Google Firebase based Modern IoT System Architecture

B Padmaja
Computer Science and Engineering,
Institute of Aeronautical Engineering,
Hyderabad, India.

E Krishna Rao Patro
Computer Science and Engineering
Institute of Aeronautical Engineering
Hyderabad, India.

Sankeerth Mahurkar
Electronics and Communications Engineering, Institute of Aeronautical Engineering
Hyderabad, India.

Akhila G
Computer Science and Engineering
Institute of Aeronautical Engineering
Hyderabad, India.

***Abstract*—With the rise of Internet Technology, IoT has advanced significantly. IoT is often integrated with other data-intense technologies such as Machine learning and Cloud computing. This integration makes the overall system too complex and sophisticated. Many different proposed solutions are available for various kinds of issues like security, scalability, and reliability. But these solutions demand their unique knowledge base, which makes it tough to implement. To overcome these issues, this paper focuses on a method to abstract complex server tasks.Abstraction of server tasks allows us to focus more on the primary goals in designing an IoT System. This paper describes how to make use of ready-to-use platforms to ease our tasks. One such platform is Firebase from Google. Google Firebase gives a one-step solution for almost all the major problems. In this paper, an IoT system using various services provided by Firebase is implemented. Anyone can use it as a base system and extend it according to their use case.**

Keywords—IoT, IoT System Architecture, Firebase, Security, Scalability.

I. INTRODUCTION

Internet of things is a trending technology that enables communications between smart embedded devices such as sensors, actuators, and wearables over the internet. These devices are also known as things. IoT is emerging rapidly. It is estimated that around 18 billion IoT devices would be connected to the internet by 2022 [1]. These "Smart devices" in IoT are often connected to the digital cloud to store the data generated by them. This data is further utilised for analytics by the consumers. Other technologies such as Big Data, Machine Learning and Blockchain go hand-in-hand with IoT. This combination results in a more functional and sophisticated system.

In general, an IoT system consists of a controller, sensors and actuators, a message broker, a data storage system, and a consumer dashboard [2]. Controllers maintain and manage other IoT devices such as sensors. These controllers exchange data with the external world. Message broker offers pub/sub model messaging service. Controllers exchange data with message brokers. Data storage systems are specially designed databases for big data and store the data generated by IoT devices. Finally, the consumer dashboard is a web portal or an app that consumers could use for managing their IoT systems.

TABLE I. LIST OF MAJOR IOT COMPONENTS

S. No	Component	Examples
1	IoT Controller	Raspberry Pi [3], Arduino [4]
2	Message Brokers	MQTT [5], XMPP
3	Database Clusters	Hadoop [5], HBase
4	Consumers' Portal	Mobile Apps, Web Application

Apart from these physical components, there are certain architectural principles. Some of these principles are component decoupling, access management system, data transmission security, and scaling. Applying these principles in designing results in an efficient, secured, and scalable IoT system. Component decoupling is a technique that prevents service blockage due to device failure. Access management system defines who is allowed to perform what task in the system. It protects the system from tampering by malicious consumers. Data transmission security enables encrypted data exchange between components. Scalability ensures the minimum latency and availability as the consumer base grows. The data generated by IoT systems satisfy 3Vs of Big Data. Therefore, IoT data management falls under Big Data management. It requires the distributed database cluster to store this data. Examples of these database clusters are Hadoop, HBase. With these many considerations, designing efficient IoT systems has led us to many varied challenges [7]. Many complex and sophisticated architectures are in use to build efficient and scalable IoT Systems

Our research aims to design a highly scalable IoT system that abstracts all the complex server tasks. Innovations in the digital cloud in recent years have brought the platforms that provide various solutions as services. One such platform is Firebase from Google [8]. It has several products that anyone can use to reach the above objectives. The research paper provides an implemented prototype of an IoT System that serves as a boilerplate for IoT-based project works. This prototype could be easily customised and extended as per the project use case. The upcoming sections are as follows. In Section II, we provide detailed

information about Firebase and discuss the related work. In section III, we describe our proposed work and provide in-depth details about the implementation. In section IV, we verify the result of the prototype and analyse it thoroughly. In section V, we conclude our work and provide the direction for future work and extension of this implementation.

II. Related Work

With the rapid rise in IoT technology and as an active research topic, there is good literature availability upon implementation and challenges [2], [9], [10]. There are many proposals available for the architecture of IoT Systems. Harsh Kumar Singh (2019) in [9] proposed the implementation of home automation using IoT. Their architecture model uses a centralised server and database. With the increase in the number of connected devices, the system requires too much server maintenance and resource costs [11]. Jay Lohakare (2017) in [10] proposed a highly scalable home automation system architecture. But due to the use of a complex ecosystem, it is hard to set up, and it demands too much configuration and maintenance.

