IoT based Industrial Information Managerial System for Industry 4.0

Arun C. A
Department of ECE
VelTech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology
Chennai, India
arunca.me@gmail.com

G. R. Kanagachidambaresan
Department of CSE
VelTech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology
Chennai, India
kanagachidambaresan@gmail.com

R. Sai Gowtham
Department of ECE
VelTech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology
Chennai, India
gowthamredrowthu01@gmail.com

Aditya Sharma
Department of ECE
VelTech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology
Chennai, India
addy24052002@gmail.com

G. S. Saddam Hussain
Department of ECE
VelTech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology
Chennai, India
saddamhusain19021@gmail.com

Janjitha V
Department of ECE
VelTech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology
Chennai, India
janjitha08@gmail.com

***Abstract*—The advent of Industry 4.0 has created a growing need for intelligent devices that can gather, store, and exchange data from various disciplines in industrial environments. Accurate real-time monitoring and analysis of data are crucial for realizing maximum productivity, reducing downtime, and facilitating predictive maintenance. To address these needs, a industrial information managerial system based on the ESP32 microcontroller were design and developed. The system communicates with industrial sensors of various types via the Modbus RS485 protocol to allow it to receive real-time data from various parts of an industrial environment. The system incorporates a Real-Time Clock (RTC) module for reliable date and time stamping and thus improved time-series data integrity. Furthermore, an micro-SD card module for local data storage is employed to provide data logging without interruption even in the event of network failure. This local backup mechanism provides integrity and accessibility of vital data for different operational scenarios. For increased remote access and monitoring, the ESP32 operates on a ThingsBoard cloud platform that features a Wi-Fi interface. Utilization of the MQTT protocol facilitates real-time data visualization, thereby enabling the monitoring of sensor data via an intuitive dashboard interface. Such integration not only provides long-term storage of data but also improves advanced analytical capabilities, such as forecasting and predictive analysis through the utilization of Machine Learning (ML) algorithms. The system's low power consumption and modularity make it especially well-adapted to deployment in both industrial remote and urban environments, providing a scalable and economically efficient solution to the smart manufacturing demands typical of the Industry 4.0 era.**

***Index Terms*—Industry 4.0, Smart Data Logger, Modbus RS485, MQTT, ThingsBoard, Predictive Maintenance, Remote Monitoring, Low Power Consumption.**

I. INTRODUCTION

Industrial revolutions made a great significance role and impact on the world, where they helped the countries to grow in terms of economy and helped them for new innovations and technologies.In these periods industries were also benefited by the integration the technologies and new methodologies at every instance. It was changed time to time where we can see the difference in raw materials used, increase in productivity, use of manpower, efficiency and it changed along with the public need and requirement. Industry 4.0 enforce the industries to become smarter as well as efficient. In today's world Industry 4.0 is making drastic changes in human life by bringing automation and data driven work with minimal human intervention. This is achieved through interconnecting the systems and machines by means of internet where the data is shared between them and based on human needs the automation will work [1-2].

The pillars of industry 4.0, which is integrated with the new-end technologies like Internet of Things (IoT), Cyber Physical systems (CPS's), Artificial Intelligence (AI), Machine learning (ML), Cloud Computing, Autonomous systems, etc. Where most of the factories, industrial with systems and machines were made to work smarter and more connected, there is necessity to collect and store information automatically. The device that is connected should record real-time data of all the connected devices which may be like energy usage, working hours, physical parameters inside the industry etc. All the things can be achieved by using a data logger. Here it must store the continuous flow of data from the machines and systems that are connected is very crucial for monitoring their operations, improving efficiency and increasing life of the machines by ensuring their safety. For predictive analysis and automation, the industries need accurate data along with time stamping to analyses system performance and to predict failures to make smart decisions. Moreover, all these things are things are done by human intervention previously but now a days a good data logger can do all these things at a

time. It reduces manual work, provides consistent and accurate measurements, supports local data storage, improves decision making, helps in predictive maintenance, enable real-time monitoring and through AI and ML trends can be analyzed [3-4]. Data logger is an electronic device which is mainly used to record and store data of the connected devices in real time automatically. It can collect data from different sensors, PLC's, Machines etc. The components used to make data logger makes it to stand alone with the others by means of features like offline data storage of data through SD card and time stamping with RTC and data visualization and retrieval through computer or cloud platform. This paper is organized as follows: section II describes related work, section III focuses on the proposed methodology adopted in this work, section IV demonstrates the results achieved and finally the conclusion and future work provided in section V.

