir to plant objects. LED indicator as a system indicator to find out yang being performed by the system.

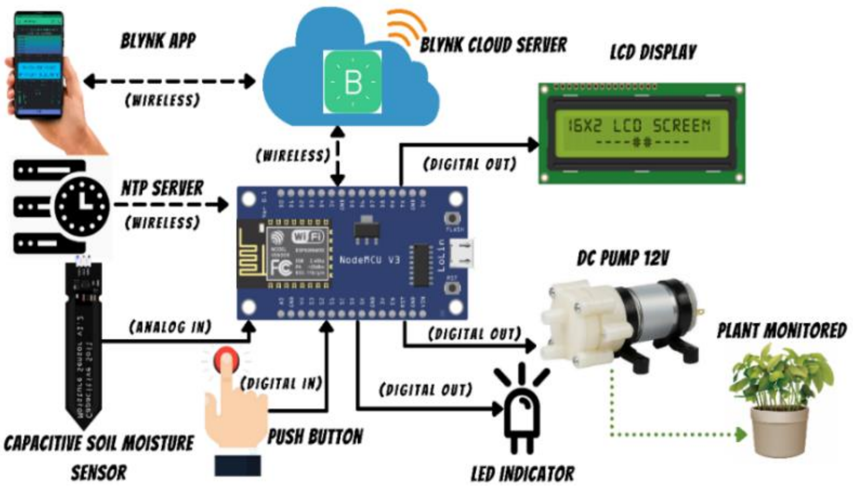


Figure 1. System block diagram

2.2. Hardware System Wiring Diagram

The cabling design of the hardware system is a series of several components, namely the soil moisture capacitive sensor, water pump, LCD display, indicators, and the NodeMCU ESP8266 microcontroller with the aim of ensuring that it works properly and optimally so that no trouble occurs.

Figure 2 is a hardware prototype design made using 1 relay to control a DC water pump that is directly connected to the power supply to turn on or connect and disconnect the electric current in the circuit and protect other components to avoid overvoltage or circuits that are accidentally short-connected (short circuit) on the circuit. The capacitive soil moisture sensor is used to read soil moisture or as an input processed by the MCU, namely the NodeMCU ESP8266 and then the system output can be monitored via a smartphone. Table 1 is the addressing of pins and hardware inputs and outputs.

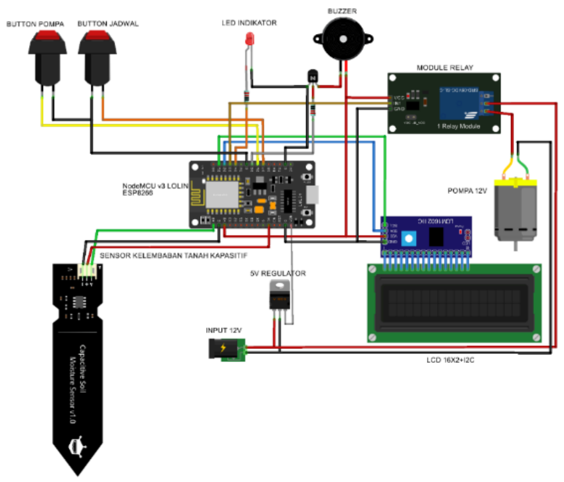


Figure 2. Schematic hardware sistem

Table 1. Addressing the output input pins

No.	Component	Microcontroller Pins	Pin Power Supply
1	Capacitive soil moisture sensor	A0	5V, GND
2	Button manual	D6	5V, GND
3	Button jadwal	D7	5V, GND
4	Relay	D3	5V, GND
5	Buzzer	D4	5V, GND
6	LED	D5	5V, GND
7	I2C LCD	SDA, SCL	5V, GND
8	DC Pump 12V	-	5V, GND

2.3. Software Design

Microcontroller programming is essential for executing the necessary commands on the microcontroller circuit. In this study, MCU programming used Arduino IDE software and C language. Flowcharts are used as a reference for writing programs that work according to commands. The flowchart in this study can be seen as in Figure 3. When the hardware input voltage is supplied, the microcontroller will start the process of initializing the required inputs and outputs and variables. The data entering the microcontroller is processed and executed on the output drive circuit, that is, the relay turns on the DC water pump to make the plants spray water.

