MANIPAL
ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

MANIPAL SCHOOL OF INFORMATION SCIENCES
(A Constituent unit of MAHE, Manipal)

BACnet to Cloud IoT Gateway

Reg. Number	Name	Branch
251100620034	Dhanvi Vijaykumar	ES
251100620030	Abhaya Y	ES
251100620014	Abhishek G M	ES

Under the guidance of

Dr B Dinesh Rao
Professor,
Manipal School of Information Sciences,
MAHE, MANIPAL

14/11/2025

MANIPAL SCHOOL OF INFORMATION SCIENCES
MANIPAL
(A constituent unit of MAHE, Manipal)

ABSTRACT

This project presents the design and implementation of a BACnet-to-Cloud IoT Gateway that enables seamless integration of Building Automation and Control Network (BACnet) devices with modern cloud platforms. Using an open-source BACnet-stack written in C, the system was deployed on a Raspberry Pi acting as the gateway, communicating with BACnet devices over RS485 (MS/TP). For testing and development, BACnet devices were simulated using a C-based BACnet server running on a laptop. A custom Linux bash script was developed to extract BACnet object values such as Analog Inputs, Binary Inputs, Multi-State Values, and Accumulators and upload them to the Firebase Realtime Database using its REST API.

The gateway collects live BACnet data, converts it into JSON format, and timestamps each upload before sending it to Firebase. The cloud database allows remote monitoring of building automation parameters from any internet-connected client. Extensive testing verified reliable communication between the BACnet simulator and the Raspberry Pi gateway, as well as successful real-time data updates in Firebase. This project demonstrates a practical approach to bridging legacy building automation protocols with modern IoT cloud platforms, enabling enhanced monitoring, scalability, and future integration with analytics or dashboard applications.

Table of Contents

List of Figures

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Building Automation and Control systems rely heavily on standardized communication protocols to ensure interoperability among devices from different vendors. One of the most widely adopted standards in this domain is BACnet (Building Automation and Control Network), a protocol designed for HVAC, lighting, access control, and energy management systems. BACnet devices typically operate in closed or local networks, limiting their ability to interact with modern cloud-based platforms that offer real-time monitoring, data analytics, and remote control.

With the rapid growth of the Internet of Things (IoT), there is a strong need to bridge legacy industrial protocols like BACnet with scalable cloud ecosystems. Cloud integration enables advanced features such as centralized monitoring, remote diagnostics, predictive maintenance, and data-driven optimization. However, connecting BACnet devices to the cloud requires a gateway capable of understanding BACnet object structures, extracting real-time data, and communicating with cloud APIs securely and efficiently.

This project focuses on developing a BACnet-to-Cloud IoT Gateway using a Raspberry Pi as the core processing unit. The Raspberry Pi runs an open-source BACnet-stack implemented in C, which allows it to communicate with BACnet devices over RS485 (MS/TP). For development and testing purposes, a BACnet simulation server running on a laptop was used to emulate various BACnet objects such as Analog Inputs, Binary Inputs, Multi-State Inputs, and Accumulators. The gateway collects values from these objects and uploads them to the Firebase Realtime Database, a cloud-based NoSQL database that stores and syncs JSON data in real time.

The integration between BACnet and Firebase is achieved through a custom Linux bash script, which extracts object-value pairs from the BACnet client, formats them into JSON, timestamps each entry, and sends it to Firebase using RESTful API

calls. This architecture demonstrates how traditional building automation systems can be enhanced with modern cloud capabilities, enabling remote access, visualization, and further IoT-based developments.

1.2 OBJECTIVES

- Understand BACnet protocol and its device communication.
- Learn RS485 communication for BACnet MSTP networks.
- To implement BACnet communication using Raspberry Pi and BACnet-stack written in C.
- Simulate BACnet devices and generate sensor data.
- To upload real-time BACnet data to Firebase Realtime Database.
- To enable remote monitoring of BACnet parameters via Firebase.

