IoT-Based Real-Time Data Monitoring of a Green House Farm with Offline Monitoring Capability

Tushar Bhadra
Department of EEE, Faculty of Engineering
American International University-Bangladesh
Dhaka, Bangladesh
tbhadrarahul08@gmail.com

Mujakkir-Ul-Islam
Department of EEE, Faculty of Engineering
American International University-Bangladesh
Dhaka, Bangladesh
mujakkirulislammehrab@gmail.com

Nowshin Alam
Department of EEE, Faculty of Engineering
American International University-Bangladesh
Dhaka, Bangladesh
nowshin.alam@aiub.edu
https://orcid.org/0000-0003-4404-7444

***Abstract*—Greenhouses have transformed modern agriculture by precisely managing the environmental conditions of delicate plant species to promote crop growth. A conceptual and practical framework for a low-cost greenhouse farm monitoring system has been proposed in this paper to track real-time data on greenhouse farms with online and offline both monitoring capabilities. The Internet of Things (IoT) technology has been used to develop a system prototype, which uses two sensors to collect data on environmental factors, including temperature, humidity, heat index, and soil moisture. An ESP32 microcontroller module collects and sends this sensor data to a Raspberry Pi 4 microcomputer, which displays it on a web application. The application is hosted on a local Raspberry Pi server and built with Node.js, a JavaScript runtime environment. The communication between ESP32 and Raspberry Pi is established with REST API (Representational State Transfer Application Programming Interface), a way to connect applications and components in microservices architectures. A highly robust PostgreSQL-based database is used to store the data in Raspberry Pi, which allows the collection of farm data locally without having to connect to any web storage. With an internet connection, data can be monitored from anywhere worldwide using a global server made of Vercel, a cloud-based platform that can host websites and web applications. The local Raspberry pi server would be accessible to the internet through SSH (secure shell) tunneling with Ngrok, a cross-platform application.**

Keywords*—*IoT, Greenhouse, Precision Agriculture, Soil Moisture, Crop Monitoring, Wireless Sensor Networks, Raspberry Pi, ESP32, DHT-22, Node.js, REST, Vercel, SSH, Ngrok.

I. INTRODUCTION

Greenhouse agriculture, which refers to performing agriculture operations within a well-sealed transparent enclosure, helps cultivate plants requiring regulated climate conditions in less-than-ideal settings. The advancement of greenhouse farming techniques in recent years has resulted in more total food for the world and has aided in the reduction of world hunger. The traditional method of monitoring a greenhouse farm is a manual and inefficient process that requires a lot of time and effort on the farmers' part. Thus, integrating intelligent software-based monitoring into greenhouse farming is vital for increasing the productivity of the agriculture sector.

As environmental parameters need to be controlled in a greenhouse farm, Internet of Things (IoT) based monitoring systems utilizing sensors, microcontrollers, and networking modules are revolutionizing the agriculture industry. In this paper, such an IoT-based solution has been developed to facilitate the easy monitoring of greenhouses. As many areas of Bangladesh suffer from poor internet connectivity, the primary goal of this work is offline data collection and storage while also providing the user the option to access the data remotely through the internet. Monitoring data in real-time through a web application using a locally hosted database is desired. The authors propose a framework with an ESP32 microcontroller and Raspberry Pi 4 microcomputer. A DHT-22 sensor and a soil moisture sensor have been placed inside the greenhouse prototype to monitor soil moisture, humidity, temperature, and heat index measurements. All the sensor readings are viewable through a local web server and can be monitored through a global server from any location with an internet connection. The data collection will not be interrupted even if there is no internet, as the Raspberry Pi will keep storing the received data in the local database. The proposed system is expected to make the crop monitoring process easier for farm owners and help them make informed decisions. The sensors were calibrated using different environmental conditions to avoid any unusual false data and ensured the accuracy of measuring data even though there was a harsh environment inside the greenhouse. Authors expect to integrate more than one in quantity for each sensor following the sensor redundancy approach to ensure more data accuracy in the future. Calculating the average sensor values or median filtering would be a great way to avoid inconsistencies in readings. Here, the readings will be taken from only active sensors in case of any sensor failure.