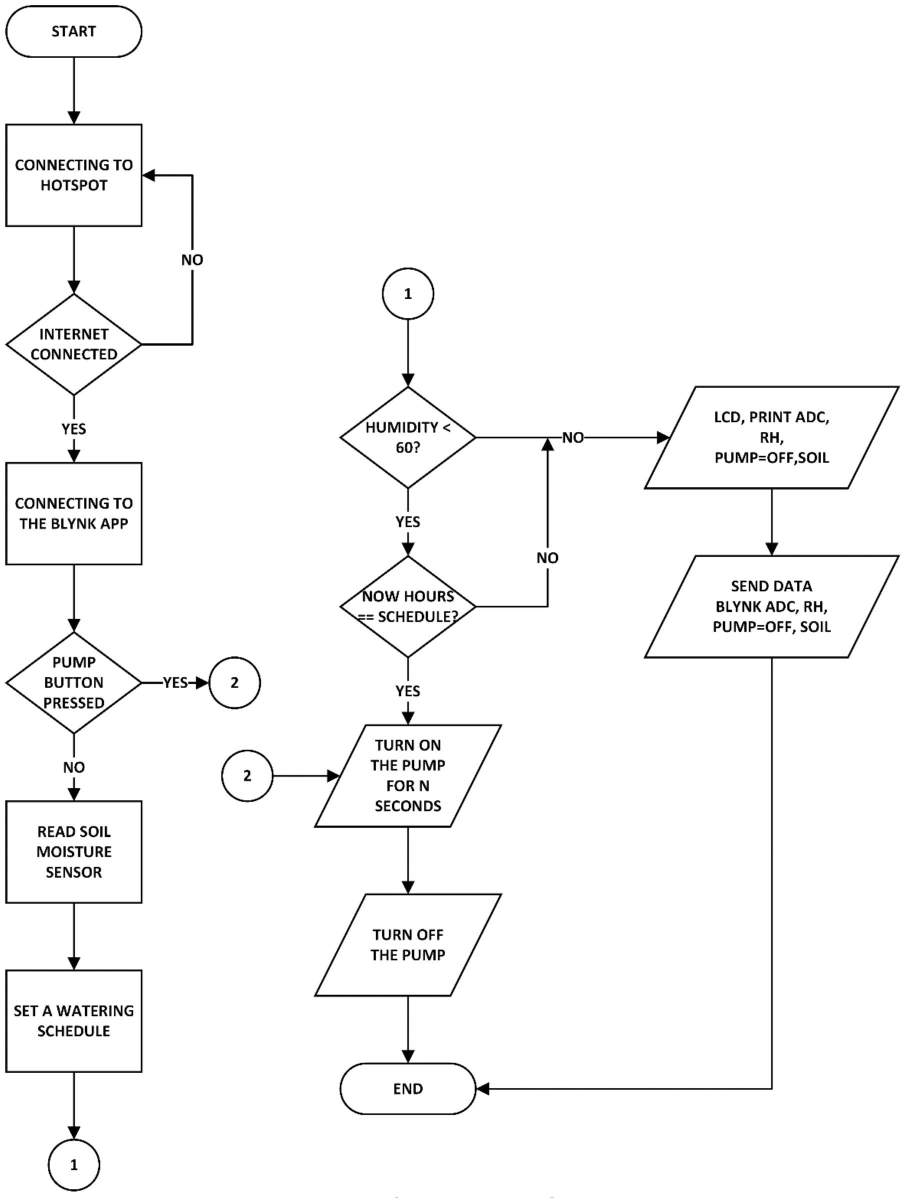


Figure 3. Software system flowchart

3. RESULTS AND DISCUSSION

3.1. Soil moisture sensor testing

The value of the soil moisture sensor must be calibrated by comparing the value read by the microcontroller with the output voltage. The test was carried out by observing the values displayed on the LCD. To obtain a variation in value, it is plugged into the soil and given water until the value changes. The output voltage of the sensor is observed by reading the voltage value using a digital multimeter tool. This method was also used in research conducted [14] to calibrate ADC values on microcontrollers. The ADC value is calculated by Equation (1).

$$Value\ ADC = \frac{sensor\ output\ voltage}{reference\ voltage} \times 1023 \quad (1)$$

$$Value\ ADC = \frac{2.115}{3.284} \times 1023 = 658$$

Based on Tabel 2, the results of the calibration test of the ADC value of the soil sensor have a very small error of 0.353%. Error occurs due to the accuracy of the multimeter readings and the interference from the noise signal that interferes with the sensor voltage output. Based on the physical g in Figure 4, the difference in the ADC value in the tool with the calculation ADC value is relatively small.