II. RELATED WORK

In the new Industrial revolution, where real-time data monitoring and intelligent decision making has become basic needs of modern industrial systems, the evolution of data loggers from simple devices storing data locally to advanced systems interfacing with cloud platforms, supporting automation, and predictive analytics makes it a very versatile device in industrial area. This literature survey aims to explore existing research and technological advancements that came related to data logging systems, particularly based on microcontrollers, Modbus RS485 communication, SD card storage, and cloud-based platforms like ThingsBoard [5-9].

Several researchers have previously evaluated microcontrollers like Arduino, Raspberry Pi, and ESP32 in real-time data acquisition. Among them, the ESP32 has gained the most consideration for making it an economical option for IoT-centric logging applications. A low-cost environmental monitoring system with the ESP32 and DHT22 sensors storing the data in SD cards and periodically uploading it to Thing Speak [10-12].

Communication between sensors and controllers is an important aspect of industrial data logging. Among the many protocols used, Modbus RTU over RS485 is more efficient because of its long-distance connectivity with reliability and with less noise. The integration of Modbus-compatible gas and temperature sensors over RS485 in a manufacturing environment to prove its robustness under noisy industrial conditions. A small drawback is that such existing systems either focus on one sensor or use proprietary software; thus, the flexibility and scale of integration get reduced [13-15].

Another major parameter for successful data logging is sensor integration. In the study of [16-18] a multi-sensor interface was implemented using RF-based Arduino for soil monitoring. But data synchronization and timestamp accuracy became a challenge. To overcome this, they made subsequent studies using Real-Time Clock (RTC) modules, such as the DS3231, for accurate time-stamping and synchronization of logged data.

For storage and transmission, systems are mainly categorized into local storage on SD cards and real-time cloud communication over MQTT/HTTP. Local storage guarantees the safety of data in the event of disconnection, whereas the cloud is used for advanced visualization and control. The recent trend shows a hybrid system, as described by [19-20] with MQTT protocols aligned with Things Board and Blynk for visualization and remote monitoring. The comparison of various data logging methodologies based on physical and operational parameters are given in table I.

TABLE I
COMPARISON OF VARIOUS DATA LOGGING METHODOLOGIES

Methodology	Cloud	Storage	RTC	RS485	Power	Cost	Scalability
Cloud-Only System	Yes	No	No	No	Medium	Low	Low
SD Card Logger	No	Yes	Yes	No	Low	Medium	Medium
SCADA-Based System	Yes	Yes	Yes	Yes	High	High	High
GSM-Based System	Yes	Yes	Yes	No	Medium	Medium	Low
PLC Based System	No	Yes	Yes	Yes	High	Very High	Medium
Arduino Ethernet System	No	Yes	Yes	Partial	Low	Low	Low
Raspberry Pi Logger	Yes	Yes	Yes	Partial	Medium	Medium	Medium

After the literature survey and gone through different existing methodologies which are available in the market, a data logger system which can collect, store and visualize sensor data in real time and works spontaneously need to be developed. In the proposed model the existing limitations were solved by adding time stamping to the collected data, and offline data storage along with online data accessing through IoT cloud platforms and Modbus for industrial compatibility made it into a powerful solution.

III. PROPOSED METHODOLOGY

The proposed device aims to address all these limitations by developing a smart industrial 4.0 based data logger of low-cost, IoT enabled with ESP32 microcontroller designed for real-time data acquisition from multiple Modbus RS485 sensors. It integrates a RTC for times tamping and a micro-SD card module for offline data storage. The ESP32 also Wi-Fi connects to the cloud IoT platform (ThingsBoard) via the MQTT protocol for real-time data visualization. The hybrid approach guarantees that during network outages data logging would be done locally, with automatic uploading as the internet connectivity resumes. The system is very power efficient and can therefore it fit well in remote applications and industrial environments. It helps in reaching our goal of the proposed methodology of designing and developing a complete industrial data logger system for collecting, storing and enables viewing sensor data instantaneously. The figure 1 shows the block diagram of the proposed model.