In the future, the authors hope to incorporate automatic irrigation and ventilation control features using water reservoirs, pumps, exhaust fans, etc. The system can be programmed to use predefined thresholds or dynamic ranges for sensor values (e.g., soil moisture, temperature, and humidity) to trigger these processes automatically. For example, when soil moisture falls below a specified level, the system could activate irrigation pumps, or if the temperature exceeds a set threshold, exhaust fans could be triggered for ventilation. Data analysis is needed based on the environmental situation before the implementation of automatic features.

Additionally, the authors aim to integrate more features into the prototype to further aid the farmers in monitoring crop conditions, such as the prediction of data with the help of machine learning algorithms. Predictive analytics of exported raw data are possible from this system, and use of machine learning can be applied to enhance data-driven decisions, such as weather forecasting and irrigation prediction. Algorithms like linear regression for basic forecasting and time series models like ARIMA would be great choices in that case.

This system is also scalable for bigger greenhouse operations when multiple sensors are used. Here, authors may need a bigger number of microcontrollers (ESP32), as there are limitations to the number of digital input pins per microcontroller. In that case, multiple microcontrollers will stream sensor data wirelessly to the Raspberry Pi in a similar way as the current prototype setup. The data storage is easily scalable as the same Pi server can still be used, though there are some limitations in terms of database scalability if more sensors and microcontrollers are added within this system. The limited resources of Raspberry Pi may not be able to handle large-scale databases. In that case, routine monitoring of database performance and data archiving after a specific time based on the performance results would improve database scalability and allow the system to operate smoothly.

In this paper, the authors provide an overview of the project and discuss its future scope in this introduction section. The methodology section showcases the entire process of the workflow, while the result section discusses the outcomes of the project, including a cost analysis before concluding the article.

II. Literature Review

From studying the existing literature about real time agricultural monitoring, several IoT based initiatives to improve greenhouse agriculture for farmers can be observed. Abhiram, Kuppili, and Manga [1] proposed a NodeMCU-based IoT-enabled smart agriculture system in Hyderabad where ESP8266 microcontroller and DHT-11, soil moisture and rain detection sensors were utilized for monitoring and Blink server was used for storage and analysis of farm data. Wan, Song and Cao in [2] also developed a greenhouse farm where NodeMCU and ESP8266 were used. They chose low-cost components like DHT-11, Soil Moisture, photo resistance, and MQ-135 gas sensors. Photo resistance sensors monitored the greenhouse's illumination and CO_2 levels. Seethalakshmi et. al's team developed an automatic irrigation system for a greenhouse using IoT [3] where soil moisture sensor and DHT-11 sensor were used to monitor environmental parameters. They used a cloud server connected with a mobile app to see the dashboard where all sensor values and the motor state are monitored.

A smart greenhouse with an automatic drip irrigation facility utilizing the concept of IoT was developed by Kodali, Jain, and Karagwal [4]. Proper water management tanks were built where an ultrasonic sensor was placed for water level monitoring. The environmental parameters were also monitored through soil moisture sensor and DHT-11 sensor. A GSM module was used to monitor the tube well activity, and the readings collected from storage containers were uploaded to the cloud service. On the other hand, Türk, Gunal, and Gurel [5] advised the use of a do-it-yourself (DIY) platform for greenhouse automation. Their project needed a smartphone, embedded device, and web server, including an SQL-based database. Raspberry Pi was used for hosting the database that was required in their project for data storage. Arduino UNO, having an ethernet shield, was the main microcontroller to operate the sensors.