CHAPTER 2
LITERATURE SURVEY

- Building automation and smart homes rely on standardized communication protocols. [1] Altmann et al. (2015) proposed a BACnet gateway integrating BACnet with embedded web services using a lightweight BACnet/WS and DPWS adaptation, enabling device discovery and data access on constrained devices via a modular OSGi framework.
- [2] Sankar et al. (2023) developed an energy-efficient LED lighting control system based on BACnet, highlighting LED advantages and BACnet's role in enabling interoperable, adaptive lighting control and remote monitoring in smart buildings.
- [3] Park et al. (2010) introduced a BACnet-ZigBee gateway translating BACnet MSTP to ZigBee messages, overcoming BACnet's wireless limitations and enhancing flexibility in smart building environments.
- [4] Yimer et al. (2025) addressed cybersecurity challenges in BACnet MSTP cloud integration, showing that edge-cloud architectures improve anomaly detection but increase attack risks, emphasizing the need for strong forensic and disaster recovery strategies.
- [5] Nast et al. (2019) evaluated BACnet/IP security impacts, finding that mechanisms like HMAC and AES reduce throughput on constrained devices, highlighting the trade-offs between security and performance in BACnet deployment.
- [6] Jian-cang Huang (2018) implemented a BACnet/IP controller on ARM9 Linux hardware, demonstrating reliable real-time building automation through hardware-software co-design.
- [7] Christian Bock et al. (2019) developed an embedded BACnet stack for ARM Cortex-M using Mbed OS, enabling standardized communication for smart sensors and actors with real-time control on Ethernet networks.

CHAPTER 3
SPECIFICATIONS

The project implements the BACnet protocol using the MS/TP (Master-Slave/Token-Passing) communication method, operating over the RS485 physical layer to support stable and long-distance communication suitable for building automation environments. The system is configured to communicate at a baud rate of 38,400 bps, providing an optimal balance between performance and data integrity.

A virtual BACnet device is executed on a Windows laptop using a C-based BACnet simulation program, which emulates standard BACnet objects such as Analog Inputs (AI), Binary Inputs (BI), Multi-State Inputs (MSI), and Accumulators. The Raspberry Pi 3 Model B functions as the BACnet gateway, running the BACnet-stack written in C, responsible for querying device objects and forwarding the data to the cloud.

Each device on the BACnet MS/TP network is assigned a unique device instance number to avoid conflicts. In this setup, the simulated BACnet server uses Device ID 1001, while the Raspberry Pi gateway operates as Device ID 2001. Communication occurs through an RS485 twisted-pair line connected via a USB-to-RS485 converter on the Raspberry Pi (/dev/ttyACM0).

The gateway software stack consists of the BACnet-stack C library for protocol handling and Linux bash scripts for packaging and transmitting JSON data to Firebase Realtime Database via REST API calls. The system collects and uploads BACnet object values periodically, enabling timestamped data logging and near real-time monitoring of building automation parameters.

The Firebase cloud backend is structured as a NoSQL JSON tree, where each data upload is stored under a unique timestamp, ensuring organized and scalable storage for future analytics. This architecture supports future enhancements such as remote dashboards, mobile applications, and integration with machine learning-based prediction engines.

CHAPTER 4

METHODOLOGY

This section outlines the systematic approach followed for designing, implementing, and validating the BACnet-to-Firebase IoT Gateway using the BACnet-stack C library, Raspberry Pi, and a simulated BACnet device. The methodology corresponds to the architecture described earlier and details each functional layer of the system.

