iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79581

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

IoT Based Smart Farming System

Adil Zafar[1], Arbaz Ahmad[2], Abdul Rahman[3], Mr. Obaidullah[4]
Computer Science and Engineering Integral University, Lucknow, India

Abstract: ***Smallholder farmers across India, including those in and around Uttar Pradesh, still manage irrigation the way their grandparents did: they walk to the field, look at the soil, and make a call. Some days that means too much water; other days, not nearly enough. Neither outcome is good for crops or for a region where water is already under stress [11]. This paper describes an IoT-driven irrigation system that our three-member team designed, assembled, and tested as our final-year project. The hardware is built around a NodeMCU ESP8266 [5], a resistive soil moisture sensor, and a DHT11 module [?] for tracking air temperature and humidity. Together they push readings to the Blynk cloud platform [4] over Wi-Fi every two seconds, where the data shows up on a smartphone dashboard in near-real time. When soil moisture drops below a threshold set in the firmware, a relay module closes the pump circuit and irrigation begins—no one needs to be present. We ran the prototype through several hours of testing across different soil wetness levels and found that pump response was consistently fast, remote visibility worked as intended, and the total component cost came out at Rs. 4500, well inside our starting budget.***

Keywords: IoT, Smart Farming, Precision Agriculture, NodeMCU ESP8266, Soil Moisture Sensor, DHT11, Blynk IoT, Automated Irrigation, Remote Monitoring, Sustainable Agriculture.

I. INTRODUCTION

A. Background

Walk through most small farms in Uttar Pradesh on an or- dinary morning and you will find irrigation decisions are still made the same way they were fifty years ago: a farmer checks the soil by hand, checks the sky, and decides. That knowledge is not worthless—experienced farmers do develop a genuine feel for what their land needs—but it has a hard limit. It re- quires the person to be physically present, and it generates no data that can be acted on remotely or looked back at later.

The core idea behind the Internet of Things is that physical objects, when fitted with sensors and a network connection, can collect and share data that removes exactly this kind of information gap [7]. In farming, that means putting low-cost sensors in the ground, connecting them to a microcontroller, and letting the readings drive decisions that would otherwise require someone on site [1]. The NodeMCU ESP8266 [5] suits this role well: it costs around Rs. 400, carries a full Wi- Fi stack onboard, and is programmable through the same Ar- duino IDE [6] that most engineering students already use.

B. Motivation

We had been thinking about this problem for a while before the project formally began. Two of us come from farming families, and the irrigation question had come up in real con- versations at home—how do you actually know when the crop needs water? Once the literature review started, we found that researchers have been building IoT-based irriga- tion systems for years [2,3,9], but most of those systems were either too expensive, too technically involved, or both, for a smallholder in rural India [13].

That gap is where our project sits. We wanted to find out whether the core functionality—automatic threshold-based pump control plus remote monitoring from a phone—could be built for under Rs. 5000 using parts available at any elec- tronics market in Lucknow. It turns out it can, and this paper documents how we did it.

II. PROBLEM STATEMENT

The problem is not that farmers do not care about their crops. The problem is that they have no instrument giving them con- tinuous feedback about what the soil actually contains [8]. A farmer who waters every morning on a fixed schedule will overwater on rainy days and underwater on unusually hot ones, simply because the schedule cannot respond to condi- tions it has no way to observe.

Add to that the reality of a busy agricultural household. Planting, harvests, markets, family—there are days when walking out to check a field simply does not happen. The crop waits. If the wait goes on long enough, yield drops. Commer- cial smart-irrigation products address this in principle, but their pricing and their assumption of reliable infrastructure make them impractical for the farmers we had in mind [3,13]. The question going into this project was: what is the simplest, cheapest version of this solution that still actually works?

©IJRASET: All Rights are Reserved

III. OBJECTIVES

A. *Primary Objectives*

Six concrete targets were defined for the prototype, all veri- fied during the testing phase:

1) Continuously sample soil moisture, temperature, and hu- midity throughout the day and night using field-mounted sensors [**?**].
2) Automatically switch the water pump on when measured soil moisture drops beneath a preset threshold, and cut it off once the threshold is recovered.
3) Push all three sensor values to the Blynk cloud [4] in real time over the NodeMCU's onboard Wi-Fi [5].
4) Display live sensor data and pump state on a mobile dashboard readable on any basic Android handset.
5) Reduce the number of manual interventions needed to maintain appropriate soil moisture in a small plot.
6) Let the farmer check field status and take manual control of the pump from anywhere with an internet connection.