In [6], wireless sensor networks were employed to remotely monitor greenhouse conditions with the help of NRF24L01 sensor network transceiver module coupled with Arduino Mega and DHT-11 sensor. Even when away from the farm, a user could receive messages about greenhouse conditions sent by a GSM module. Shah et. al's work [7] also utilized Arduino as the main microcontroller and offered similar GSM message updates, but it also included an option to livestream the greenhouse condition with the help of a ESP32 webcam. Additionally, the automatic system consisting of fan, heater, water pump, and artificial light system operated based on soil moisture, DHT-11, LDR, and flame sensor readings. Alam [8] proposed a conceptual framework for a IoT based smart farming system utilizing Wifi, LoRa and GSM protocols and consisting of multiple ESP32 microcontrollers, a Raspberry Pi for hosting a local database and a python-based web application. In order to effectively optimize the use of agricultural resources and improve crop yields, Shinde [9] suggested an IoT-based crop monitoring system that combines sensors and microcontrollers to automate irrigation and fertilizer delivery and provide real-time information on soil moisture, temperature, and humidity. Lastly, Ruchi [10] developed an IoT-based smart agricultural monitoring system featuring automatic roof shading, smart irrigation, and real-time monitoring of environmental parameters such as soil moisture, temperature, and humidity, aimed at reducing labor costs and improving agricultural efficiency.

III. Methodology

The data monitoring system proposed by this paper is created by merging various hardware and software technologies into it. The following are the primary components that have been implemented in the project prototype.

- Soil Moisture Sensor
- DHT-22 sensor
- ESP-32
- Raspberry Pi 4 (4 GB RAM)
- RaspAp
- Node.js
- GitHub
- Vercel
- Ngrok
- DBeaver

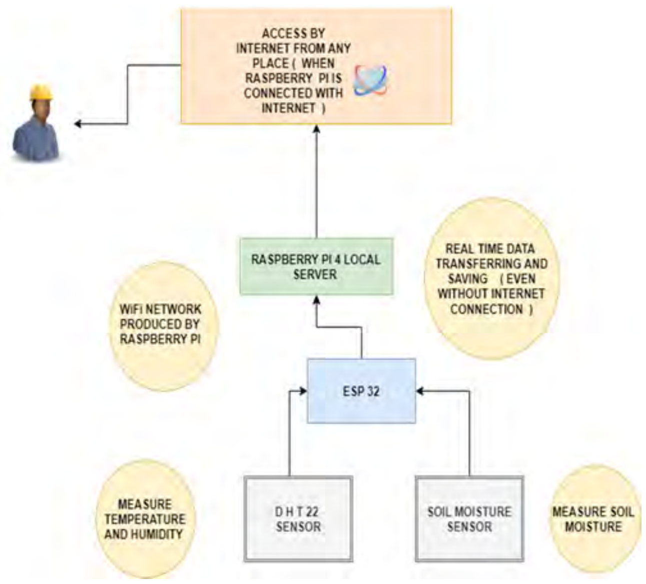

ACCESS BY INTERNET FROM ANY PLACE (WHEN RASPBERRY PI IS CONNECTED WITH INTERNET)
RASPBERRY PI 4 LOCAL SERVER
REAL TIME DATA TRANSFERRING AND SAVING (EVEN WITHOUT INTERNET CONNECTION)
WIFI NETWORK PRODUCED BY RASPBERRY PI
ESP 32
MEASURE TEMPERATURE AND HUMIDITY
D H T 22 SENSOR
SOIL MOISTURE SENSOR
MEASURE SOIL MOISTURE

Fig. 1. Block diagram of the monitoring system.