4.1 OVERVIEW OF SYSTEM ARCHITECTURE

The methodology is based on a layered communication framework that interconnects a virtual BACnet device running on a laptop with a Raspberry Pi gateway over RS485 (MS/TP), which then forwards processed data to the Firebase Realtime Database. The architecture consists of the following core functional modules:

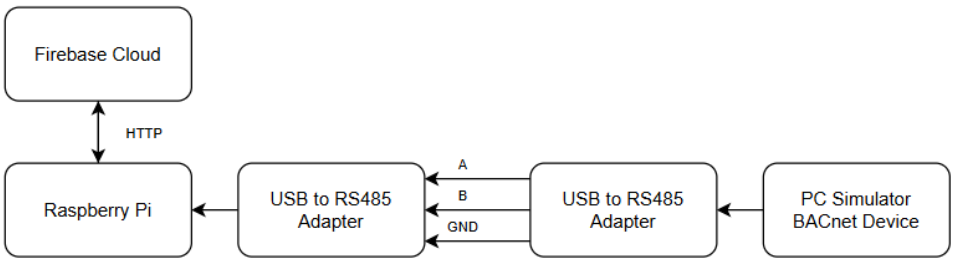

Figure 1: Block Diagram of the system

Simulated BACnet Device (PC-Based):

- A virtual BACnet device is executed on a laptop using a C-based BACnet simulation program.
- It emulates multiple BACnet objects such as Analog Inputs (AI), Binary Inputs (BI), Multi-State Inputs (MSI), and Accumulators.
- The simulator generates varying sensor-like values to replicate real-time building automation parameters.

- This acts as the primary data source for the gateway during testing.

RS485 Communication Layer:

- Data transfer between the PC simulator and Raspberry Pi is established using USB-to-RS485 converters, forming a BACnet MS/TP communication channel.
- RS485 is selected due to its industrial-grade noise immunity and ability to support multi-drop topology over long cable lengths.
- The communication operates at a baud rate of 38,400 bps, ensuring reliable and consistent token-passing between BACnet devices.

Raspberry Pi Gateway:

- The Raspberry Pi runs the BACnet-stack written in C, compiled natively on the Pi.
- It performs MS/TP device discovery, token passing, and Read-Property operations on the simulated BACnet objects.
- Extracted values are processed, formatted, and prepared for cloud transmission.
- The Pi serves as the central decision-making unit, handling data acquisition and protocol translation.

Data Formatting and Bash Automation:

- A custom bash script on the Raspberry Pi automatically captures BACnet object values output by the C program.
- The script converts collected parameters into JSON format, assigns a timestamp, and prepares the structure for cloud uploading.
- Error handling, retries, and continuous monitoring mechanisms are incorporated to maintain upload reliability.

Cloud Integration (Firebase):

- Data is transmitted to the Firebase Realtime Database using HTTPS-based REST API calls with curl.

- Each data point is stored under a unique timestamp, ensuring organized representation and traceable historical logging.
- Firebase provides synchronized data access across clients, enabling real-time monitoring from web or mobile interfaces.
- This module transforms local BACnet automation data into globally accessible IoT information.

4.2 STEP-BY-STEP IMPLEMENTATION

4.2.1 BACnet Device Simulation:

- Installation of the BACnet-stack C-based simulator on the laptop for generating BACnet object data.
- Configuration of virtual BACnet objects including:
  - Analog Inputs (AI) such as temperature or voltage.
  - Binary Inputs (BI) such as ON/OFF states.
  - Multi-State Inputs (MSI) representing enumerated modes.
  - Accumulator objects for counting operations.
- Assignment of a unique BACnet device instance (e.g., Device ID = 1001).
- Execution of the simulator to periodically update object values and respond to Read-Property requests from the Raspberry Pi gateway.

4.2.2 RS485 Communication Setup:

- Physical wiring of RS485 lines using twisted-pair cables for A and B differential signals, along with GND for reference.
- Connection of the PC simulator and Raspberry Pi through USB-to-RS485 adapters.
- Verification of serial detection on the Raspberry Pi (/dev/ttyACM0) and configuration of baud rate to 38400 bps.
- Testing MS/TP communication using BACnet-stack diagnostic tools to ensure reliable token passing.