Table 2. Sensor ADC calibration test

NO.	ADC	Voltage	ADC Theory	|ADC Theory-ADC Tools|	Error (%)
1	655	2.115	658.844	3.844	0.375
2	527	1.688	525.829	1.170	0.114
3	493	1.584	493.430	0.432	0.042
4	451	1.455	453.247	2.247	0.219
5	507	1.623	505.581	1.418	0.138
6	382	1.246	388.141	6.141	0.600
7	392	1.275	397.175	5.175	0.505
8	284	0.946	294.688	10.688	1.044
9	416	1.346	419.292	3.292	0.321
10	351	1.152	358.859	7.859	0.768
11	415	1.346	419.292	4.292	0.419
12	269	0.865	269.456	0.4564	0.044
Total					**4.596**
Error Average (%)					**0.353**

The difference in data or error values is obtained from Equations (2) and (3) with the calculation example.

$$Difference = |Tool\ ADC\ value - Displayed\ ADC\ value| \quad (2)$$

$$Percentage\ Error = \frac{|Difference|}{|Reference\ Value|} \times 100\% \quad (3)$$

$$Diference = |658.844 - 655| = 3.844$$

$$Percentage\ Error = \frac{3.844}{1023} \times 100\% = 0.375\%$$

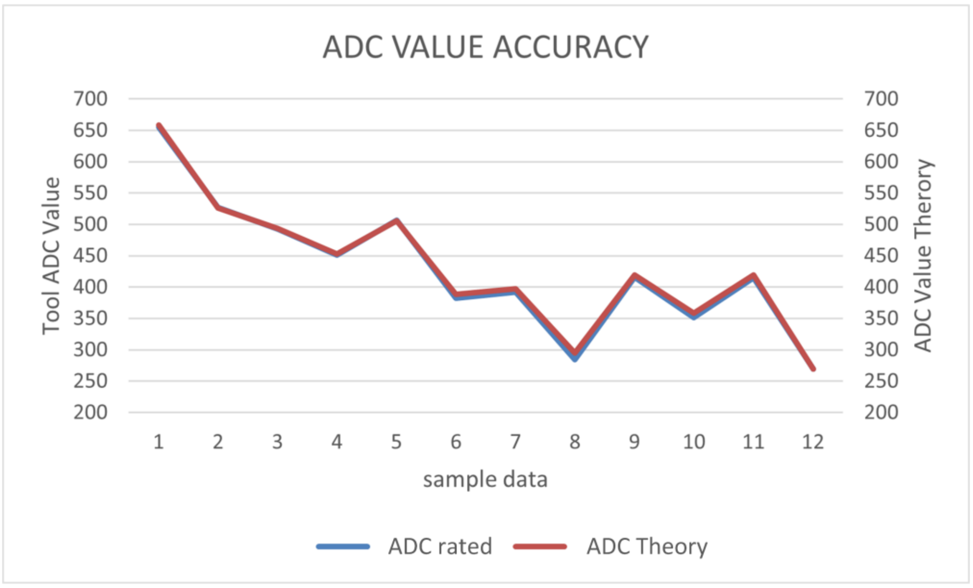


Figure 4. Sensor ADC comparison test graph

3.1.1 Soil Moisture Value Calibration Test

In the calibration test, the value of the soil moisture is carried out by observing the results of the moisture value which is read on the tool and then compared by the calculation of the moisture value in theory. Sampling is carried out by giving water as much as+- 200 ml. The moisture value of the tool is obtained from the ADC value and then processed by a microcontroller. Based on the calculation from Equation (4), the soil can be calculated with an example of data collection.

$$Soil\ Moisture = \frac{wet\ soil\ weight - dry\ soil\ weight}{dry\ soil\ weight} \times 100\% \tag{4}$$

$$Soil\ Moisture = \frac{643 - 517}{517} \times 100\% = 24.3\%$$

As can be seen from Table 3, the average error is 0.711%. Error is caused by noise from external interference and the depth of attachment of the sensor to the ground. Error is obtained by the calculation:

$$Difference = |10.57 - 9.5| = 1.07$$

Based on Table 3, the sample data yielded an average error of 0.711% which was compared to the theoretical value. This calibration testing method was also carried out in research [15] using the American Standart Method theory. As seen from the graph in Figure 5, the difference in the graph is relatively small, the difference value looks larger at values above 60% humidity.