Firebase is a platform for developers developed by Google [8]. Its feature-rich services have made server-side development too easy and gained a good amount of popularity. The platform has many different kinds of products for almost any type of problem. It provides hosting, database services, cloud functions, analytics engine, cloud messaging, and much more. Its clear documentation and simple-to-use services have made it highly popular. Apart from these products, it goes hand-in-hand with the Google Cloud Platform [12]. This integration with GCP enables Data Warehousing and Big Data management much more accessible [13]. This platform is worth considering for IoT development. Unfortunately, not much literature is available on using Firebase for IoT System development. The use of Firebase for IoT is still in the experimental stage.

In the paper "JustIoT Internet of Things based on the Firebase Real-time Database" (2018) by Wu-Jeng Li [14], the authors have implemented an IoT system using this platform. They used Firebase Realtime Database (RTDB) [15] as the primary database. Firebase Database demands to keep the data tree as flat as possible. Too much data branching leads to poor performance of the database [16]. Hence to overcome this problem, we propose using an alternate NoSQL database from Firebase called Firestore. But Firestore does not support real-time data streaming through REST APIs. To make the system more reliable, we used Firestore in combination with RTDB [17].

TABLE II. Existing Implementations and their Limitations

Ref	Technology Used	Limitations
Harsh Kumar Singh et al. [9]	Nodejs JS and MongoDB	Scalability, High Maintenance is needed with the growth of connected device
Jay Lohakare et al. [10]	MQTT, Apache Kafka, Apache Spark, and MongoDB	Complex design and demands too much configuration and is tough to maintain.
Wu-Jeng Li et al. [14]	Firebase Real-time Database	Use of RTDB as the primary storage.

III. Proposed Work

The proposed IoT system mainly consists of an IoT network, Firebase cloud, and the consumer portal. An IoT network is the part that contains a controller and other devices connected to it. A consumer portal is an application that allows consumers to view their IoT network status and interact with it.

A. System Details

The IoT network consists of the following components.

- Raspberry Pi is used as the IoT network controller.
- 3 LED bulbs and push-button are used as IoT Devices.
- A python program that runs inside Raspberry Pi and does the following tasks:
    - Receives incoming commands from the consumer and does the relevant operation.
    - Operates the other devices and changes their state based on the events.
    - On pressing the push button, it generates the alert and updates it in the Firestore.

Here, we used push-button to simulate an alert. It can be replaced with alert sensors like a flame sensor. The consumer portal consists of a login page that authenticates the user to perform the actions. The routes of the web application for consumer portal are:

- /auth/login page, where the user can log in.
- /home page, where the user can view the status and monitor it.

Firebase cloud serves as the back-end. The primary services used for this system are the Firebase Realtime Database (RTDB) [18] and Firebase Firestore [19]. They serve two purposes:

- Firebase RTDB creates a real-time channel between Raspberry Pi and the consumer dashboard.
- Firestore stores the data that the system generates.

B. System Flow

When the user login successfully, the application sends the request via RTDB to raspberry pi, asking the current status. Then raspberry pi sends the response to the application by updating data in Firestore.

Similarly, when the user wants to change the device state like turning on/off the lights, the application sends the request to the pi via, and pi acts and acknowledges the application.

Whenever an alert is required to send, pi writes it to the Firestore. Then the cloud function triggers this update from Firestore. The cloud function then uses the FCM service to send push notifications to consumers.

C. *Implementation Details*

For the consumer portal, to develop the login mechanism, Firebase authentication [20] is used. It is a ready-to-use service and doesn't require too much effort. And in the dashboard, the system status is fetched through RTDB and Firestore.

Fig. 1 below shows the circuit diagram of the IoT Network.

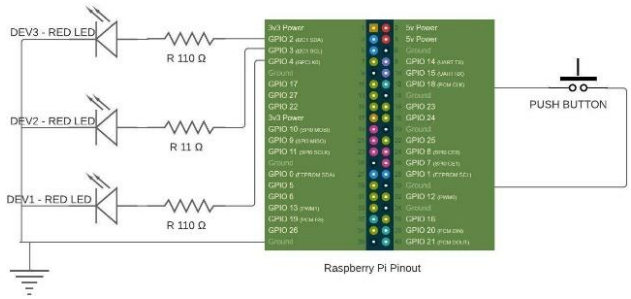

Fig. 1. The IoT network using Raspberry Pi 3B+

In this circuit diagram, 3 LEDs are interfaced to the raspberry pi on pins 3, 5, and 7. We can control these LEDs remotely. A push button is connected to pin 12. Whenever we press this push button, an alert notification is receivedthrough the consumer portal. The specifications of the components used in the circuit diagram are given in the below table.