4.2.3 Raspberry Pi Gateway Operation:

- Compilation and deployment of the BACnet-stack (C library) on the Raspberry Pi to enable BACnet MS/TP communication.
- Initialization of serial communication parameters on the Pi for MS/TP interaction with the simulator.
- Execution of a custom BACnet client program that:
    - Sends Read-Property requests to the simulated BACnet device.
    - Parses incoming frames containing AI, BI, MSI, and Accumulator data.
    - Logs extracted values to the Pi terminal for verification.
- Integration of a bash script to automate:
    - Reading BACnet output.
    - Converting values into JSON format.
    - Generating timestamps for each dataset.
    - Uploading the structured data to Firebase via HTTPS (REST API).
- Implementation of error handling to address:
    - Serial port access issues.
    - Communication interruptions.
    - BACnet device unreachable states.
    - Failed Firebase upload attempts.

4.3 ADVANTAGES OF THIS APPROACH

- **Reliability:** RS485-based BACnet MS/TP communication ensures stable and noise-resistant data transfer between devices.
- **Scalability:** Additional BACnet devices or cloud endpoints can be integrated without modifying the core gateway architecture.
- **Flexibility:** Supports legacy BACnet systems through the BACnet-stack while enabling seamless integration with modern IoT platforms like Firebase.
- **Real-Time Monitoring:** Timestamped uploads to Firebase allow continuous tracking of sensor and actuator data from any location.

- **Cost-Effectiveness:** Utilizes open-source BACnet-stack and low-cost Raspberry Pi hardware, reducing overall deployment expenses.
- **Cloud Accessibility:** Data stored in Firebase can be accessed through web dashboards, mobile apps, or analytics services.

CHAPTER 5
RESULTS AND OBSERVATIONS

The project successfully demonstrated complete end-to-end data flow from BACnet MS/TP device simulation to BACnet polling on the Raspberry Pi gateway, JSON conversion, and final storage in the Firebase Realtime Database. All major components - BACnet-stack, RS485 communication, bash-based data extraction, and Firebase REST API uploads operated reliably across multiple test cycles.

5.1 BACNET MSTP DEVICE AND GATEWAY OPERATION

The BACnet virtual device running on the laptop and the Raspberry Pi gateway were connected using an RS485 USB interface and configured according to BACnet MS/TP requirements. As seen in the attached terminal output screenshots:

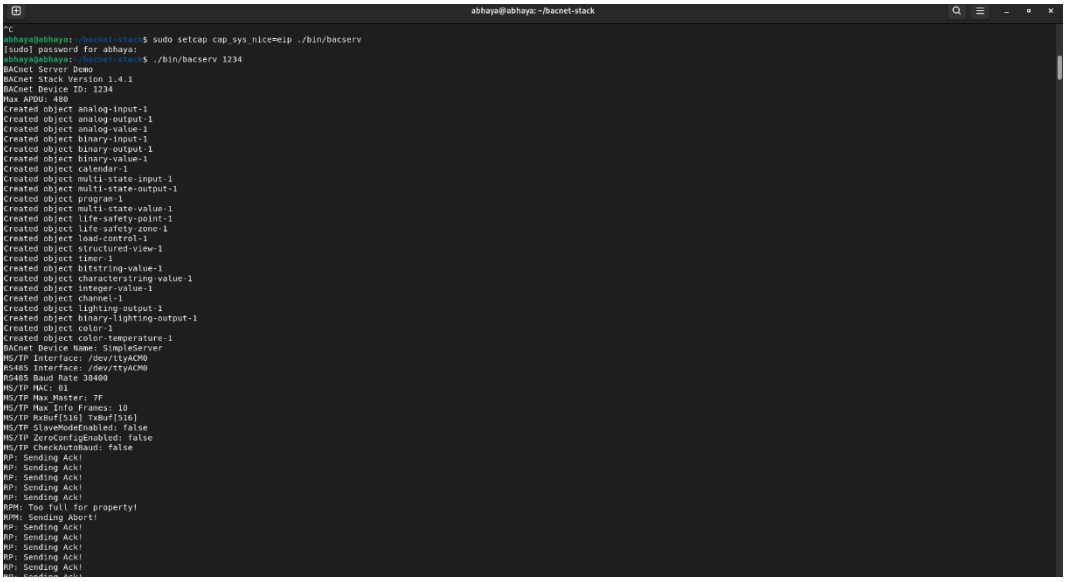

Figure 2: BACnet Server Startup

- The BACnet server initializes correctly with:
    - Device ID: 1234
    - Baud Rate: 38400
    - MS/TP MAC: 01
    - Max Master: 7F

    - Max Info Frames: 10
- The server successfully creates multiple BACnet objects including:
    - Analog Input
    - Analog Value
    - Binary Input
    - Binary Value
    - Multi-State Value
    - Accumulator
    - Timer, Schedule, Channel, Lighting Output, etc.
- Continuous MS/TP frames appear in the log:
    - “RP: Sending Ack!”, “RPM: Sending Abort!”, confirming active BACnet traffic.

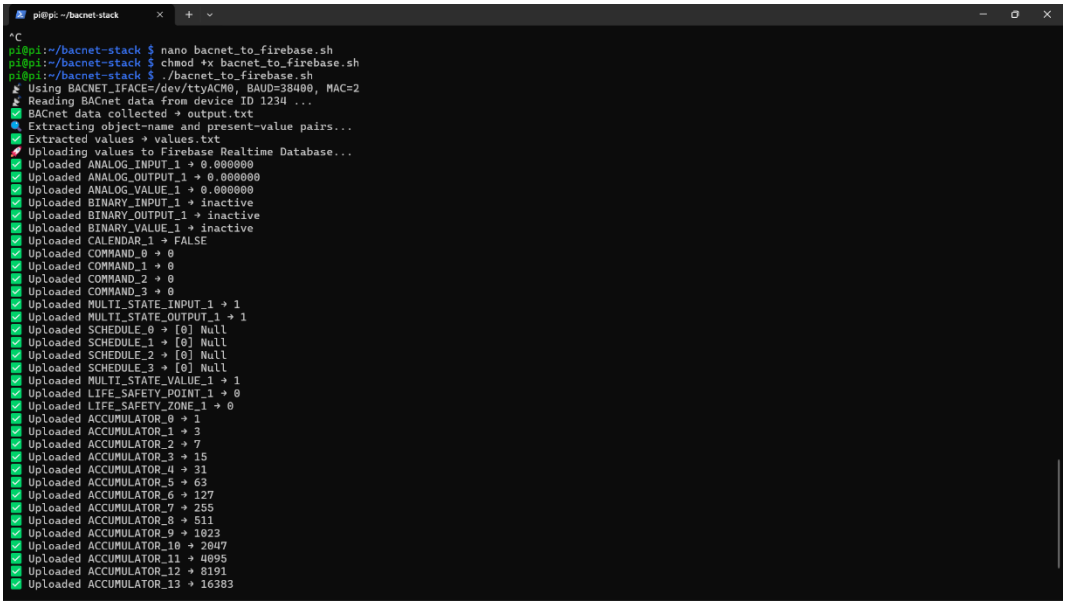

Figure 3: Pi Gateway Reading and Uploading Data

The Raspberry Pi executes the bacnet_to_firebase.sh script which performs the following operations:

- Captures BACnet output into output.txt
- Extracts object/value pairs into values.txt
- Uploads each object to Firebase with timestamps