B. *Secondary Objectives*

Beyond the automation core, a few additional things had to be right. A manual override was non-negotiable from the beginning—automation should make life easier, not remove control from the person who owns the land. The hardware was also structured so that additional sensors, such as a rain gauge or soil pH probe [9], could be added later without dis- turbing the existing wiring. Finally, the Blynk dashboard was kept as simple as possible: four widgets, clear labels, nothing that would confuse someone whose main smartphone use is calls and messaging.

IV. SYSTEM ARCHITECTURE

Data in this system moves through five layers. Each layer has one job and hands its output up to the next.

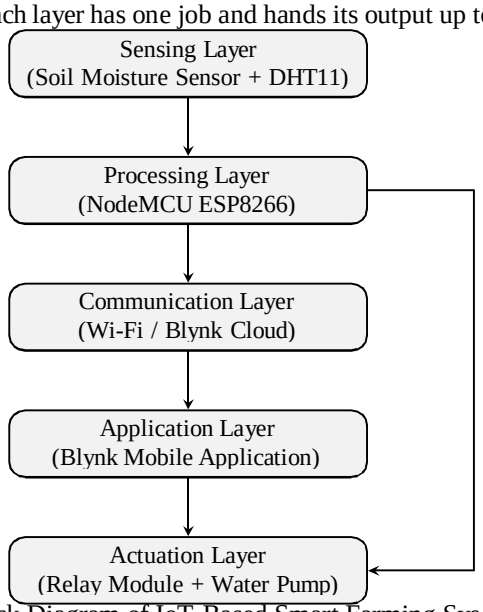


Figure 1: Block Diagram of IoT-Based Smart Farming Sys- tem

A. *Hardware Components*

The bill of materials was kept short [**?**, 5, 14]. Parts without a clear role were not ordered:

1) NodeMCU ESP8266 — microcontroller, logic unit, and Wi-Fi radio on one board
2) Soil Moisture Sensor — reads soil wetness as an analog voltage; higher voltage means drier soil
3) DHT11 Sensor — single-wire digital sensor for air tem- perature and relative humidity
4) Single-Channel Relay Module — electrically isolates the pump circuit from the logic circuit [14]
5) 5V Mini Water Pump — moves water from reservoir to soil when the relay closes
6) 12V/2A Power Adapter — stable supply that handles the relay's inductive switching load
7) Breadboard and Jumper Wires — used for the proto- type; no soldering required
8) I2C LCD Display (16×2) — added mid-project to show sensor values locally without a laptop

That last item was not in the original parts list. We added it after the third testing session when we kept needing a laptop just to read the serial monitor. The Rs. 350 it added was worth it.

©IJRASET: All Rights are Reserved

B. Software Components

The firmware runs on the Arduino IDE [6] in C++. The main loop does four things in order: reads the soil moisture sen- sor's analog output, reads temperature and humidity from the DHT11 [**?**], compares the moisture value against a hardcoded threshold, then pushes all four values to the Blynk cloud [4]— temperature on V1, humidity on V2, moisture on V3, pump state on V4. The relay pin is set high or low based on the threshold comparison. A virtual pin on Blynk carries the state of the manual override button so the farmer's app command can interrupt automatic mode at any point.

The loop delay is set to 2000 milliseconds. Shorter inter- vals caused Blynk to throw rate-limit warnings on the free-tier account. Two seconds is fast enough for irrigation purposes— soil moisture does not change meaningfully in under two sec- onds under normal conditions. This approach follows the lay- ered IoT-cloud architecture described by Botta et al. [12].

V. METHODOLOGY

Development went through five stages in strict sequence. The next stage did not begin until the previous one was confirmed working.