The ESP32 microcontroller manages data collection and communication in the system. ESP32 microcontroller is selected due to its dual-core processing, integrated Wi-Fi and Bluetooth, low power usage, and serial interface support. It can perform various things at the same time, such as sensor reading, local storage, and real-time cloud communication, and thus it is ideally suited for IoT-based data logging applications. Industrial sensors like the XY-MOD02 (for temperature and humidity) connect via RS485 Modbus RTU using an RS485

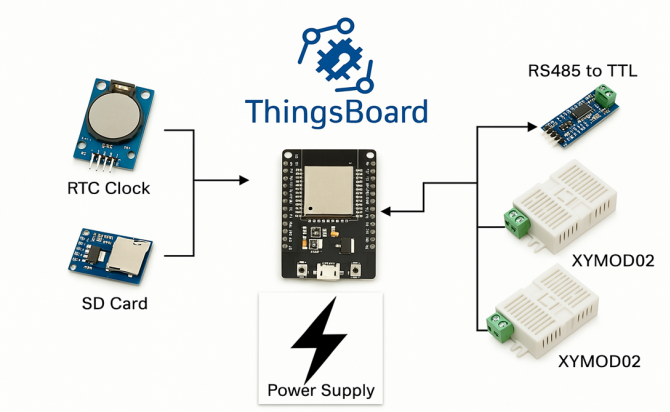

Fig. 1. Block Diagram of Proposed Model

to TTL converter. One of the most critical decisions of this project is the selection of the communication protocol. Because the data logger will be communicating with a variety of industrial sensors, Modbus RTU over RS485 is chosen due to its common use in industry, use over long distances, immunity to noise, and support for multiple devices to communicate. For communicating Modbus-compatible sensors to the ESP32, an RS485 to TTL converter is utilized. The ESP32 sends queries to these sensors, receives data, and timestamps it with a DS3231 RTC module. This data is stored on an SD card for offline access. For local data storage, there is a micro-SD card module. This will ensure that all the data is safely stored even if there is internet or power loss. The Data is stored in .txt format on a micro-SD card, supporting up to 32GB, to ensure minimal memory usage. This data is logged by the system in a formatted file format, and each sensor reading is time-stamped. There is a DS3231 Real-Time Clock (RTC) module for accurate keeping of time, hence ensuring proper data logging and traceability. It is also sent over Wi-Fi to the ThingsBoard cloud platform using the MQTT protocol for remote monitoring. An uninterruptible power supply ensures the system runs without interruption. Real-time monitoring enables early detection of issues and supports predictive maintenance. The system is compact, energy-efficient, and suitable for both urban and remote industrial setups. In order to further facilitate the use of the device in urban and rural settings, a deep emphasis on low-power operation is noted. Future iterations will have power-saving coding practices and hardware sleep modes to maximize battery life in off-grid applications. In this work, a custom DC-DC buck converter was developed to step down the input voltage, typically 24V from PLC to stable 5V to power the SD card module, RS485 transceiver, and for the sensors and 3.3V levels required by ESP32, Real Time Clock in the system. A buck converter was chosen instead of a linear regulator because of its higher efficiency and low heat generation, especially when running for long periods with multiple sensors and peripherals. Using this buck converter, it was ensured that the system remained stable during long data logging sessions, even when all sensors were active and writing data simultaneously to the SD card while uploading to the cloud. For remote monitoring and visualization of data, the ESP32 communicates with the ThingsBoard cloud platform using MQTT protocol, which provides support for real-time dashboards, alerting, and historical data visualization. This cloud integration provides transparency of operations and improves decision-making through remote access.

The design of an industrial data logging system with high-end features requires careful attention to the hardware and software components to provide reliability, expandability, and interoperability in real industrial environments. The goal of this system is to design a compact, economic, and rugged data logging system that can efficiently acquire, store, and forward industrial sensor data in real time. The system is also modular and scalable in design—more sensors can be added by reprogramming the Modbus registers and recompiling the firmware. This renders the data logger flexible to handle different industrial applications and future additions.