Raspberry Pi is programmed using Python. It is a multi-process application program and consists of four processes. They are:

- Bootstrap process, which maintains the auth state and refreshes the access token. This process also relaunches the other sub-processes with the updated access token.
- The second process is a sub-process that actively listens to requests asking for network status and sends the network status. Whenever a consumer requires network status, they update it in RTDB.
- The third process is a sub-process that actively listens to requests asking to change the device states and updates the device states accordingly.
- The fourth process is also a sub-process that gets triggered on pressing the push button. It updates this alert in the Firestore.

The Raspberry pi and Consumer application interact through RTDB.

1) *To fetch network status:*

The consumer portal first creates a document in Firestore at path /devices/uid/states. Here uid is the user's unique id provided by Firebase Auth. It returns the id of the newly created document, say doc_id. Then add this doc_id in RTDB at the path requests/uid/request/get. RPi process listens to changes in RTDB through SSE protocol [21]. The procedure to listen and send network status in Raspberry Pi is given below.

- Frame the URL to make a REST call to RTDB. The URL is of form
- Now make the REST call as an SSE Client to stream data in real-time.
- Whenever the data updates, check the path of data updated, and lastly, get the doc id of Firestore to update.
- Create the HashMap object for firestore. Add devices' state to this HashMap.
- Update this document as the HashMap object using Cloud Firestore REST API.

2) *To change the device state:*

The RTDB maintains the state of devices at the path /states/{uid}/. RPi process listens to changes at this path using SSE protocol [21] and changes the device's state accordingly. The procedure to listen for requests to change device state in the Raspberry Pi program is given below.

- Similar to the procedure above, frame the URL to make a REST call to RTDB.
- Now make the REST call as an SSE Client to stream data in real-time.
- Whenever the data updates, get the device id and its state to be changed.
- Update the GPIO PIN of raspberry pi accordingly.

IV. IMPLEMENTATION DETAILS

After the text edit has been completed, the paper is ready for the template. The above-proposed system was built and named FireIoT. This system has worked as intended. Consumer's portal was tested on two devices simultaneously, and both of them have maintained the sync of network status. The portal was able to retrieve the network status successfully.

LED states were changing while toggling them on the consumer's portal. This action was instantaneous without significant latency. Figure 2 shows the portal screenshots.

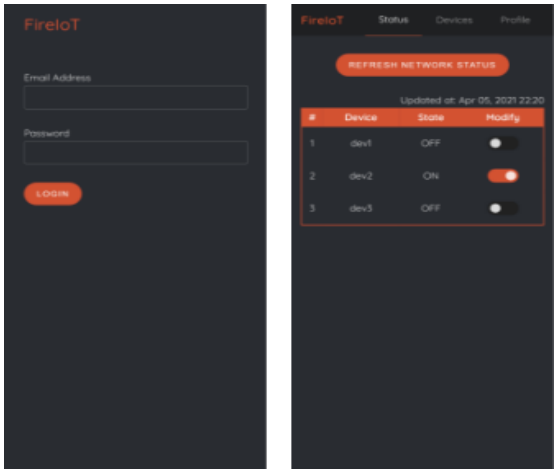

Fig. 2. Screenshots of the consumer portal.

Figure 3 is Activity happening in the raspberry pi. It is displaying the logs from the python program.

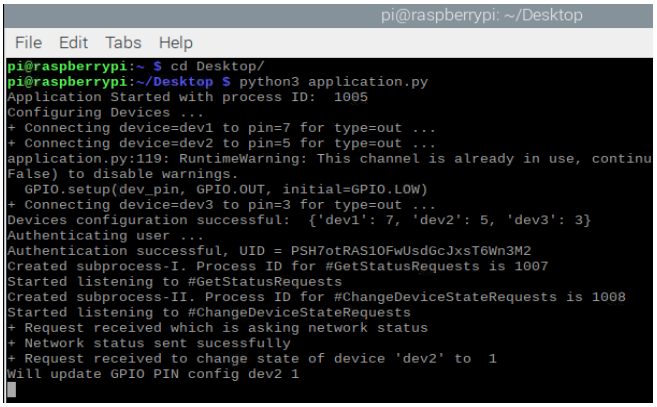

Fig. 3. The activity log of the python program that running in Raspberry Pi.

After the above experimentation, the Alerting System was tested. As intended, upon pressing the push button, the alert notification was received on the consumer devices. Figure 4 shows the push notification received on the consumer device.

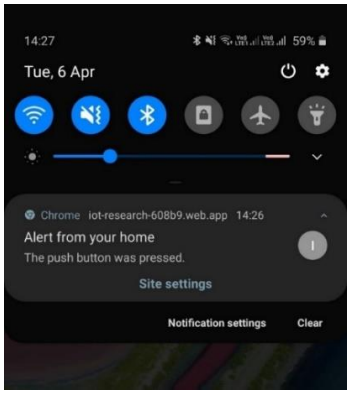

Fig. 4. Screenshot of Notification being received to Consumer Device.