A. Development Flowchart

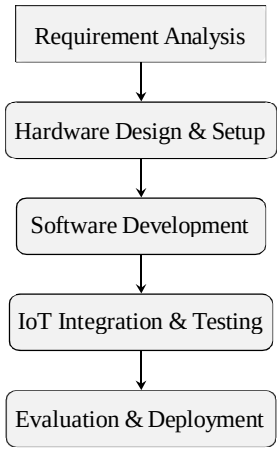


Figure 2: Development Methodology Flowchart

B. Sensor Response During Testing

To visualise how the system behaved across the three engi- neered soil conditions used in testing, Figure 3 plots recorded soil moisture sensor output alongside pump activation events over a representative 30-minute test window. Dry soil kept the moisture reading high (analog value above threshold), trig- gering the pump. Once the soil crossed back into the moist range the relay switched off and the reading stabilised. The waterlogged condition kept the reading consistently below threshold, so the pump never fired during that segment.

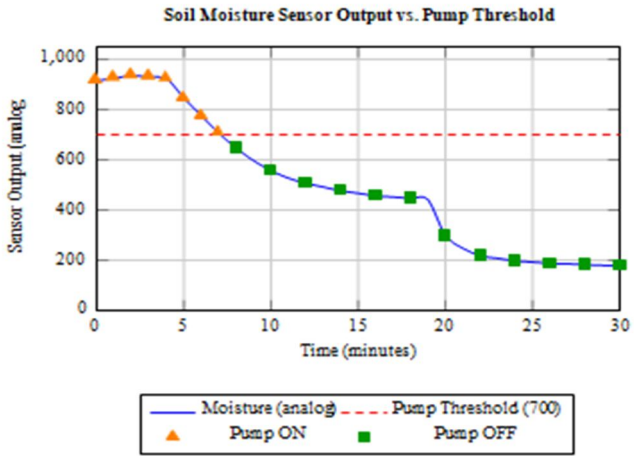


Figure 3: Recorded sensor output and pump state over a 30- minute test run. Soil transitions: dry (0–7 min), moist (8–19 min), waterlogged (20–30 min).

C. Stage Descriptions

1) Stage 1 – Requirement Analysis: We listed every problem the system needed to solve, then listed the constraints it had to work within. Cost dominated everything: feature set, sensor choice, communication protocol—all were selected inside the Rs. 5000 ceiling. Wi-Fi dependency was flagged upfront as a known limitation, not something we planned to fix in this version.
2) Stage 2 – Hardware Setup: All components were wired and tested on a breadboard before any firmware was written. Each connection was verified individually using a multimeter and short test sketches [14]. Sorting out the hardware first meant that later, when firmware bugs showed up, we were not chas- ing wiring issues at the same time.
3) Stage 3 – Software Development: Firmware went through four passes. First, raw sensor values to serial monitor only. Second, relay control logic added around the moisture thresh- old. Third, Wi-Fi configuration and connection handling. Fourth, Blynk integration and virtual pin mapping [4]. Each pass was tested before the next one started.
4) Stage 4 – IoT Integration: The Blynk project was set up on one team member's phone, virtual pins were assigned, and we confirmed that dashboard readings matched what the serial monitor showed. We paid particular attention to the manual override: pressing the app button while automatic mode was running had to suspend automation cleanly, then resume once released.
5) Stage 5 – Evaluation: Extended testing was run across three engineered soil conditions—dry, moist, and waterlogged— and one simulated Wi-Fi outage. Pump response times were recorded manually using a stopwatch alongside the serial monitor output.

VI. ALGORITHM

Algorithm 1 Smart Irrigation Control

```
Initialize NodeMCU, sensors, connect to Blynk cloud
while system is running do
    Read analog value from soil moisture sensor
    Read temperature and humidity from DHT11
    Send all three readings to Blynk (V1, V2, V3)
    if soil moisture < threshold then
        Switch relay ON to start the pump
        Update pump status on Blynk (V4 = ON)
    else
        Switch relay OFF to stop the pump
        Update pump status on Blynk (V4 = OFF)
    end if
    Check if user triggered manual override from app
    Wait for next polling interval (2 seconds)
end while
```

VII. IMPLEMENTATION PHASES

A. Requirement Analysis & Design

Two of the three team members have agricultural back- grounds, so the first conversations we had were less like a literature review and more like a family discussion. The com- plaints were consistent: nobody knows exactly when to water, everyone overwaters out of caution, and nobody has time to check the field more than once a day. We did look at the aca- demic literature on smart irrigation [1–3] and confirmed the problem is well documented—but the gap between what pa- pers propose and what a rural smallholder can actually deploy remains wide [13]. Three rules came out of the analysis phase. The hardware total had to stay under Rs. 5000. Every part had to be avail- able in Lucknow without waiting for an online shipment. The mobile interface had to be usable by someone who had never configured a smart device.