The working procedure of the complete system can be expressed using the block diagram shown in Fig. 1. The ESP32 module is responsible for the operation of the sensors, which detect temperature, humidity, and soil moisture of the greenhouse. The local server on the Raspberry Pi wirelessly receives data that was collected from sensors by the ESP32 module. RaspAp software has been used to provide a wireless network through Raspberry Pi 4, individuals are able to connect their own devices to the local network. Due to the presence of Raspberry Pi's hotspot capability, a Wi-Fi router is not essential for the establishment of a network. So, data can be transferred even without an internet connection. When the Raspberry Pi 4 is connected to the internet, the global server will be activated and can be accessed from anywhere in the world. A web server has been built in Raspberry Pi 4 which is accessible by the user in both local and global networks.

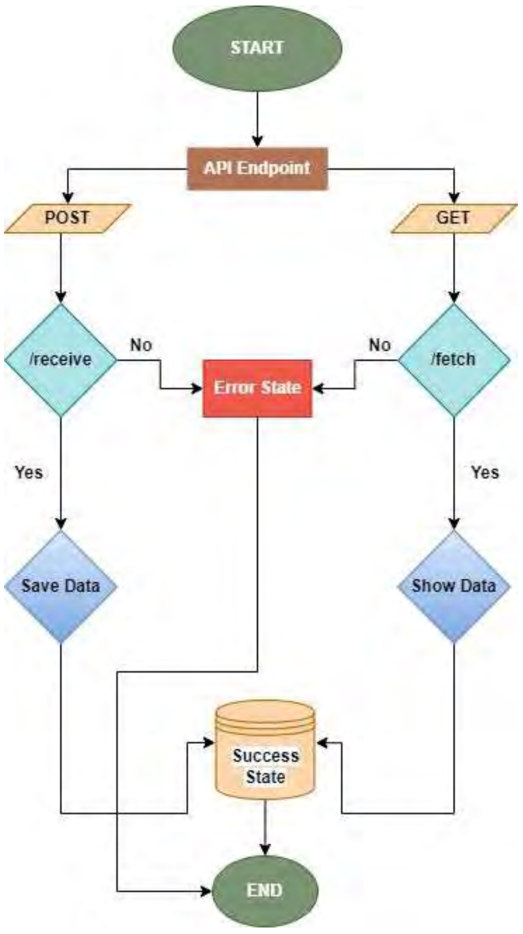


Fig. 2. Flow Chart of the data transfer process.

The data transferring process of the framework implemented in this work is described by the flowchart in Fig. 2. Data transmission occurs from ESP32 periodically after selected time intervals to the API (Application Programing Interface) endpoints. API is a mechanism that can be said to be a way for two or more programs to communicate with each other. API endpoints are of two types following two different methods. One is receiving, following the post method, and another one is fetching, following the get method. Post method refers to the backend of the server, which is actually used for sending data to the server. Fetch is used to request to the server and load the information on webpages. The method, Get used in fetching refers to the front end of the server which is responsible for showing the data to a server applying the method of HTTP. That means with a URL we can browse the server for data.

The overall process has been done through Node.js technology which is a JavaScript runtime environment. React JS framework has been used for the front end and the Express JS framework has been used for the back end. Most importantly, communication between ESP32 and Raspberry Pi has been established using REST API. REST API is basically responsible for data streaming here, and it is usually used to create web services that exchange data between client and server applications.

To get access to this server globally, another process has been followed. All the backend and frontend codes have been pushed to GitHub after opening a GitHub account. GitHub is a platform where any kind of project code can be stored. The GitHub repository has been uploaded or can be said deployed into the Vercel application. The Vercel application hosted the front end of the project which is visualized to the users through a global server with an internet connection. The data are accessible with the help of Ngrok which is a cross-platform application that exposes local server ports to the internet, in order to be shown globally or to be accessed from anywhere. Ngrok actually provides the link to get access globally with the Vercel application. Ngrok simply mirrors the constantly updating data in the locally hosted Raspberry Pi server via SSH (secure shell) tunneling, a method for securely routing traffic. Communication is encrypted using HTTPS to secure data transmission. Leveraging HTTPS ensures the confidentiality, authenticity and integrity of data as it provides an extra layer of protection on top of SSH tunneling.