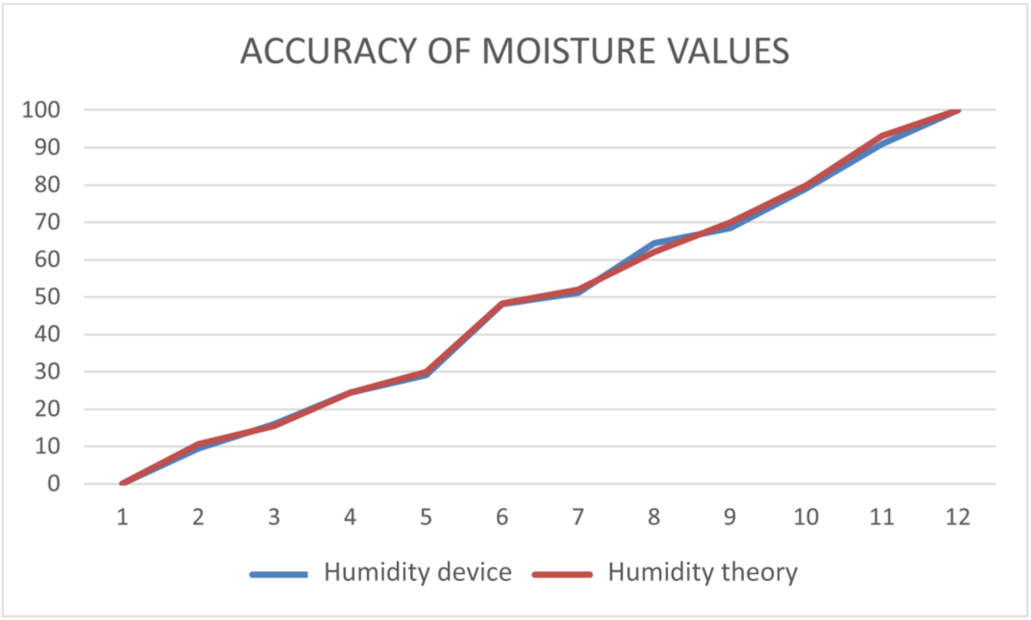


Figure 5. Graph of the difference in testing moisture values

Table 3. Sensor moisture calibration testing against ASM theory

NO.	Tool humidity	Humidity theory	| The difference|	Error
1	0	0	0	0
2	9.5	10.57	1.07	1.07
3	15.9	15.50	0.40	0.40
4	24.3	24.37	0.07	0.07
5	29.0	30.00	1.00	1.00
6	45.4	45.60	0.20	0.20
7	51.1	52.00	0.90	0.90
8	61.3	61.00	0.30	0.30
9	68.4	71.00	2.60	2.60
10	76.0	76.90	0.90	0.90
11	90.0	91.10	1.10	1.10
12	100	100	0	0
		Total		**8.54**
		Average error		**0.711**

3.2. Schedule Review

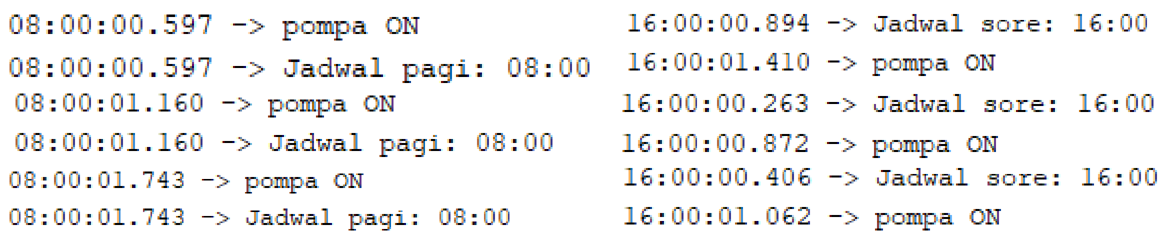
Schedule testing aims to test the accuracy of the system's response to time. The schedule is TRUE if the schedule value/time is equal to the current time. The system response can be observed through the serial monitor software Arduino IDE as in Figure 6. This system does not use Realtime Time Clock (RTC) hardware, but uses NTP client or Network Time Protocol where time is taken from the accompaniment protocol which functions to time synchronization in the form of dates, hours, minutes, and seconds [16][17][18]. From the time reference, the schedule can be adjusted. However, the response of the system is not 100% the same as the time of the NTP, that is, there is a slight difference. The results of the test schedule can be seen in Tabel 4.

As can be seen in Table 4, the system does not execute the schedule 100% accurately, but there is a lead time error or precedence and also a delay time or delayed in a few milliseconds. However, the error is relatively small, which is based on Table 4 is 1.261% or within 0.756 seconds. Error is small because it does not reach 1 second. Error is obtained by Equations (2) and (3).

$$Difference = |\text{set time} - \text{executed time}|$$

$$Difference = |08{:}00{:}00{,}000 - 08{:}00{:}00{,}597| = 597 \text{ ms}$$

$$Percentage\ Error = \frac{597}{60} \times 100\%/1000 = 0.995\%$$



```
08:00:00.597 -> pompa ON                16:00:00.894 -> Jadwal sore: 16:00
08:00:00.597 -> Jadwal pagi: 08:00      16:00:01.410 -> pompa ON
08:00:01.160 -> pompa ON                16:00:00.263 -> Jadwal sore: 16:00
08:00:01.160 -> Jadwal pagi: 08:00      16:00:00.872 -> pompa ON
08:00:01.743 -> pompa ON                16:00:00.406 -> Jadwal sore: 16:00
08:00:01.743 -> Jadwal pagi: 08:00      16:00:01.062 -> pompa ON
```