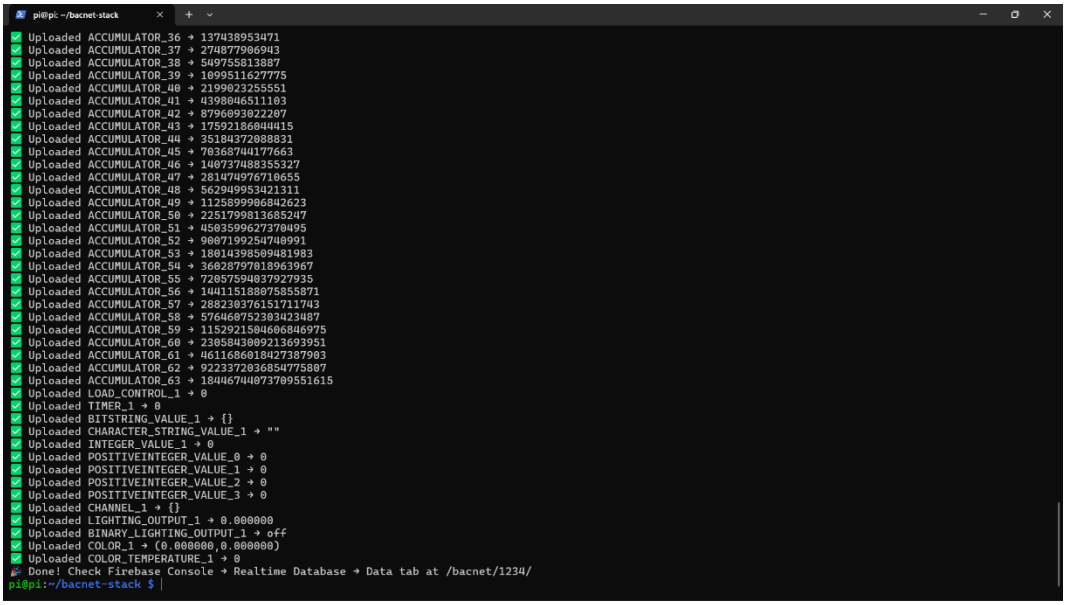

Figure 4: Pi Gateway Reading and Uploading Data

- Each upload is confirmed with lines such as:
  - Uploaded ANALOG_INPUT_1 → 0.000000
  - Uploaded BINARY_INPUT_1 → inactive
  - Uploaded MULTI_STATE_INPUT_1 → 1
  - Uploaded ACCUMULATOR_37 → 274877906977
  - Uploaded COLOR_1 → (0.000000, 0.000000)
- The green success icons in the screenshot confirm that each individual BACnet object was transmitted to Firebase without errors.
- All accumulator objects (ACCUMULATOR_0 to ACCUMULATOR_63) were uploaded, including extremely large values demonstrating the script's capability to process and transfer high-resolution numerical BACnet data.
- No communication interruptions or script failures were observed throughout multiple testing cycles.

5.2 CLOUD DATA STORAGE AND VISUALIZATION IN FIREBASE

To verify cloud integration, the Firebase Realtime Database was inspected using the Firebase Console.