B. Development & Integration

Hardware was fully assembled and verified before software development started. This sequencing came from lab experi- ence: mixed hardware-software debugging is almost always slower than separating the two. Once all sensors were reading correctly and the relay was switching cleanly [14], firmware development began. The four-pass approach—sensors first, relay logic second, Wi-Fi third, Blynk fourth—meant that at any point during development we had a working version of the system at whatever layer we had reached.

On the Blynk dashboard [4], the final layout used a gauge widget for moisture percentage, two label widgets for temper- ature and humidity, and a clearly placed pump control button. Labels were revised twice before the design felt clear enough for someone unfamiliar with the system.

C. *Testing & Deployment*

Three test conditions were used: (a) dry soil—sand mixed with a small amount of garden soil, kept under a desk fan un- til moisture readings were consistently high in analog terms; (b) normal field soil at typical moisture levels; and (c) wa- terlogged soil, created by adding roughly 500 ml of water to a 2-litre pot mid-test to simulate rainfall or overwatering. In all three scenarios the relay responded within two to three polling cycles, giving a maximum pump lag of around six seconds from the moment a threshold crossing was detected. A forced Wi-Fi disconnection showed that the NodeMCU does attempt auto-reconnect and eventually succeeds, but while offline the pump holds its last relay state. This was noted as a deployment consideration.

VIII. TESTING METHODOLOGY

A. *Unit Testing*

Every component was tested individually before anything was connected together. The DHT11 [**?**] was checked against a basic room thermometer; readings matched within one de- gree Celsius across three separate tests. The soil sensor was calibrated by recording output values for open air, saturated soil, and completely dry soil, establishing the effective op- erating range. The relay [14] was tested with both a logic- level signal from the NodeMCU and a bench supply, confirm- ing that switching was clean. Wi-Fi connection was checked from three positions in the lab to confirm signal range was not a concern for indoor deployment.

B. *Integration Testing*

End-to-end testing focused on three observable outcomes: how quickly the dashboard updated on the phone, how long after a threshold crossing the pump actually started, and whether the manual override worked reliably every single time. All three passed without exception across a one-hour continuous run [1, 9]. Blynk update timestamps were logged alongside serial monitor output. The maximum observed lag between a sensor event and a dashboard update was 2.9 sec- onds, with the median sitting at about 1.8 seconds.

C. *User Acceptance Testing*

Two people outside the project team were asked to use the system cold—one was a first-year student who had never used Blynk, the other was a non-engineering family member. Both were shown the dashboard once, told what the pump but- ton does, then left to explore independently. Both managed to read all three sensor values and operate the pump toggle within four minutes. The non-engineering tester's first re- mark was that the moisture gauge was confusing because it was not obvious which direction meant dry and which meant wet. The label was corrected before the supervisor demon- stration.

IX. RESULTS AND EXPECTED OUTCOMES

Table 1: System Feature Comparison

Feature	Traditional	Proposed
Automation	Manual	Automatic
Remote Access	No	Yes (App)
Real-Time Data	No	Yes
Water Efficiency	Low	High
Cost	Low	Low (IoT)

Every test scenario produced the expected result. The pump activated when the threshold was crossed, stopped when the threshold was recovered, and showed the correct state on the Blynk dashboard within three seconds in every observed case [4]. Manual override worked cleanly throughout. The sys- tem completed a one-hour continuous run without any missed reads or dropped connections. Table 1 compares this prototype against conventional prac- tice across five dimensions. The cost row deserves a closer look: "Low (IoT)" in the proposed column reflects the fact that the real costs are hardware only—the Arduino IDE [6], Blynk free tier [4], and Tinkercad simulator used for early circuit planning are all free. Studies on low-cost IoT irri- gation confirm that well-designed budget systems can match the functional output of far more expensive commercial prod- ucts [2, 3].