All the transferred data has been stored in the Raspberry Pi database, which has been made of a PostgreSQL database management system. It is a relational database management system connected by Prisma with the back end. Prisma is an ORM that acts as a helping tool for saving and updating data into a database, reducing the complexity of debugging SQL queries.

All database information can be shown through DBeaver software. DBeaver is a free, open source, graphical database management tool for database developers which is used to create and manage databases across a wide range of database management systems (DBMSs) and works with most of the popular DBMSs, such as MySQL, PostgreSQL, MariaDB, SQLite, Oracle. This GUI based management tool displays the objects of the database, can create visual diagram of the database objects and whole schemas and can be exported according to the requirement.

IV. RESULTS

The interconnection between ESP32 and the different type of sensors and the necessary power connections to ESP32 and Raspberry Pi in the implemented circuit are shown in Fig. 3.

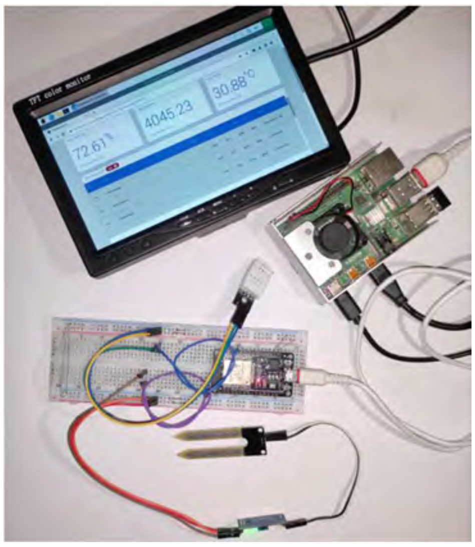


Fig. 3. Implemented Circuit.

A polycarbonate-enclosed greenhouse prototype is created in this project (Fig. 4). The sensors have been placed inside the greenhouse farm. As shown in Fig. 5, a demo login interface has been developed for the client who wishes to monitor the greenhouse data. The client will have a unique client id, and get access to global and local servers through submitting that id.

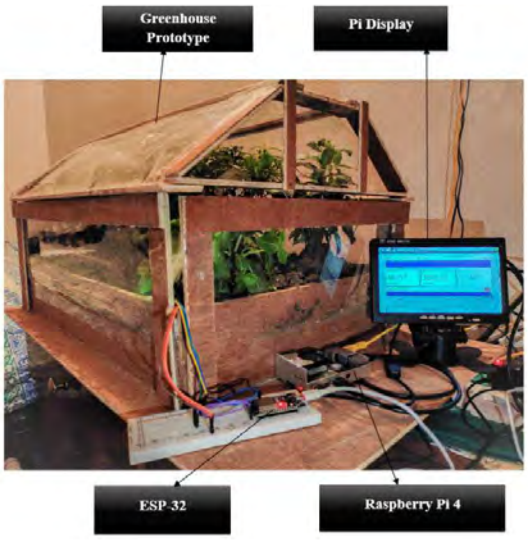


Fig. 4. Implemented full hardware.

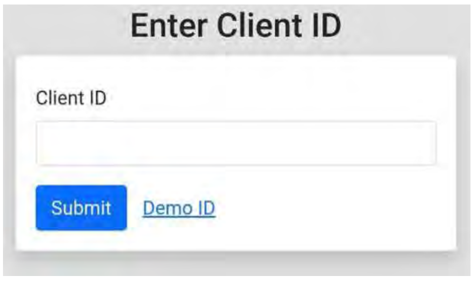
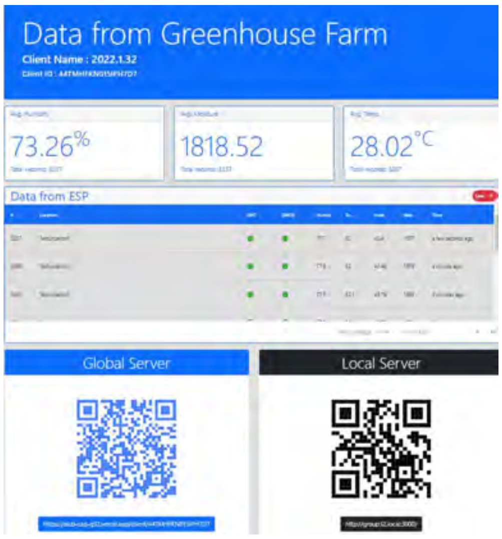