The figure 2 shows the breadboard-level setup of a data logging system using an ESP32 microcontroller. It includes two XYMOD02 sensors connected through an RS485 to TTL converter for reliable communication. The XY-MOD02 is a compact, reliable sensor that uses a Modbus RTU protocol and measures temperature and humidity accurately. This sensor was chosen based on RS485 compatibility, such that it could easily interface with the multi-sensor RS485 bus already in place without the need for additional analog or I2C lines. In this testing environment, multiple XY-MOD02 sensors were deployed at different positions for the purposes of monitoring ambient conditions while logging data. The sensor reliably responded to Modbus requests from the ESP32 and could retrieve both temperature and humidity by polling the holding registers (typically 0x0001 and 0x0002 for T and RH). The readings remained stable during continuous operation, and no significant drift or noise was observed, even after 48 plus hours of runtime.

The components are connected on a breadboard for testing and development. This setup helps in collecting and saving environmental data in real-time. To make the developed system more compact, durable and ease of deployment in industrial and harsh environments, a custom PCB was designed. The schematic includes core components like ESP32, RS485 module, RTC(DS3231), SD card module, power management circuit, LED indicators. The main objective of this design is to eliminate breadboard-based interconnections, thereby system reliability and signal routing for RS485 communication is enhanced. All sensor connections and interfaces were mapped to dedicated screw terminals to support industrial-grade wiring. For the Modbus communication in the data logger, no external RS485 breakout modules like the typical MAX485 boards were used. Instead, a customized RS485 transceiver circuit design, directly placed into the main PCB. It offers better electrical stability, less clutter, smaller and more professional design. This PCB was designed using Ki Cad tool and fabricated using standard FR4 material as shown in figure 3. This view provides a clear representation of how the

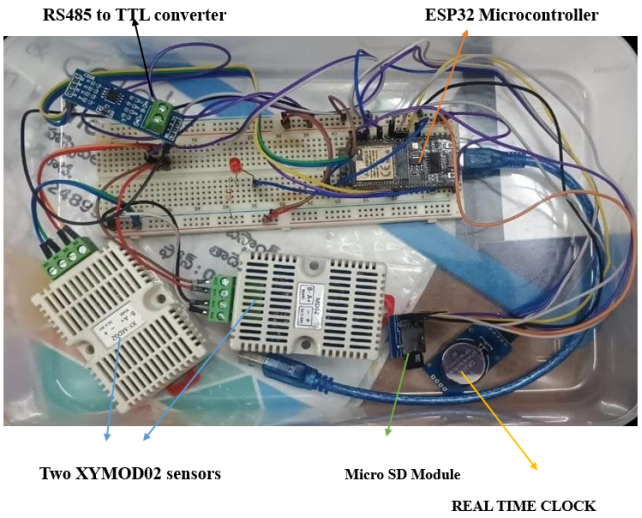

Fig. 2. Bread Board Level Implementation

components are arranged on the board helping with a better understanding of the hardware layout and assembly.