V. CONCLUSION

As per the objective of this research, the complex server tasks were abstracted, and it was pretty easy to create the Firebase project and work with it. In this paper, an implementation of the IoT System using Firebase was done. Because of the use of the serverless platform, it can scale to any number of devices. Firestore was used in combination with the RTDB to increase the reliability of the system. RTDB served as the real-time channel between IoT Controller and Consumer portal. Firestore served as the database to store a large amount of data. We also developed an alerting system using Firebase Cloud Messaging. One could easily extend this system as per their use case. Firebase is continuously evolving and getting more feature-rich. They have also released the Machine Learning platform. Hence the integration of machine learning in this IoT system is much more effortless. Future work includes the integration of ML into this IoT system using Firebase.

REFERENCES

[1] P. Colella, "Ushering in A better connected future," Businessworld.in. [Online]. Available: http://businessworld.in/article/Ushering-In-A-Better-Connected-Future/16-01-2017-111504. [Accessed: 16-May-2021].
[2] B. Mishra and A. Kertesz, "The use of MQTT in M2M and IoT systems: A survey," IEEE Access, vol. 8, pp. 201071–201086, 2020.
[3] "Teach, learn, and make with Raspberry Pi," Raspberrypi.org. [Online]. Available: https://www.raspberrypi.org/. [Accessed: 16-May-2021].
[4] "Arduino - Home," Arduino.cc. [Online]. Available: http://arduino.cc. [Accessed: 16-May-2021].
[5] "MQTT - The Standard for IoT Messaging," Mqtt.org. [Online]. Available: http://mqtt.org. [Accessed: 16-May-2021].
[6] "Apache Hadoop," Apache.org. [Online]. Available: http://hadoop.apache.org. [Accessed: 16-May-2021].
[7] S. A. Goswami, B. P. Padhya, and K. D. Patel, "Internet of things: Applications, challenges and research issues," in 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019, pp. 47–50.
[8] "Firebase," Firebase Official Website. [Online]. Available: https://firebase.google.com/. [Accessed: 16-May-2021].
[9] H. K. Singh, S. Verma, S. Pal, and K. Pandey, "A step towards Home Automation using IOT," in 2019 Twelfth International Conference on Contemporary Computing (IC3), 2019, pp. 1–5.
[10] J. Lohokare, R. Dani, A. Rajurkar, and A. Apte, "An IoT ecosystem for the implementation of scalable wireless home automation systems at smart city level," in TENCON 2017 - 2017 IEEE Region 10 Conference, 2017, pp. 1503–1508.
[11] A. Banafa, "3 Major Challenges IoT is Facing," Bbvaopenmind.com, 21-Mar-2017. [Online]. Available: https://www.bbvaopenmind.com/en/technology/digital-world/3-major-challenges-facing-iot. [Accessed: 16-May-2021].
[12] D. Stevenson, "What's the relationship between Firebase and Google Cloud?", Google Developers, 13-Jan-2019. [Online]. Available: https://medium.com/google-developers/whats-the-relationship-between-firebase-and-google-cloud-57e268a7ff6f. [Accessed: 16-May-2021].
[13] "Firebase Extensions." [Online]. Available: https://firebase.google.com/products/extensions/firestore-bigquery-export. [Accessed: 16-May-2021].
[14] W.-J. Li, C. Yen, Y.-S. Lin, S.-C. Tung, and S. Huang, "JustIoT Internet of Things based on the Firebase real-time database," in 2018 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE), 2018, pp. 43–47.
[15] "Firebase Realtime Database." [Online]. Available: https://firebase.google.com/products/realtime-database. [Accessed: 16-May-2021].
[16] "Structure Your Database." [Online]. Available: https://firebase.google.com/docs/database/web/structure-data. [Accessed: 16-May-2021].
[17] "Choose a database: Cloud firestore or realtime database." [Online]. Available: https://firebase.google.com/docs/database/rtdb-vs-firestore. [Accessed: 16-May-2021].
[18] "Installation & Setup in JavaScript." [Online]. Available: https://firebase.google.com/docs/database/web/start. [Accessed: 16-May-2021].
[19] "Get started with cloud firestore." [Online]. Available: https://firebase.google.com/docs/firestore/quickstart. [Accessed: 16-May-2021].
[20] "Get started with firebase authentication on websites." [Online]. Available: https://firebase.google.com/docs/auth/web/start. [Accessed: 16-May-2021].
[21] Wikipedia contributors, "Server-sent events," Wikipedia, The Free Encyclopedia, 09-May-2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Server-sent_events&oldid=1022333423. [Accessed: 16-May-2021].