Figure 6. Testing schedule via serial monitor Arduino IDE

Table 4. Schedule accuracy testing

No.	Set time watering schedule	Time to watering on the tool	Time difference (ms)	Error (%)
1	08:00:00.000	08:00:00.597	0.97	0.995
2	08:00:00.000	08:00:01.160	1.160	1.933
3	08:00:00.000	08:00:01.743	1.743	2.905
4	08:00:00.000	08:00:00.560	0.560	0.933
5	08:00:00.000	08:00:00.230	0.230	0.383
6	08:00:00.000	08:00:00.020	0.020	0.033
7	16:00:00.000	16:00:01.410	1.410	2.350
8	16:00:00.000	16:00:00.872	0.872	1.453
9	16:00:00.000	16:00:01.062	1.062	1.770
10	16:00:00.000	16:00:00.327	0.327	0.545
11	16:00:00.000	16:00:00.094	0.094	0.156
12	16:00:00.000	16:00:01.008	1.008	1.680
Total			**9.083**	
Average error (%)				**1.261**
Error in units of time (s)			**0.756**	

3.3. Pump Actuator Testing

Pump testing aims to find out the function and accuracy of the pump that acts as one of the system outputs. The pump becomes an actuator or drive that will drain water from the reservoir into the ground or plant objects [19]. Pump testing is carried out by dividing three tests, namely testing the discharge of water discharged by pump, testing the volume of water discharged by the pump, and testing the set of time the pump turns on.

3.3.1 Testing the Discharge of Water Discharged by the Pump

Testing the water discharge on this pump aims to determine the accuracy of the water discharge released by the pump. The water discharge is calculated by calculating the volume of water divided by the time the pump turns on. Data sampling is carried out by turning on the pump for 5 seconds.

In Table 5, the pump is turned on for 5 seconds on each sample. This is done to find out the accuracy of the water discharge released by the pump with the same length of time. The discharge of water released experiences a difference in expenditure due to several factors such as inaccurate flame time and fluctuating voltage on the pump. The water discharge can be calculated using Equation (5).

$$water\ discharge = \frac{the\ volume\ of\ water\ that\ comes\ out}{when\ the\ pump\ is\ running} \quad (5)$$

$$water\ discharge = \frac{150\ ml}{5s} = 30\ \mathrm{ml/s}$$

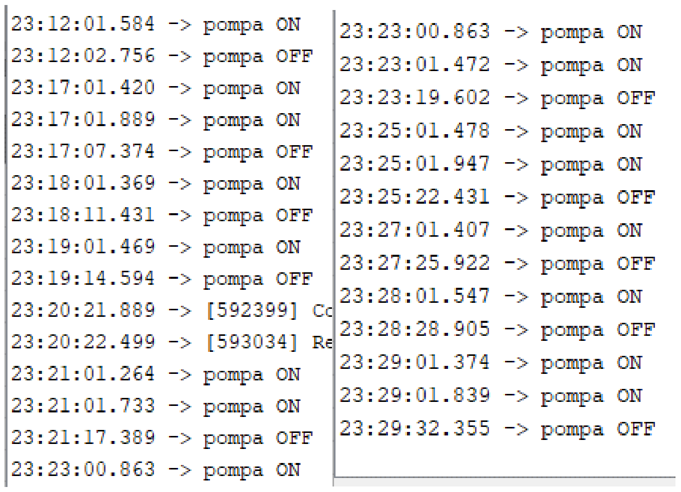
Tabel 5. Water discharge testing

No.	Length of flame(s)	Volume (ml)
1	5	150
2	5	148
3	5	149
4	5	134
5	5	136
6	5	156
7	5	135
8	5	153
9	5	154
10	5	139
Total		**1454**
Average discharge (ml)		**145.40**
Discharge (ml/s)		**29.08**

After obtaining a water discharge of 29.08 ml/s, the value can be used as a reference to set the pump on time parameter so that the water released by the pump can be in accordance with the needs of the safe tan object in order to maintain dampness.