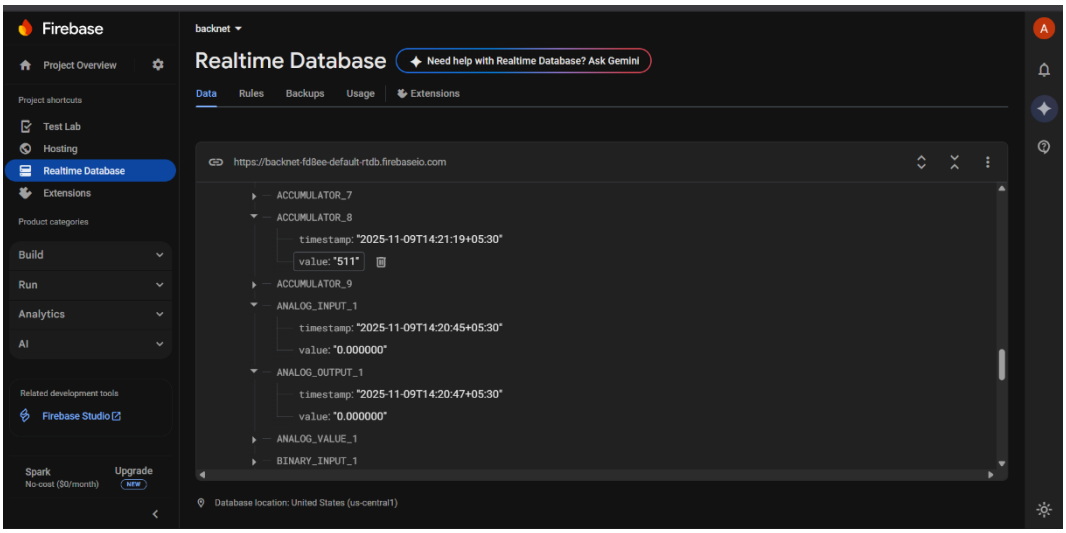

Figure 5: Firebase Database View

- The uploaded data is correctly stored under the database path: /bacnet/1234/
- Each BACnet object is stored as a structured JSON node with fields:
    - timestamp
    - value
- ACCUMULATOR_8
    - timestamp: "2025-11-09T14:21:19+05:30"
    - value: “511”
- ANALOG_INPUT_1
    - timestamp: "2025-11-09T14:20:45+05:30"
    - value: "0.000000"
- ANALOG_OUTPUT_1, BINARY_INPUT_1, and other objects similarly show correct uploads.
- Firebase’s JSON tree view confirms:
    - All object names match the BACnet-stack output.
    - Data is consistently timestamped.

    - Multiple BACnet object types (AI, BI, MSI, AV, BO, ACC) appear correctly in the cloud.
    - No malformed JSON nodes or upload failures were found.

- This verifies the correct functioning of:
    - BACnet polling
    - Data extraction
    - JSON formatting
    - REST API upload
    - Database structuring in the cloud

CHAPTER 6
FUTURE WORK

While this project successfully demonstrated BACnet MS/TP device simulation, Raspberry Pi gateway operation, data extraction, and cloud upload to Firebase, there remain several strong opportunities for enhancement, scalability, and real-world deployment readiness.

6.1 FULL BACNET INTEROPERABILITY AND COMPLIANCE

- Extend the BACnet-stack implementation to support additional BACnet standard objects such as Trend Logs, Calendars, Schedules, Loop objects, and Notification Classes.
- Enable WriteProperty and Commandable properties, allowing remote actuation of BACnet devices through cloud commands.
- Add support for BACnet alarming, including event notifications and fault detection mechanisms.
- Perform interoperability testing with multiple BACnet clients (YABE, BACnet explorer tools, and actual BMS controllers).

6.2 CLOUD AND MOBILE INTEGRATION

- Implement full bi-directional communication between Firebase and the Raspberry Pi gateway for remote control actions.
- Integrate more secure and scalable cloud protocols such as MQTT with TLS, AWS IoT, Azure IoT Hub, or Google IoT Core.
- Develop a complete web dashboard using real-time graphs, historical trend charts, and device status pages.
- Build Android/iOS mobile applications to provide building managers direct access to BACnet device data and notifications.
- Add user authentication, access control, and database read/write security rules for deployment-grade cloud communication.

6.3 MULTI-DEVICE SUPPORT

- Expand support for multiple BACnet MS/TP devices on the same RS485 bus to simulate a real multi-node building automation network.
- Add intelligent polling mechanisms for large BACnet networks to optimize bandwidth and avoid congestion.
- Implement device discovery (Who-Is/I-Am) for dynamically attaching new BACnet devices.
- Introduce support for additional BACnet datalinks such as BACnet/IP, enabling hybrid MS/TP and IP-based building systems.

6.4 ADVANCED DATA ANALYTICS AND AUTOMATION

- Implement machine learning models for anomaly detection, energy prediction, or fault diagnostics using Firebase data.
- Add automated reporting features such as daily/weekly summaries, performance indicators, and consumption insights.
- Integrate real-time alerting via email, SMS, or push notifications for critical system events.