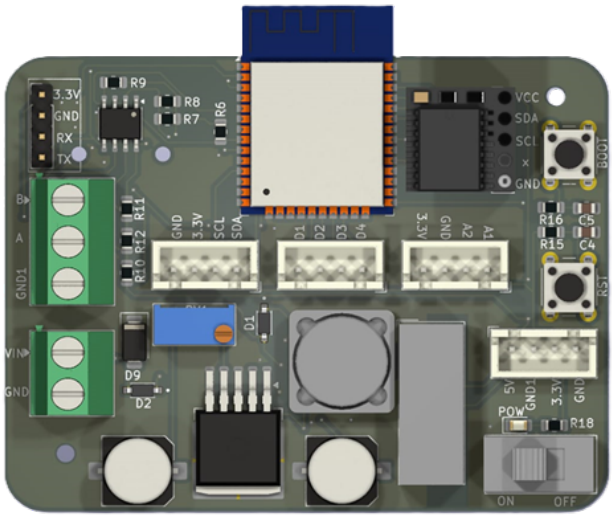

Fig. 3. 3D View of PCB

IV. Results and Discussion

The key focus areas during testing included communication stability with Modbus RS485 sensors, real-time data logging to an SD card, time synchronization using RTC, and cloud integration via MQTT protocol with ThingsBoard. All components were enclosed in an IP65-rated casing and powered via a DC-DC buck converter supplying 5V and 3.3V. The system was tested with several Modbus RS485-compatible sensors, including the XY-MOD02, NPK sensor and Pressure sensors. The ESP32 successfully polled each sensor using its unique slave ID and register addresses. During the tests, the communication speed was set to 9600 bps, and the sensors responded within 150–250 ms consistently. Over a 48-hour period, no communication failures, CRC errors, or response delays were observed, confirming stable and reliable Modbus RS485 data exchange. For Slave ID configuration, QModMaster was used. In order to make sure the ESP32 and Modbus components would communicate over Modbus RTU, it utilizes QModMaster, a free and open-source Windows application for testing Modbus RTU and TCP devices. It was an important part of verifying the code development in the early development and debugging phases. Prior to integrating the Modbus sensors into the ESP32 code, it was manually connected the Modbus sensors to a USB to RS485 converter, and executed Modbus commands with QModMaster. Using QModMaster, it successfully changed the Modbus slave IDs of multiple sensors to avoid address conflicts when multiple sensors were connected to the same RS485 bus. The entire setup including designed PCB is shown in figure 4.

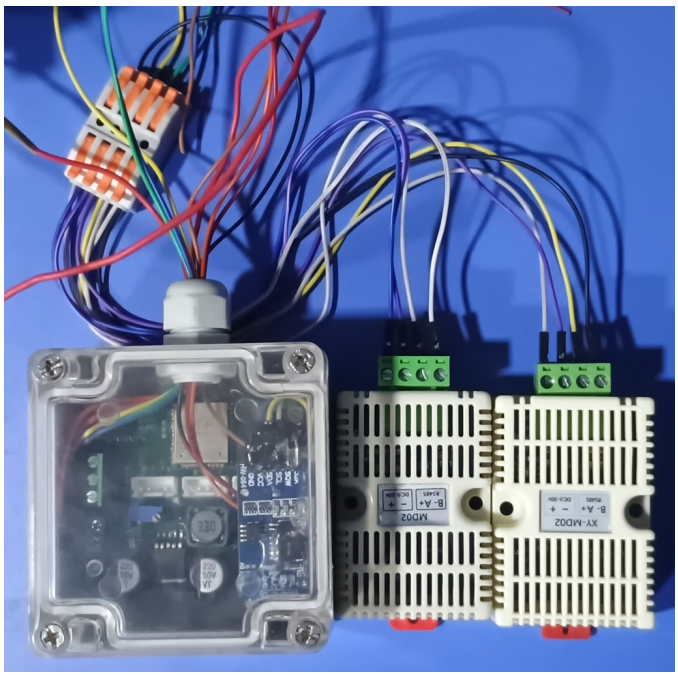

Fig. 4. Experimental Setup

Every 4 seconds, sensor data was logged onto an SD card along with DS3231 RTC time stamp. The RTC module kept a steady time, without any recognizable drifting, during the 48-hour test period. The ESP32 would flush the buffer of the SD card for every write operation, thus preserving the data in case of power outage. The SD card is the main backup method for offline data. The data is logging continuously in a state where all RS485 sensors (temperature, humidity, gas, NPK, and ultrasonic) wrote every few seconds with real time timestamps from the RTC. The file handling remained consistent even after long sessions (tested over 12 hours). Each sensor had its own log file, making it easier to identify and access data. The CSV formatting was used, making it easy to open and review in spreadsheet software. The data entries

were placed in the correct order, with no sign of corruption or duplication as shown in figure 5.