3.3.2 Testing Pump Live Time Set

Testing the time set of the pump on aims to find out the accuracy of the pump time on. The time the pump is on will be related to the water discharge that will be released by the pump, so the accuracy of the pump time turns on greatly affects the system. The test is carried out by setting a varied time as in Table 6. After setting the pump time to turn on, the system response is observed by looking at the serial monitor Arduino IDE as in Figure 7.



```
23:12:01.584 -> pompa ON
23:12:02.756 -> pompa OFF
23:17:01.420 -> pompa ON
23:17:01.889 -> pompa ON
23:17:07.374 -> pompa OFF
23:18:01.369 -> pompa ON
23:18:11.431 -> pompa OFF
23:19:01.469 -> pompa ON
23:19:14.594 -> pompa OFF
23:20:21.889 -> [592399] Cc
23:20:22.499 -> [593034] Re
23:21:01.264 -> pompa ON
23:21:01.733 -> pompa ON
23:21:17.389 -> pompa OFF
23:23:00.863 -> pompa ON
```

```
23:23:00.863 -> pompa ON
23:23:01.472 -> pompa ON
23:23:19.602 -> pompa OFF
23:25:01.478 -> pompa ON
23:25:01.947 -> pompa ON
23:25:22.431 -> pompa OFF
23:27:01.407 -> pompa ON
23:27:25.922 -> pompa OFF
23:28:01.547 -> pompa ON
23:28:28.905 -> pompa OFF
23:29:01.374 -> pompa ON
23:29:01.839 -> pompa ON
23:29:32.355 -> pompa OFF
```

Figure 7. Accuracy testing of the pump on

In Table 6, the pump time test was on with 10 data collection samples with varying times of 1000 ms, 5000 ms, 10000 ms, 13000 ms, 15000 ms, 18000 ms, 20000 ms, 24000 ms, 27000 ms, and 30000 ms. Respons system or pump output response turned on turned out to be inaccurate according to the timer that had been set. The response experienced a difference between 62 ms to 1125 ms. This happened due to an inaccurate timer from the internal microcontroller. However, the error that occurs is very small with an average value of 620 ms or 0.6 seconds.

Table 6. Pump life time testing

No.	Set on time (ms)	Length of flame pump (ms)	Difference (ms)
1	1000	1208	208
2	5000	5954	954
3	10000	10062	62
4	13000	13125	125
5	15000	16125	1125
6	18000	18739	739
7	20000	20953	953
8	24000	24515	515
9	27000	27538	538
10	30000	30981	981
		Total	**6200**
		Average (ms)	**620**

3.4. System Testing

Testing the system as a whole is a benchmark and aims to ensure that the system can work as it has been designed. Sample sampling is carried out for three days in the morning to evening (12 hours) and data is taken every 2 hours. Data parameters include ADC value, soil p-to-earth value, pump condition, and pump on time[20].

Based on Table 7, Table 8, and Table 9, the overall system testing obtained observational data for three days with data collection once every 2 hours per day from 6 am to 6 pm resulting in 7 data samples per day, so that the number of data collection samples was 21 data samples.

Table 7. Overall system testing day 1

No.	Hit	ADC	Moisture	Moisture	Set schedule	Condition pump	When the pump is running
1	06:00	375	47.0 (%)	DRY		OFF	-
2	08:00	375	47.0 (%)	DRY	8:00	ON	4
3	10:00	278	95.5 (%)	WET		OFF	-
4	12:00	304	82.5 (%)	WET		OFF	-
5	14:00	313	78.0 (%)	NRM	16:00	OFF	-
6	16:00	350	59.5 (%)	DRY		ON	4
7	18:00	275	97.0 (%)	WET		OFF	-

Table 8. Overall system testing day 2

No.	Hit	ADC	Moisture	Moisture	Set schedule	Condition pump	When the pump is running
1	06:00	375	47.0 (%)	DRY		OFF	-
2	08:00	375	47.0 (%)	DRY	8:00	ON	4 s
3	10:00	278	95.5 (%)	WET		OFF	-
4	12:00	304	82.5 (%)	WET		OFF	-
5	14:00	313	78.0 (%)	NRM	16:00	OFF	-
6	16:00	350	59.5 (%)	DRY		ON	4 s
7	18:00	275	97.0 (%)	WET		OFF	-

Table 9. Overall system testing day 3

No.	Hit	ADC	Moisture	Moisture	Set schedule	Condition pump	When the pump is running
1	06:00	350	59.5 (%)	DRY		OFF	-
2	08:00	353	58.0 (%)	DRY	8:00	ON	4 s
3	10:00	283	93.0 (%)	WET		OFF	-
4	12:00	288	90.5 (%)	WET		OFF	-
5	14:00	295	87.0 (%)	WET	16:00	OFF	-
6	16:00	299	85.0 (%)	WET		OFF	-
7	18:00	302	85.5 (%)	WET		OFF	-

On day 1 data collection as in Table 7, the system works as desired by the indicator, namely in No. 2 the system reads soil moisture worth 47.0% on the system LCD which looks like in Figure 8 testing the system as a whole where this value is less than the 60% limit and at 8:00 the pump is on. This is because the running time is equal to the schedule time that has been set. The pump turns on for 4 seconds based on testing the discharge of water discharged by the pump, the pump discharges water discharge of about -+116.32 ml. Similarly, at no. 6 at 16:00 the moisture is worth 59.5 which means that this value is less than the 60% limit so the pump turns on.