6.5 HARDWARE AND SYSTEM OPTIMIZATION

- Upgrade the gateway to Raspberry Pi 4/5 for improved performance, higher throughput, and advanced security features.
- Add a local storage buffer to store BACnet data during internet outages and sync upon reconnection.
- Improve error recovery mechanisms for serial communication to handle line disturbances and MS/TP collisions more efficiently.

CHAPTER 7
CONCLUSION

This project successfully demonstrated the design and implementation of a complete BACnet-to-Cloud IoT Gateway using the open-source BACnet-stack in C, a Raspberry Pi as the gateway device, and Firebase as the cloud storage platform. By integrating BACnet MS/TP communication over RS485 with cloud-based data logging, the system proved its capability to bridge traditional building automation technologies with modern IoT infrastructures.

The Raspberry Pi gateway reliably collected data from the simulated BACnet device, parsed object values, converted them into JSON format, and uploaded them to the Firebase Realtime Database with accurate timestamps. The test results, supported by terminal logs and Firebase screenshots, confirmed that the data flow from BACnet device creation to cloud visualization was stable, consistent, and error-free.

The project also demonstrated that legacy BACnet systems can be effectively modernized without altering device hardware, simply by introducing a lightweight and scalable gateway. This provides a practical pathway for buildings and industries looking to integrate real-time monitoring, cloud analytics, and remote access into their existing BACnet-controlled environments.

Although the current implementation focuses primarily on read-only data acquisition and cloud uploading, the architecture lays a solid foundation for further advancements such as bi-directional control, multi-device support, stronger security, and advanced data analytics. With these enhancements, the system can evolve into a robust, enterprise-ready platform for smart building automation and IoT-enabled facility management.

REFERENCES

[1] V. Altmann, B. Butzin, R. Balla, F. Golatowski, and D. Timmermann, "A BACnet Gateway for Embedded Web Services," *Proc. IEEE Int. Conf. on Industrial Informatics*, pp. 1–8, 2015.

[2] P. Sankar, R. Vallikannu, S. Karg, and G. Justin, "Ambient Intelligence based LED Lighting Control System Using BACnet Protocol," in *Proc. 2023 Int. Conf. on Artificial Intelligence and Applications (ICAIA)*, pp. 1–5, Chennai, India, 2023.

[3] J. Park, K. Kang, D. Kang, and K. Cho, "The Design and Implementation of BACnet-ZigBee Gateway," in *Proc. 2010 IEEE Int. Symposium on Industrial Electronics*, pp. 2243–2248, 2010.

[4] S. Yimer, A. Zhang, Y. Shi, and K. V. S. R. Anjaneyulu, "Cybersecurity and Forensic Framework for BACnet MS/TP Cloud Integration," in *Proc. IEEE Int. Conf. on Smart IoT*, pp. 1–6, 2025.

[5] H. Nast, A. Wallner, and W. Kastner, "Performance Analysis of the BACnet Secure Connect Protocol on Resource-Constrained Devices," in *Proc. 24th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*, pp. 1477–1480, 2019.

[6] J.-c. Huang, "Design of Building Automation Controller Based on Embedded ARM9 Platform and BACnet/IP Protocol," *International Journal of Smart Home*, vol. 12, no. 2, pp. 31–40, 2018.

[7] C. Bock, S. Kleiminger, and S. Fiedler, "Embedded BACnet Stack for ARM Cortex-M Microcontrollers with Mbed OS," in *Proc. IEEE Int. Conf. on Industrial Informatics*, pp. 755–758, 2019.

[8] "BACnet Home," BACnet International, [Online]. Available: https://www.bacnet.org/.

[9] BACnet Stack – Open Source BACnet Protocol Stack. Available: https://bacnet.sourceforge.net/