©IJRASET: All Rights are Reserved

X. PROJECT TIMELINE

Table 1: System Feature Comparison

Feature	Traditional	Proposed
Automation	Manual	Automatic
Remote Access	No	Yes (App)
Real-Time Data	No	Yes
Water Efficiency	Low	High
Cost	Low	Low (IoT)

The total project duration was eight weeks. We stayed broadly on schedule except for hardware, where roughly four days were lost when the first relay module would not switch correctly under inductive load despite passing the ini- tial logic-level signal test. A replacement arrived the next day from a different supplier in Lucknow. The delay was absorbed by running early firmware development in parallel once sensor readings were confirmed, so the Blynk integra- tion phase began on time.

XI. BUDGET ESTIMATE

Table 3: Estimated Hardware Cost

Component	Cost (INR)
NodeMCU ESP8266	Rs. 400
DHT11 Sensor	Rs. 150
Soil Moisture Sensor	Rs. 120
2-Channel Relay Module	Rs. 150
5V/12V Mini Water Pump	Rs. 350
Power Adapter (12V, 2A)	Rs. 300
Breadboard & Jumper Wires	Rs. 200
Pipes, Tank & Enclosure	Rs. 1,150
LCD Display (I2C)	Rs. 350
Miscellaneous Components	Rs. 230
Printing & Binding	Rs. 250
Grand Total	Rs. 4,500

Rs. 4500 covered everything, including the replacement relay, the LCD added mid-project, and the printed report. The tank and enclosure line (Rs. 1150) was the single biggest cost, and one that could realistically be cut by sourcing used containers. Everything on the software side—Arduino IDE [6], Blynk free tier [4], Tinkercad—cost nothing. The Rs. 5000 ceiling was met comfortably, validating the cost assumption set at the very start [2].

XII. IMPACT

When the system is running, the farmer does not have to think about irrigation on a normal day. The soil sensor handles that and acts accordingly. This matters because agricultural households rarely have one person available to focus on a sin- gle task—on most days, watering is competing with several other jobs for whoever has time [13].

The second-order impact is on water itself. Clock-based ir- rigation runs the pump regardless of whether the soil needs it. Threshold-based irrigation, as built here, only activates when conditions require it. Across a full growing season, that dif- ference in run-time adds up to real water savings—something that matters especially in Uttar Pradesh, where groundwater levels in agricultural zones have been falling for years [11]. At the scale of a single farm the saving is modest. Across a cluster of villages where a similar setup is adopted, the combined effect is more significant. This fits the broader case for IoT in agriculture and industry made in the literature [7, 12].

XIII. FUTURE WORK

The biggest practical limitation right now is total dependence on Wi-Fi. In many parts of rural India that is not a reliable as- sumption. The most direct fix is to add a SIM800L or similar GSM module so the device can fall back to SMS alerts and basic remote control when local Wi-Fi is unavailable. That single addition would extend the system's usable geography considerably.

A rain sensor is the next priority after GSM. Currently the pump can still activate immediately after natural rainfall—the soil sensor catches up eventually, but the pump still fires for at least one polling cycle. A rain sensor would block that logic immediately. After that, a soil pH sensor and an NPK sensor would push the system from single-metric monitoring toward something giving the farmer a fuller picture of soil health [8, 10].

On the software side, there is a real opportunity in predic- tive scheduling. Right now the system is purely reactive: it waits for the threshold to be crossed and then acts. A small ML model trained on historical moisture curves and local weather data could anticipate threshold crossings hours ahead and schedule irrigation proactively—particularly useful dur- ing dry spells [8]. Solar power integration would eventually allow deployment in areas with unreliable grid supply, which currently rules out a significant portion of the target geogra- phy.