On the 2nd day of data collection as in Table 8, the system does not run due to the moisture value of the soil read by the tool <60% so that the pump does not turn on even though the schedule has been set according to the running time. The humidity value of <60% occurs due to cloudy or even rainy weather conditions so that the water contained in the soil does not evaporate quickly or is even exposed to rainwater and the pump will not turn on.

On day 3 data collection as in Table 9, the system turns on pump at 08:00 because the humidity read by the tool is 58.0 or <60%. In the afternoon, namely at 16:00, the system does not accept the pump because the humidity value has not reached a value of <60%.

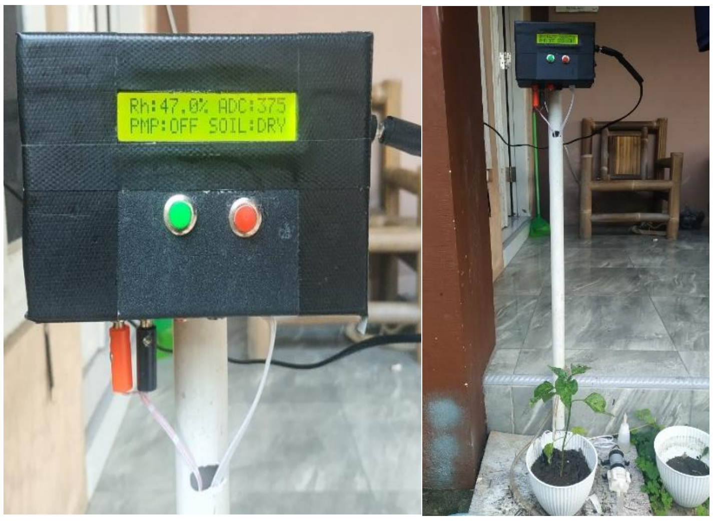


Figure 8. Testing the system as a whole

4. CONCLUSIONS

After designing the prototype of the tool and conducting a tool test by observing the results of data that has been taken from each component and the entire system, it can be concluded that the capacitive soil moisture sensor can be used to detect soil moisture with an error of 0.875% so that the design of an automatic watering tool can be used to detect soil moisture with an error of 0.875% so that the design of an automatic watering tool can be done with the ESP8266 NodeMCU as a processor. The system can detect soil loss and can determine the soil in dry, normal, and wet conditions so that it can control the pump as an actuator can drain water from the reservoir into the plant soil automatically and monitored through the blynk application in IoT. The system is able to turn on the pump / water the plant if the humidity is <60% and the schedule is in accordance with the running time.

REFERENCES

[1] L. Cortesi, Filtering Dirty Water and Finding Fresh One: Engaging with Tradition in Dug-Well Intervention in North Bihar," In *Informing Water Policies in South Asia*, pp. 314-333, 2018, https://www.taylorfrancis.com/chapters/edit/10.4324/9781315734156-13/filtering-dirty-water-finding-fresh-one-luisa-cortesi.

[2] M. J. Schneider, *Introduction to public health*, Jones & Bartlett Learning, 2020, https://books.google.co.id/books?id=Of_2DwAAQBAJ&dq=+Introduction+to+environmental+health+.

[3] M. U. Hassan, *et al.*, "Heat stress in cultivated plants: Nature, impact, mechanisms, and mitigation strategies—A review," *Plant Biosystems-An International Journal Dealing with all Aspects of Plant Biology*, vol. 155, no. 2, pp. 211-234, 2021, https://doi.org/10.1080/11263504.2020.1727987.

[4] M. Mayuree, P. Aishwarya and A. Bagubali, "Automatic Plant Watering System," *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking* (ViTECoN), pp. 1-3, 2019, https://doi.org/10.1109/ViTECoN.2019.8899452.

[5] F. Steiner, A. M. Zuffo, A. Busch, T. D. O. Sousa, end T. Zoz, "Does seed size affect the germination rate and seedling growth of peanut under salinity and water stress?," *Pesquisa Agropecuária Tropical*, vol. 49, 2019, https://doi.org/10.1590/1983-40632019v4954353.

[6] R. B.-F. Rachel F. Adler, "Juggling on a high wire: Multitasking effects on performance," *Int. J. Human-Computer Stud.*, vol. 70, no. 2, pp. 156–168, 2012, https://doi.org/10.1016/j.ijhcs.2011.10.003.

[7] S. Bhardwaj, S. Dhir and M. Hooda, "Automatic Plant Watering System using IoT," *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 659-663, 2018, https://doi.org/10.1109/ICGCIoT.2018.8753100.