Fig. 5. Login interface prototype.



Data from Greenhouse Farm

Client Name : 2022.1.32

73.26%

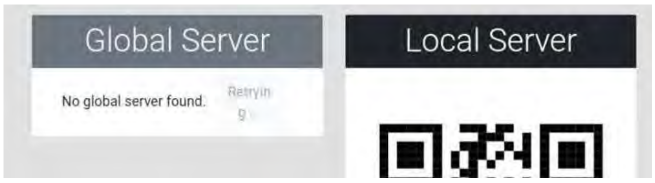
1818.52

28.02°C

Data from ESP

Fig. 6. User Interface Prototype.

On the user interface of the webpage (Fig. 6), DHT-22 and soil moisture sensor values are visible after logging in. The upper portion of this webpage shows the average values of the environmental parameters, which are based on records. The middle part seems like a table displaying whether the sensor is on or off and the updated value of sensors as well. After every minute, the value of the sensors is updated. This portion displays ten values at a time though it is possible to check all of the previous values by clicking the right arrow under the table. In the lower part, two QR codes of URL are visible on both global and local servers. Both links can be accessed by scanning from the Raspberry Pi monitor through mobile phones.

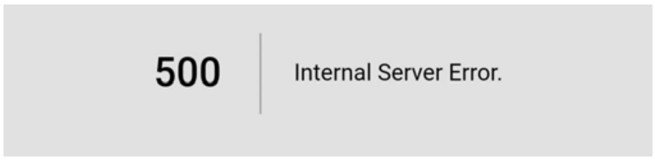


Global Server

No global server found. Retrying

Local Server

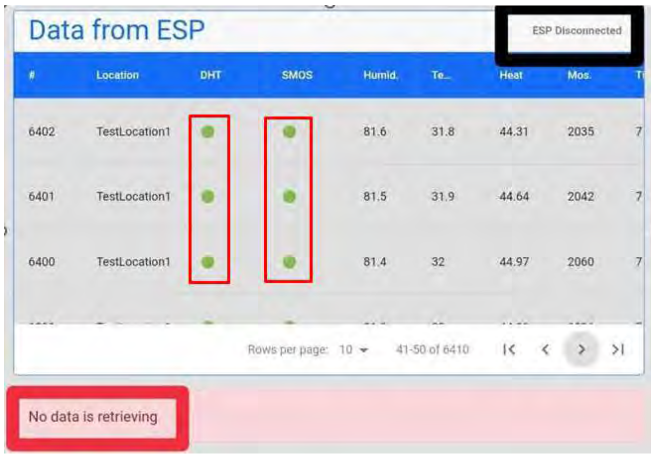
Fig. 7. QR code of global server lost visibility on webpage.



500 | Internal Server Error.

Fig. 8. Global Server is in error state.

Fig. 7 and Fig. 8 shows how the webpage notifies the user about any disruption of the internet. The QR code of the global server will not be visible on the web page while Raspberry Pi 4 is disconnected from the internet. The local server can still be accessed, and the webpage remains visible in the local network. But the global server cannot be accessible in this case, and the webpage will display an error code 500.