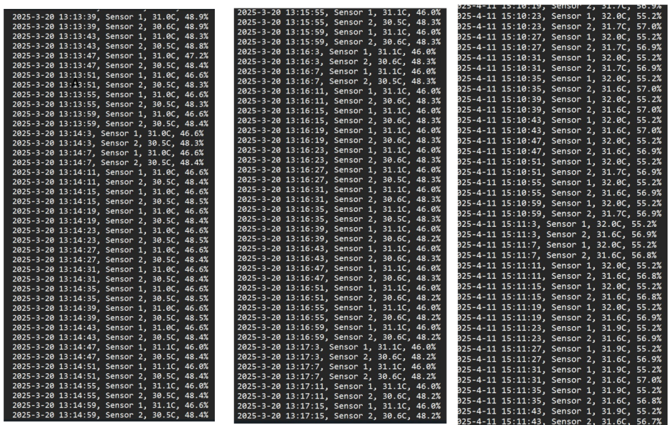

Fig. 5. Data Formatted to SD card with Timestamp

Along with logging onto the SD card, the system published sensor readings to the ThingsBoard server over Wi-Fi using MQTT. The ESP32 connected to a local Wi-Fi router and sent data every 10 seconds. There was an average delay of 200 to 300 milliseconds for MQTT messages to be received by ThingsBoard, and the connection remained stable for the whole test period. All data points were correctly read onto the dashboard with no points missing. This enabled real-time visualization of parameters including temperature, gas concentrations, and water levels. The dashboard widgets made it easy to monitor everything live and have alerts based on thresholds as show in figure 6. The SD card saved local data, as well, so data integrity was maintained during internet outages.

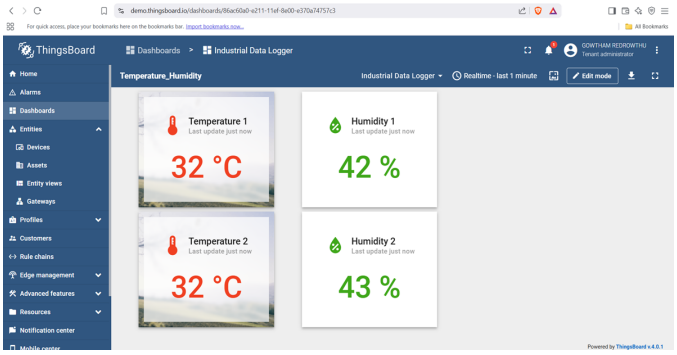

Fig. 6. Data Visualization on Thingsboard

After the logging session, the SD card was taken out and accessed on a PC to ensure data integrity and formatting. The CSV files opened well in Microsoft Excel, with columns lining up rightly so that one column held timestamps while another held sensor values. Data originating from one sensor could be viewed alongside data from another, and Excel tools were utilized for line graphs and trend curves, as well as distributions and statistical summaries. This confirmed that CSV output from the data logger was both reliable and immediately usable for industrial data analysis workflows in Excel shown in figure 7.

	Column1	Column2	Column3	Column4
2	20-03-2025 14:44	Sensor 1	31.3C	0.437
3	20-03-2025 14:44	Sensor 2	30.5C	0.463
4	20-03-2025 14:44	Sensor 1	31.3C	0.437
5	20-03-2025 14:44	Sensor 2	30.5C	0.463
6	20-03-2025 14:44	Sensor 1	31.3C	0.437
7	20-03-2025 14:44	Sensor 2	30.5C	0.463
8	20-03-2025 14:44	Sensor 1	31.3C	0.437
9	20-03-2025 14:44	Sensor 2	30.5C	0.463
10	20-03-2025 14:44	Sensor 1	31.3C	0.437
11	20-03-2025 14:44	Sensor 2	30.5C	0.464
12	20-03-2025 14:45	Sensor 1	31.3C	0.437
13	20-03-2025 14:45	Sensor 2	30.5C	0.464
14	20-03-2025 14:45	Sensor 1	31.3C	0.437
15	20-03-2025 14:45	Sensor 2	30.5C	0.464
16	20-03-2025 14:45	Sensor 1	31.3C	0.437
17	20-03-2025 14:45	Sensor 2	30.5C	0.463
18	20-03-2025 14:45	Sensor 1	31.3C	0.437
19	20-03-2025 14:45	Sensor 2	30.5C	0.463
20	20-03-2025 14:45	Sensor 1	31.3C	0.437
21	20-03-2025 14:45	Sensor 2	30.5C	0.463
22	20-03-2025 14:45	Sensor 1	31.3C	0.437
23	20-03-2025 14:45	Sensor 2	30.5C	0.463
24	20-03-2025 14:45	Sensor 1	31.3C	0.437
25	20-03-2025 14:45	Sensor 2	30.5C	0.463