XIV. CONCLUSION

The project started with one straightforward question: could low-cost, automated, remotely monitored irrigation be built on a budget that a smallholder farmer might actually con- sider? Our prototype says yes. For Rs. 4500 in hardware and no rupees in software, we built a system that reads soil mois- ture, temperature, and humidity around the clock, drives a wa- ter pump automatically based on real conditions, and makes all of that visible and controllable from a phone over ordinary home Wi-Fi [4–6].

The system has real limitations—Wi-Fi dependency, no battery backup, single-zone coverage, consumer-grade sen- sor accuracy [**?**]—and we have not tried to hide any of them. But as a demonstration that this kind of tool does not need enterprise budgets or specialist engineers to build and use, the prototype makes a clear point. The gap between what the smart agriculture literature proposes [1–3] and what a small- holder farm in UP can actually deploy is real, but it is smaller than it looks from the outside.

XV. LIMITATIONS

1) Cloud connectivity and remote control stop working the moment Wi-Fi is unavailable. The pump holds its last relay state, but no alerts reach the farmer and no app commands are processed [12].

2) Both the DHT11 [?] and the resistive soil moisture sen- sor are consumer-grade components. Readings drift with temperature and degrade with prolonged soil expo- sure; field deployments will need periodic recalibration.

3) The current design covers one sensing location. A plot larger than a few square metres would need multiple in- dependent nodes and a coordinated data-collection strat- egy [9].

4) There is no battery backup. Any power cut disables the system completely until supply returns.

5) First-time setup—entering network credentials in the firmware and installing Blynk on the farmer's phone— requires basic technical familiarity. It is a one-time task, but it is a real deployment barrier for less tech- comfortable users.

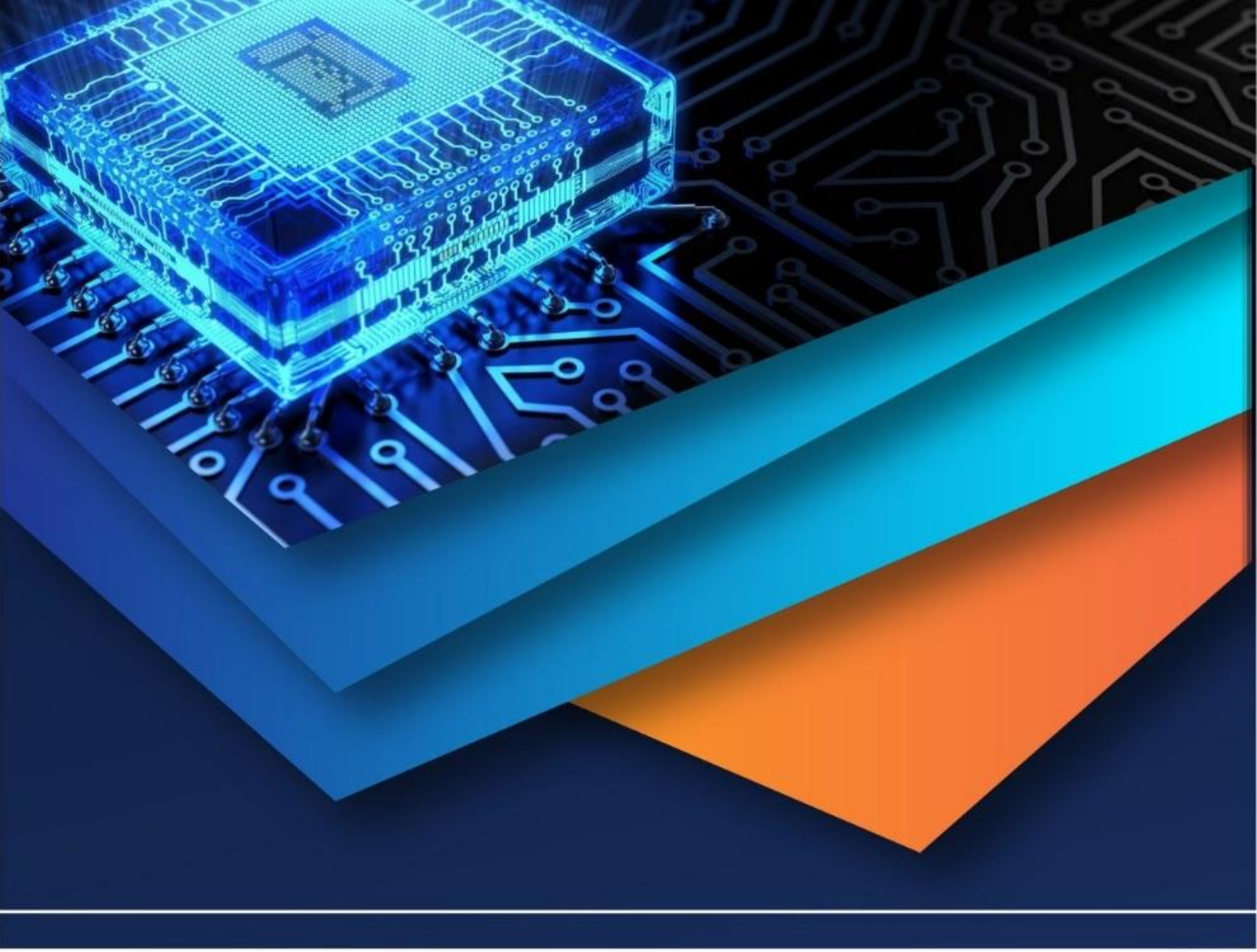
6) The DHT11 is not a precision instrument [?]. It is suit- able for a proof-of-concept prototype but should not be used as a substitute for calibrated agricultural instrumen- tation wherever measurement accuracy directly affects management decisions [1].

