Table of Contents

1. Introduction to Conceptual IOT Application

Selected Home Application:

“Smart thermostat to control heating and hot water based on household consumption behaviors”.

The temperature settings for heating and cooling in your home are automatically fixed by a smart thermostat for best result. Based on actual field data, smart thermostats that receive the ENERGY STAR designation have been independently certified to save energy. The main consumer of energy in homes is heating, ventilation, and cooling. Here, we show how to automatically detect occupant and sleep patterns in a home using low-cost and straightforward sensing technology, and how to use these patterns to automatically shut down the system of the home in order to conserve energy. This strategy is referred to as the smart thermostat. By installing sensors in 8 homes and contrasting our algorithm's anticipated energy consumption with that of other methods, we evaluate this strategy (Mandlem et al., 2020).

With 61% of residential energy consumption in Canada and the U.K., which have colder climates, heating, ventilation, and cooling is the single largest contributor to a home's energy costs and carbon emissions. According to studies, shutting off the heating system while residents are away or asleep could save 20–30% of this energy. These savings, however, have been challenging to achieve because most homeowners do not manually regulate the thermostat numerous times each day, and most people find it too challenging to utilise programmable thermostats properly. The Energy Star certification programme for all programmable thermostats was consequently recently discontinued by the EPA, effective December 31, 2009 (Rau et al., 2016).

The meagre financial incentive for individual homes is a significant barrier to energy saving. The average American home would save $15 per month with a 20–30% reduction in energy costs. For many people, this meagre financial gain is insufficient to outweigh the challenges of everyday operation optimization. To solve this problem, a fresh approach should be developed that "simply works" and it conserves energy without requiring consideration on the daily basis or action from family occupants (Kull et al., 2020).

Since immediately after their creation in 1906, more than a century ago, smart thermostat have been a cornerstone of energy conservation programmes which was programmable. The fundamental concept is to operate this system according to the schedule. This strategy, however, wastes energy in a number of ways. Let's imagine that the following scenarios occur: First, the residents vacate the property shortly after nine in the morning, but the heating system wastes energy because it is set to run until 10 o’ clock in the morning. Secondly, even when the house is empty, energy is consumed since the setback temperature is far higher than the safety limit for the building (center). Typically, this kind of short setback is utilised to lessen the possibility of comfort loss in the event. Thirdly, because the system is not planned to heat the house until much later, the residents feel uncomfortable when they arrive shortly in the afternoon. The static setback schedules used by smart thermostat cannot capture the highly dynamic occupancy

patterns of some of the homes, and due to this house becomes less comfortable. On the surface, this last issue appears to be only a comfort issue, but it is actually a significant source of energy waste. People who use setback schedules less frequently or give them up altogether do so because of the possibility of losing comfort. According to reports, more than 50% of homes with programmable thermostats don't employ setback periods at night or during the day (Ayan and Turkay, 2018).