Data from ESP

ESP Disconnected

#	Location	DHT	SMOS	Humid.	Te...	Heat	Mos.	T
6402	TestLocation1	●	●	81.6	31.8	44.31	2035	7
6401	TestLocation1	●	●	81.5	31.9	44.64	2042	7
6400	TestLocation1	●	●	81.4	32	44.97	2060	7

Rows per page: 10 ▾ 41-50 of 6410

No data is retrieving

Fig. 9. Notifying client about circuit problem.

As seen in Fig. 9, a feature also has been added to the webpage to notify the client if there is any problem with the circuit when data is not being retrieved. This will happen if the data is not updated within two minutes, as the data should be updated every minute. The green dots in the third & fourth columns also indicate the active condition of the sensors.

Sensor values are saved in the Raspberry Pi database, which has been exported to a Microsoft Excel file in the CSV format using DBeaver tool. Here approximately 150 data are used for graphical analysis.

In the graph shown in Fig. 10, the percentage of soil moisture increased after watering on the farm which is indicating wet state, and after a while, the values started to return to their previous dry state. The graph in Fig. 11 shows the stability of DHT-22 sensor values. Temperature, humidity, and heat index remained stable as the greenhouse farm is an enclosed area.

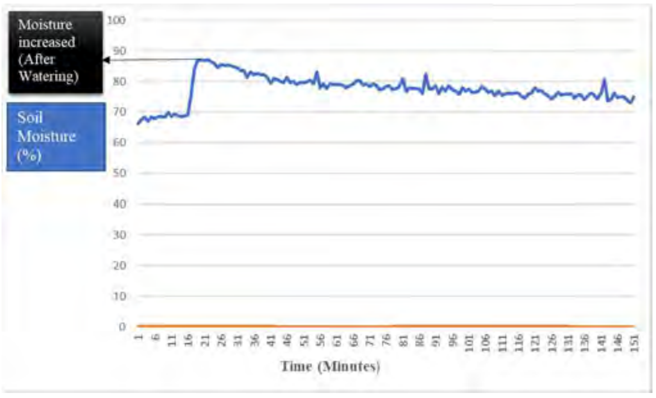


Fig. 10. Analytical graph of soil moisture sensor values.

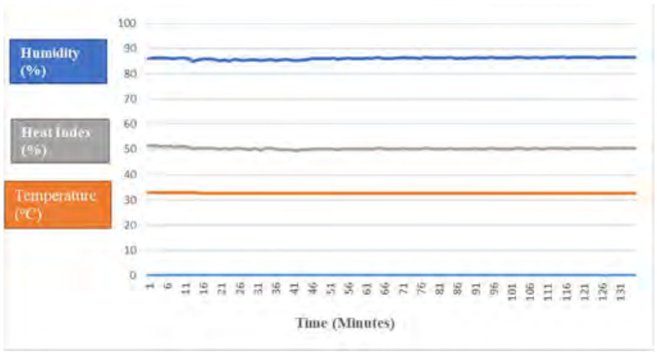


Fig. 11. Analytical Graph of DHT-22 Sensor Values.

The authors aimed to develop a cost-effective solution. The cost analysis for this project, detailing various components and their respective costs, is presented in the following section.

TABLE I. Cost Analysis

SL.	Item	Quantity	Estimated Price	Final Cost	Cost Deviation
1	ESP-32 Module	1	500৳	400৳	-100৳
2	DHT-22 Sensor	1	300৳	220৳	-80৳
3	Soil Moisture Sensor	1	100৳	80৳	-20৳
4	Raspberry Pi 4	1	15,000৳	17,000৳	2,000৳
5	Pi 7 inch Monitor (Optional)	1	3,500৳	3,000৳	-500৳
6	USB Cable B type	1	100৳	100৳	0৳
7	Connecting Wires	3 (Set)	300৳	150৳	-150৳
8	Breadboard	1	150৳	100৳	-50৳
Total cost			19,950৳	21,050৳	1,100৳

The final cost can be considered affordable compared to the prices of locally available commercial products, as the price of even entry-level solutions starts from around 50,000 BDT. The addition of sensors and microcontrollers could increase, but it would still be a great solution in terms of cost.