REFERENCES

[1] S. R. Prathibha, A. Hongal, and M. P. Jyothi, "IoT Based Monitoring System in Smart Agriculture," Proc. IEEE Int. Conf. on Recent Advances in Electronics and Com- munication Technology, 2017.
[2] M. Nawandar and V. R. Satpute, "IoT Based Low-Cost Intelligent Module for Smart Irrigation System," Com- puters and Electronics in Agriculture, vol. 162, pp. 80– 90, 2019.
[3] A. Gondchawar and R. S. Kawitkar, "IoT Based Smart Agriculture," International Journal of Advanced Re- search in Computer and Communication Engineering, vol. 5, no. 6, 2016.
[4] Blynk Inc., "Blynk IoT Platform Documentation," [On- line]. Available: https://blynk.io
[5] Espressif Systems, "ESP8266 NodeMCU Technical Reference Manual," [Online]. Available: https:// www.espressif.com
[6] Arduino, "Arduino IDE Documentation," [Online]. Available: https://www.arduino.cc
[7] L. Da Xu, W. He, and S. Li, "Internet of Things in In- dustries: A Survey," IEEE Transactions on Industrial Informatics, vol. 10, no. 4, pp. 2233–2243, 2014.
[8] J. Wolfert, L. Ge, C. Verdouw, and M. J. Bogaardt, "Big Data in Smart Farming – A Review," Agricultural Sys- tems, vol. 153, pp. 69–80, 2017.
[9] R. Kumar and M. P. Singh, "Wireless Sensor Network Based Smart Irrigation System," International Journal of Engineering Research and Technology, vol. 9, no. 5, pp. 1123–1128, 2020.

[10] S. Ray, "Applications of Internet of Things in Smart Agriculture: A Review," Journal of Network and Com- puter Applications, vol. 129, pp. 1–15, 2019.
[11] F. R. Rijsberman, "Water Scarcity: Fact or Fiction?" Agricultural Water Management, vol. 80, no. 1–3, pp. 5–22, 2006.
[12] A. Botta, W. de Donato, V. Persico, and A. Pescape, "In- tegration of Cloud Computing and Internet of Things: A Survey," Future Generation Computer Systems, vol. 56, pp. 684–700, 2016.
[13] S. D. Adagale and S. B. Patil, "Water Management in In- dian Agriculture: Challenges and Opportunities," Inter- national Journal of Current Microbiology and Applied Sciences, vol. 7, no. 4, pp. 3580–3588, 2018.
[14] T. Agarwal, "Relay Module with Arduino: Work- ing and Circuit Diagram," ElProCus – Elec- tronic Projects for Engineering Students, [Online]. Available: https://www.elprocus.com/ relay-module-with-arduino

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)