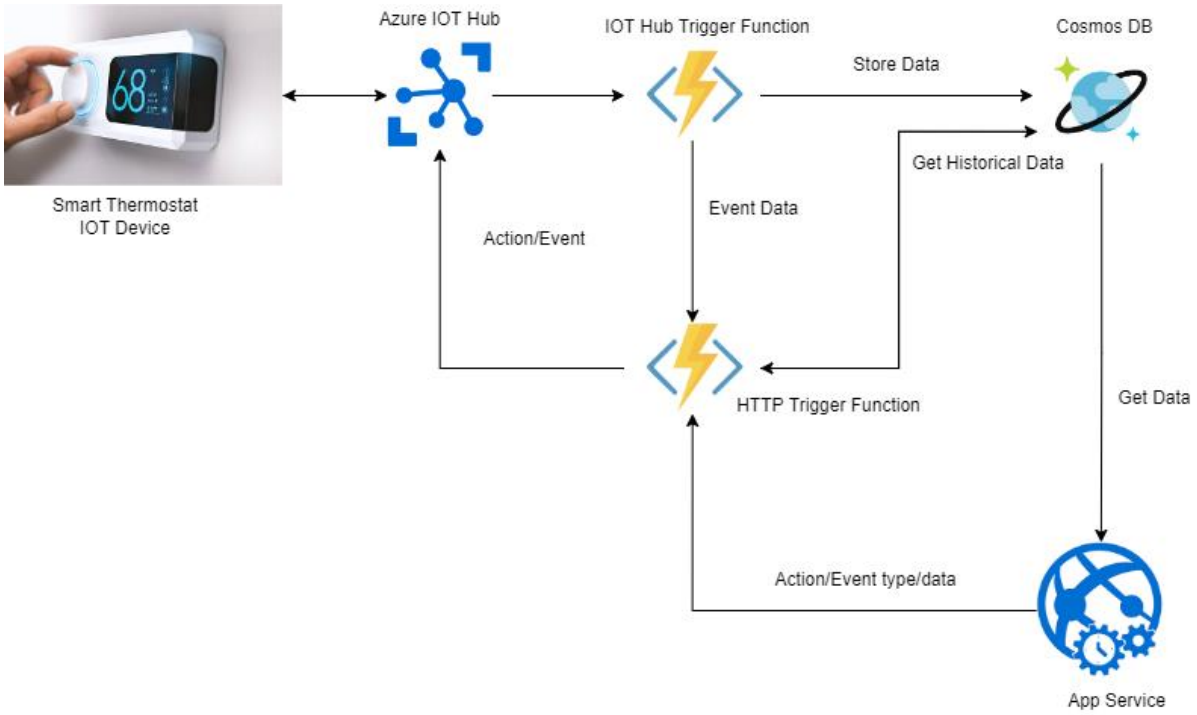
As a result, households using the less complicated dial-type thermostats actually save more energy on average than those who use programmable thermostats. This is because they can quickly modify the temperature settings before going to bed or leaving the house. To address this issue, two of the authors created and assessed a self-programming thermostat in the preliminary study. This thermostat automatically selects the best setback schedule based on past occupancy data. However, that method still generates a static timetable, and given that daily occupancy patterns change, every static plan must make a trade-off between comfort and energy. As the occupancy state of the house changes, we employ real-time sensor data in this research to dynamically control the heating system. Space heating accounts for the majority of the energy consumed in the residential sector, which uses almost one third of all energy. Understanding a home's thermal characteristics can help homeowners become more aware of energy-saving choices by, for instance, highlighting the need for increased insulation. However, accurate thermal studies are required in order to calculate the correct value of these features (Yang and Newman, 2012).

With features like huge colour screens, the capacity to learn consumption patterns, and an internet connection that allows for remote control and online information, more and more thermostats are becoming intelligent appliances. These capabilities are frequently also employed to give users access to information on household energy consumption. Two options exist for smart thermostats to support more cost-effective energy use. On the one hand, they can manage the heating system economically, such as by deciding not to heat the house if they notice no one is home. On the other hand, they are crucial in helping the residents of the house become aware of how much energy they are using, for instance by letting them know where and when the energy is consumed and how it might be able to cut back. By doing this, they urge consumers to alter their heating habits or to make their homes more energy efficient, such as by insulating them (Marantos et al., 2018).

An app-enabled smart thermostat enables homeowners to remotely manage the temperature of their homes from other internet-connected devices. This device can be utilised in residential, commercial, and industrial settings and can be connected using wired or wireless technologies. In our situation, we are leveraging the Internet of Things to connect smart thermostates and control the heating and hot water. Smart thermostats assist customers identify energy waste by revealing that the amount of fossil fuel used may be greater than what is actually required. Such revelations result in behavioural adjustments that lower costs, lower energy usage, and lower the demand for fossil fuels that are bad for people and the environment. By maintaining the temperature as low as possible for as long as possible, smart thermostats reduce energy costs. Usually, you specify the temperature you'd like it to be and a description of your daily plan (Ozgur et al., 2018).

2. Flow Diagram

We are using below cloud Architecture Diagram for the implementation of this conceptual IOT Application.