V. Conclusion

After the hardware implementation of the system prototype, the main objective of reliably tracking the greenhouse data has been accomplished. The developed device monitors data in real-time, and the data gets updated every minute with the option to change the data recording frequency. The system consistently stores data in the database hosted by Raspberry Pi, and both the current values and previous data log can be accessed by the user. The client can access the data locally offline, as well as remotely through an internet connection using an assigned client ID. If the ESP32 module becomes disconnected from the system for some reason, the owner will be notified through both global and local servers. In conclusion, it can be said that the IoT-based real-time data monitoring system discussed in this paper presents a viable alternative to previous cloud-based monitoring systems by circumventing the need for an internet connection and improving data security through the use of a locally hosted database. The choice of open-source software makes the system cost-efficient and allows superior scalability, so more features can be added in the future for a smoother monitoring experience for the clients.

References

[1] M. S. D. Abhiram, J. Kuppili, and N. A. Manga, "Smart Farming System using IoT for Efficient Crop Growth," in *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, Feb. 2020, pp. 1–4. doi: 10.1109/SCEECS48394.2020.147.

[2] Z. Wan, Y. Song, and Z. Cao, "Environment Dynamic Monitoring and Remote Control of Greenhouse with ESP8266 NodeMCU," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Mar. 2019, pp. 377–382. doi: 10.1109/ITNEC.2019.8729519.

[3] E. Seethalakshmi, M. Shunmugam, R. Pavaiyarkarasi, S. Joseph, and J. Edward paulraj, "An automated irrigation system for optimized greenhouse using IoT," *Mater. Today Proc.*, Feb. 2021, doi: 10.1016/j.matpr.2020.12.636.

[4] R. K. Kodali, V. Jain, and S. Karagwal, "IoT based smart greenhouse," in *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, Dec. 2016, pp. 1–6. doi: 10.1109/R10-HTC.2016.7906846.

[5] A. M. Türk, E. S. Gunal, and U. Gurel, "An Automation System Design for Greenhouses by Using DIY platforms," in *International Conference On Science, Ecology And Technology (ICONSETE 2015)*, Aug. 2015, p. 11.

[6] S. B. Joseph, E. G. Dada, E. P. Musa, M. S. Abdullahi, and S. F. Unekwu, "Environmental Parameters Monitoring for Greenhouse Farming Using Wireless Sensor Networks," *NIPES J. Sci. Technol. Res.*, vol. 2, no. 3, p. 218, Aug. 2020, doi: 10.37933/nipes/2.3.2020.23.

[7] R. Shah, M. Inamdar, S. Nalawade, S. Mujawar, and R. Sonkamble, "Automated Monitoring and Controlling of Greenhouse," *IRJET*, vol. 7, no. 3, p. 10, 2020.

[8] N. Alam, "Opportunity Assessment and Feasibility Study of IoT based Smart Farming in Bangladesh for Meeting Sustainable Development Goals," in *2021 International Conference on 4th Industrial Revolution and Beyond (IC4IR2021)*, Dec. 2021.

[9] S. Shinde, T. Dhanurkar, V. Jain, V. Pal, and P. Taware, "Crop Monitoring using IoT for Precision Agriculture," in *2024 4th International Conference on Pervasive Computing and Social Networking (ICPCSN)*, Salem, India: IEEE, May 2024, pp. 706–710. doi: 10.1109/ICPCSN62568.2024.00118.

[10] Ruchi, V. Wasson, Muskan, and Gargi, "IoT-Based Smart Control System for Monitoring Agriculture," in *2023 International Conference on Advanced Computing & Communication Technologies (ICACCTech)*, Banur, India: IEEE, Dec. 2023, pp. 385–390. doi: 10.1109/ICACCTech61146.2023.00070.