Fig. 7. Data on Excel Sheet

The data from handheld measuring instruments confirmed the results of cross-validation, with a deviation of ±2%, thus ensuring a high precision standard that meets industrial standards. The system exhibits low latency in communication, with an average of 150-300 ms data acquisition to upload to the cloud using MQTT. Local data logging on an SD card presents very low delays ($<$ 100 ms) so that real-time capture is ensured even during cloud storage outages. For long-term tests lasting 24-48 hours, the SD card module demonstrated steady performance. It was able to store the data safely in a well-structured folder of.txt files with 100% data retention, even after power cycles. The DS3231 RTC timestamps remained stable and consistent, allowing chronological logging with no observed drift. While it was drawing about 150-240 mA in the periods when the active sensors collected the data and when the communication is also active with Wi-Fi. If it is configured in deep sleep or just idle, it will only be around 10-20 mA so this is truly appropriate for low-power industrial as well as remote applications. The entire system consumption under 5V, including sensors and SD card, stayed below 300-350 mA. The overall comparison of the proposed system with existing system along all parameters is shown in table II.

TABLE II
COMPARISON WITH EXISTING SYSTEMS

Parameter	Our Proposed System	Conventional Commercial Data Logger
Hardware Cost	Very Low	High
Sensor Interface	RS485 (up to 22 sensors)	Limited or fixed number of inputs
Data Storage	SD card with timestamped CSV files	Internal memory or external cloud only options
Time Synchronization	DS3231 RTC with NTP connectivity with accuracy	Usually, RTC integrated
Cloud Integration	MQTT to ThingsBoard	Vendor-specific cloud platforms
Expandability	Highly scalable	Limited expandability; often requires hardware upgrade
Power Consumption	Low power consumption	Moderate to high
Ease of Maintenance	Easy	Requires trained personnel
Suitability for Industry 4.0	High (open protocol, real-time monitoring, remote access)	Moderate, unless upgraded with IoT extensions

V. CONCLUSION AND FUTURE ENHANCEMENT

IoT-based data logging system that incorporates various aspects of Industry 4.0, and, as such, stands to significantly improve all industrial monitoring and control processes. At the center of this system is the ESP32 microcontroller, RS485 Modbus RTU communication standard, DS3231 Real-Time Clock (RTC) module is integrated into the system to ensure data accuracy and storage as relevant to time, allowing precise time stamping of sensor readings. With another strength, the system allows dual data storage, meaning data can be stored offline on an SD card and online on the cloud via ThingsBoard using the MQTT protocol. Thus, data are available when there are network outage circumstances, allowing historical analysis and remote monitoring through an intuitive IoT dashboard. Low-power design was emphasized in the design. The system works on 5V with average currents below 350 mA, thus finding use for day-and-night industrial sites. Further, the modularity of the system, combined with RS485 communication, makes it possible to connect more than 22 sensors over long distances with great reliability-a feature set required for Industry 4.0 real-time monitoring, inter-device communication, and cloud-based analytics, thus pre-empting the huge level of automation. The addition of a relay interface allows the system to be further extended for automatic actuation depending on sensor thresholds, facilitating closed-loop control. AI and ML algorithm additions will be next for predictive maintenance, anomaly detection, and intelligent decision-making. The system can be modified for uses in aquaculture, agriculture, water treatment, and industrial manufacturing, among others. Essentially, this work presented a highly reliable, scalable, and efficient IoT data logger system, which is based on Industry 4.0 core principles and can serve as a strong building block for future smart industrial systems.

REFERENCES

[1] Ghobakhloo, Morteza. "Industry 4.0, digitization, and opportunities for sustainability." Journal of cleaner production 252 (2020): 119869.