Flows of the System

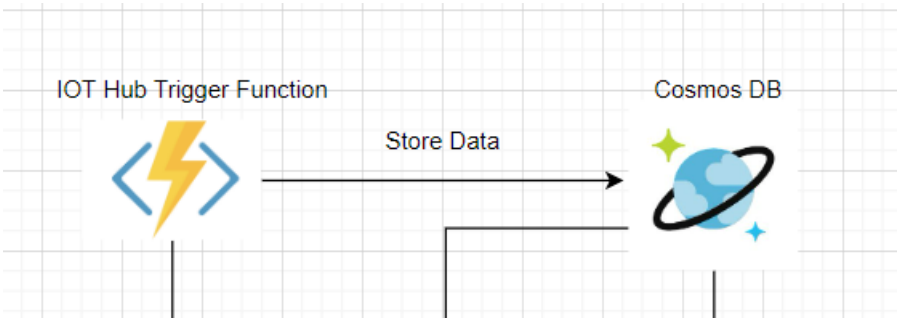
- Smart Thermostat IOT Circuit App to Azure IOT Hub.
- IOT Hub Trigger Azure Function to Azure Cosmos DB.
- HTTP Trigger Function from Cosmos DB to Azure App Service.
- HTTP Trigger Azure Function from Cosmos DB to Azure IOT Hub.
- HTTP Trigger Azure Function from App Service to Azure IOT Hub.

2.1.1 Smart Thermostat IOT Circuit App to Azure IOT Hub

Firstly, the sensor device that is attached with Smart Thermostat generate the data and sent to the IOT Hub that will hold the data in its own environment. Then an automatically trigger Azure function Azure IOT Hub Trigger Azure Function will be called that will get the data from the IOT Hub and send it to the Cosmos DB to store the data for future analysis and prediction.

2.2.2 IOT Hub Trigger Azure Function to Azure Cosmos DB

As IOT Hub Trigger Function is a type of Azure serverless functions that is automatically triggered when IOT Hub received any data from the Thermostat device. Its automatically triggered and get data and call the Cosmos DB API to send the data to the Cosmos DB to save it for future use.

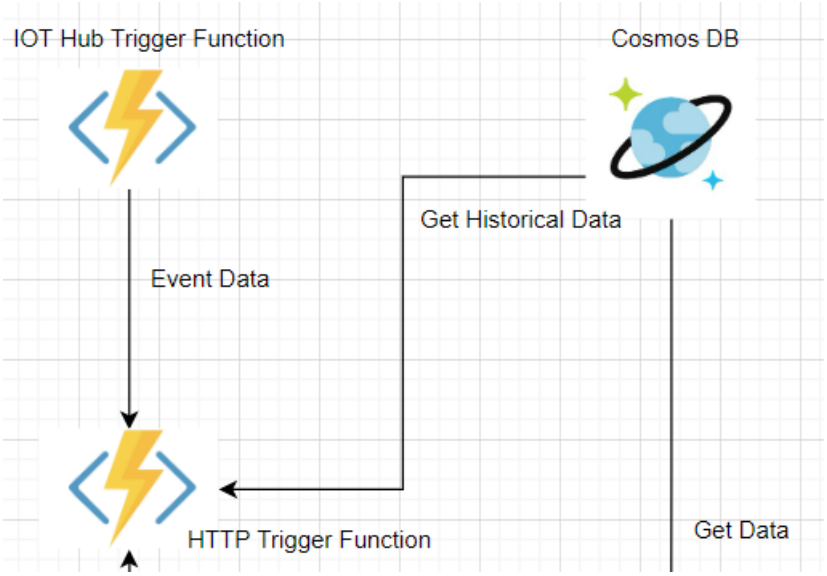


2.2.3 HTTP Trigger Azure Function to Azure App Service

"IOT Hub Trigger Function" is a type of azure function that will execute when an HTTP request is processed. It gets data from the Cosmos DB and the show the data onto the web page in graphs and charts for data visualization.