[8] M. F. Obead, I. A. Taha, and A. H. Salaman, "Design and implementation of irrigation prototype system based GSM," *Journal Port Science Research*, vol. 4, no. 2, pp. 90-95, 2021, https://doi.org/10.36371/port.2021.2.5%20.

[9] M. A. Tuan Tran, T. N. Le and T. P. Vo, "Smart-Config Wifi Technology Using ESP8266 for Low-Cost Wireless Sensor Networks," *2018 International Conference on Advanced Computing and Applications (ACOMP)*, pp. 22-

28, 2018, https://doi.org/10.1109/ACOMP.2018.00012.

[10] Y. Setiawan, H. Tanudjaja, and S. Octaviani, "Penggunaan Internet of Things (IoT) untuk Pemantauan dan Pengendalian Sistem Hidroponik," *TESLA J. Tek. Elektro*, vol. 20, no. 2, p. 175, 2019, https://doi.org/10.24912/tesla.v20i2.2994.

[11] I. Irfan, A. Aswar, and E. Erviana, "Hubungan Smartphone Dengan Kualitas Tidur Remaja Di Sma Negeri 2 Majene," *J. Islam. Nurs.*, vol. 5, no. 2, p. 95, 2020, https://doi.org/10.24252/join.v5i2.15828.

[12] R. Tullah, Sutarman, and A. H. Setyawan, "Sistem Penyiraman Tanaman Otomatis Berbasis Mikrokontroler Arduino Uno Pada Toko Tanaman Hias Yopi," *J. Sisfotek Glob.*, vol. 9, no. 1, pp. 100–105, 2019, https://doi.org/10.38101/sisfotek.v9i1.219.

[13] M. S. Sihombing, S. Suhada, and I. P. Sari, "Prototype of Automatic Water Sprayer Based on Humidity Sensor and ATmega8 AVR Microcontroller in Oil Palm Nurseries," *JOMLAI: Journal of Machine Learning and Artificial Intelligence*, vol. 1. no. 2, pp. 159-166, 2022, https://doi.org/10.55123/jomlai.v1i2.934.

[14] J. S. Saputra and Siswanto, "Prototype Sistem Monitoring Suhu Dan Kelembaban Pada Kandang Ayam Broiler Berbasis," vol. 7, no. 1, 2020, https://doi.org/10.30656/prosisko.v7i1.2132.

[15] F. N. Shuhaimi, N. Jamil, and R. Hamzah, "Evaluations of Internet of Things-based personal smart farming system for residential apartments," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 6, pp. 2477-2483, 2020, https://doi.org/10.11591/eei.v9i6.2496.

[16] A. Mtibaa and S. Mastorakis, "NDNTP: A Named Data Networking Time Protocol," in *IEEE Network*, vol. 34, no. 6, pp. 235-241, November/December 2020, https://doi.org/10.1109/MNET.011.2000169.

[17] O. E. Amestica, P. E. Melin, C. R. Duran-Faundez and G. R. Lagos, "An Experimental Comparison of Arduino IDE Compatible Platforms for Digital Control and Data Acquisition Applications," *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pp. 1-6, 2019, https://doi.org/10.1109/CHILECON47746.2019.8986865.

[18] W. Gay, "Real-Time Clock (RTC)," *In Beginning STM32 Apress*, pp. 175-193, 2018, https://doi.org/10.1007/978-1-4842-3624-6_10.

[19] E. A. Sideris, and H. C. de Lange, "Pumps operated by solid-state electromechanical smart material actuators-A review," *Sensors and Actuators A: Physical*, vol. 307, p. 111915, 2020, https://doi.org/10.1016/j.sna.2020.111915.

[20] I. Chakraborty, A. Agrawal and K. Roy, "Design of a Low-Voltage Analog-to-Digital Converter Using Voltage-Controlled Stochastic Switching of Low Barrier Nanomagnets," in *IEEE Magnetics Letters*, vol. 9, pp. 1-5, Art no. 3103905, 2018, https://doi.org/10.1109/LMAG.2018.2839097.

AUTHOR BIOGRAPHY

Nuril Mustofa is a graduate student of the Department of Electrical Engineering at Universitas Ahmad Dahlan (Yogyakarta, Indonesia). Born in Tanjung Makmur on March 18, 1999. Nuril Mustofa is interested in electronics, especially microcontrollers, IoT and embedded systems.

Sunardi completed his undergraduate education in the Department of Electrical Engineering at Gadjah Mada University, completed his Masters in the Department of Electrical Engineering at the Bandung Institute of Technology, and completed his Doctorate in the Department of Electrical Engineering at Universiti Teknologi Malaysia. Currently Sunardi is a permanent lecturer in the Department of Electrical Engineering at Universitas Ahmad Dahlan (Yogyakarta, Indonesia). His research field is Wireless Communication.