[2] Moraes, E.B., Kipper, L.M., Hackenhaar Kellermann, A.C., Austria, L., Leivas, P., Moraes, J.A.R. and Witczak, M. (2023), "Integration of Industry 4.0 technologies with Education 4.0: advantages for improvements in learning", Interactive Technology and Smart Education, Vol. 20 No. 2, pp. 271-287. https://doi.org/10.1108/ITSE-11-2021-0201.

[3] Soori, Mohsen, Behrooz Arezoo, and Roza Dastres. "Internet of things for smart factories in industry 4.0, a review." Internet of Things and Cyber-Physical Systems 3 (2023): 192-204.

[4] Mastos, Theofilos D., et al. "Industry 4.0 sustainable supply chains: An application of an IoT enabled scrap metal management solution." Journal of cleaner production 269 (2020): 122377.

[5] N. Choudhary and S. Pal, "IoT-based predictive maintenance with ESP32 and ThingsBoard integration," *J. Ind. Autom. AI*, vol. 2, no. 1, pp. 25–31, 2023.

[6] P. Thakur and J. Joseph, "Industrial data acquisition and control using Modbus protocol on ESP32 platform," *Int. J. Eng. Trends Technol.*, vol. 71, no. 9, pp. 48–54, 2023.

[7] H. Joshi and R. Mehta, "Smart agriculture using RS485-enabled soil NPK sensors and ESP32," *Asian J. Electron. Commun.*, vol. 16, no. 3, pp. 112–118, 2022.

[8] S. Raj and N. Kumar, "Cloud-integrated data acquisition system using MQTT protocol and ThingsBoard platform," *Int. J. Adv. Res. Comput. Sci.*, vol. 13, no. 4, pp. 123–130, 2022.

[9] L. Zhou, X. Zhang, and H. Chen, "Design of Soil NPK Monitoring System Based on ESP32 and RS485 Sensors," *J. Phys.: Conf. Ser.*, vol. 2203, p. 012045, Mar. 2022.

[10] R. Sharma and P. Roy, "Design of a reliable IoT-based temperature and humidity monitoring system using ESP32," *Int. J. Sci. Eng. Res.*, vol. 13, no. 2, pp. 1020–1025, 2022.

[11] A. Kumar and P. Sinha, "Modbus RTU protocol implementation for industrial sensor networks," *J. Instrum. Technol.*, vol. 38, no. 3, pp. 145–152, 2021.

[12] A. Singh and A. Verma, "Real-time data logging using SD card and ESP32 for remote monitoring," *J. Embedded Syst.*, vol. 14, no. 1, pp. 33–40, 2021.

[13] A. Khan and F. Iqbal, "Secure IoT data transmission using MQTT protocol on ESP32 platform," *J. Secure Syst. Eng.*, vol. 9, no. 2, pp. 73–80, 2021.

[14] T. Chanwimalueang, L. Aufiero, and D. T. H. Le, "An ESP32-Based Open-Source Module for Wearable Physiological Signal Monitoring," *Sensors*, vol. 21, no. 19, p. 6549, Sep. 2021.

[15] R. Banerjee and Z. Ahmed, "Low power IoT systems using ESP32 for real-time environmental monitoring," *J. Internet Things Cloud Comput.*, vol. 8, no. 1, pp. 45–52, 2020.

[16] F. Li and J. Wang, "Development of a cloud-based data logging system for environmental sensors," *Sensors Appl. J.*, vol. 29, no. 4, pp. 310–316, 2020.

[17] D. Patel and K. Sharma, "IoT based industrial automation using ESP32 and cloud integration," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 9, no. 6, pp. 785–789, 2020.

[18] D. Patel and A. Verma, "Autonomous temperature data logger using ESP32 with RTC and SD Card," *J. Instrum. Sci.*, vol. 11, no. 3, pp. 101–108, 2020.

[19] ThingsBoard Authors, "Modbus Integration with ThingsBoard," ThingsBoard Documentation. [Online]. Available: https://thingsboard.io/docs/iot-gateway/modbus/ (Accessed: May 2024).

[20] B. Silva and L. Silva, "A Modbus RTU to MQTT IoT Gateway using ESP32," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Las Vegas, NV, USA, 2021, pp. 1–2.