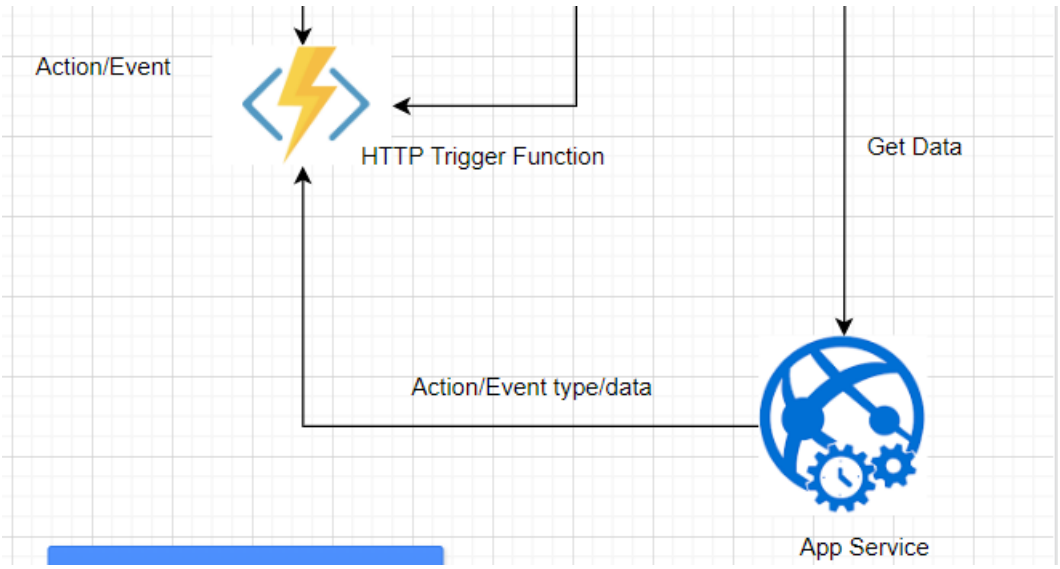
2.2.4 HTTP Trigger Azure Function from Cosmos DB to IOT Hub

As we are using household consumption behavior to control the Thermostat device temperature so we need the past data to use it for future prediction. So, we will use all the days temperature data and predict current days temperature and consumption behavior.



2.2.5 HTTP Trigger from App Service to IOT Hub

We also can control our thermostat sensor device from a controlled device, in our case It will be a web app that will get the data from the Cosmos DB and display the data on the web page. We will also add some buttons on the web page to control thermostat device. We just have to press on specific buttons and it will automatically trigger the HTTP Trigger Azure Function and then HTTP Trigger Azure Function will call the Azure IOT Hub to update the Thermostat state.



3 Azure Cloud Platform Services

Below is the list of services that we are using for the implementation of this Conceptual IOT Application.

1. Azure IOT Hub
2. Azure Cosmos DB
3. Azure Serverless Functions
4. Azure Web Apps

Explanation

3.1 Azure IOT Hub

The Internet of Things (IoT) offers companies an immediate and real opportunity to reduce costs, increase revenue and transform their business. Azure IoT Hub is a managed IoT service hosted in the cloud. It enables two-way communication between an IoT application and the devices it manages. This cloud-to-device connection means that data can be received from the device, but commands and policies can also be sent back to the device. Azure IoT Hub differs from existing solutions in that it also provides infrastructure for authenticating, connecting, and managing connected devices (Halabi et al., 2021).

Azure IoT Hub enables fully functional and scalable IoT solutions. Connect almost any device to Azure IoT Hub and scale to billions of devices. Events can be tracked and monitored. B. Device Creation, Failure, and Attachment (Familiar and Barnes, 2017).

- Device libraries for the most commonly used platforms and languages for easy device connectivity.
- secure communications with multiple options for device-to-cloud and cloud-to-device hyperscale communications.
- Query able storage of per-device state information and metadata.

Usage

We will use Azure IOT Hub to trigger different events and to receive data to and from the IOT Hub. We can send commands to Smart Thermostat and also receive data from the Smart Thermostat. IOT Hub has its own infrastructure space that can hold the data in this own environment. Azure IOT Hub is fully managed and customized solution to build an Internet of Things Application that can interact with the world with different sensors to receive the data. For IOT the device should attached with the Internet 24/7 (Klein, 2017).

Here we are using Azure IOT Hub to receive the data from the Smart Thermostat IOT Device based on different events and actions to receive data from a specific sensor device, and save the data onto the Cosmos DB for future use.

We will configure the Keys, Endpoints and action/events in the Azure IOT Hub settings to connect it to the target sensor device and to the Azure IOT Hub Trigger Function. Bases on action types and events we can receive a specific sensor data and send the commands to a specific sensor. We can also control the sensor device by custom events by calling the frontend buttons as actions.

3.2 Cosmos DB

Azure Cosmos DB is a fully managed globally distributed NoSQL database for modern app development. Unmatched millisecond response times and instant automated scalability ensure speed at any scale. App Enhancement is a fast and highly efficient way to turnkey distribute

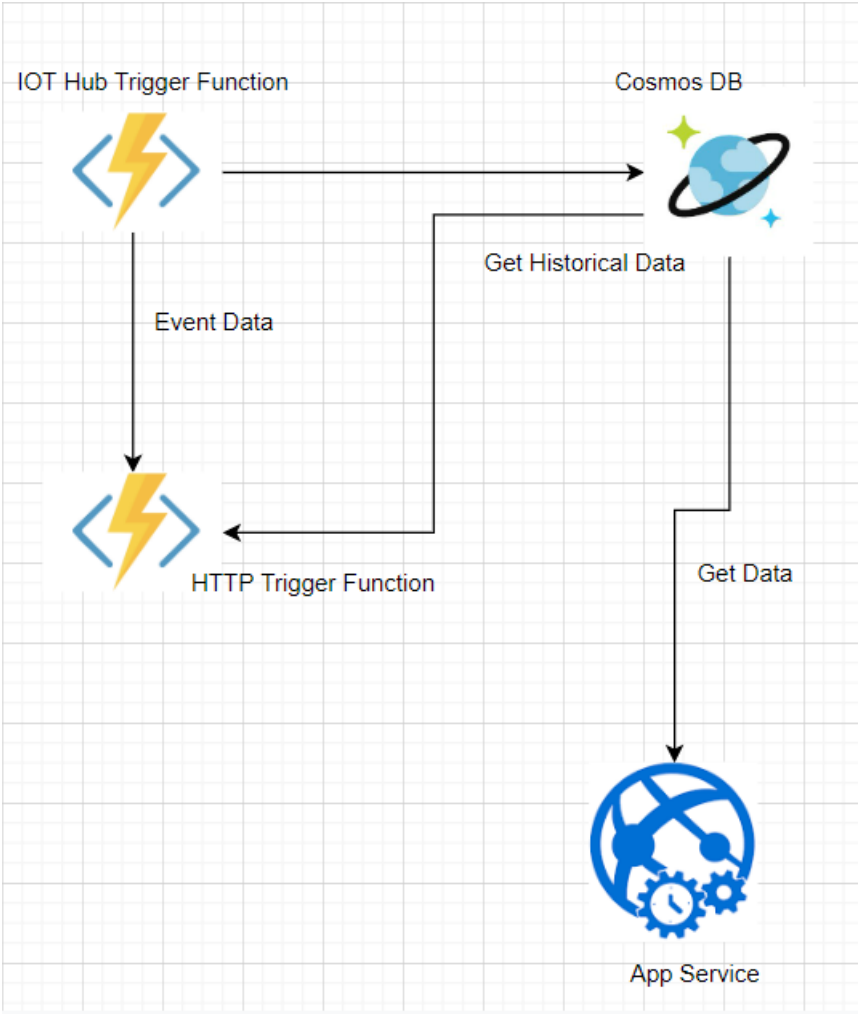
information to multiple locations within global open-source APIs and SDKs in popular languages. A fully managed service, Azure Cosmos DB handles database management with computerized control, updates, and patches. It also does capacity management with valuable serverless and autoscaling options that match capacity to demand as your software demands. Multiple data models can be supported in Cosmos DB (Sahay, 2020).

Features

Easy to use: This provides a complete product that runs on Azure and can be replicated regularly to international clearinghouses.
API's: Fact sets are enumerated periodically and can be accessed by users using any API. For example SQL API, MongoDB API etc.
Consistency: It uses unique integrity levels: limited obsolescence, strong, advisory, final, and stable prefixes.
Latency: With <10ms reading stats and <15ms writing stats, very low latency is almost guaranteed.

Usage

As Cosmos DB is a globally distributed database and it is fast and it have highest availability and also have different availability zones. We are using it to store the Thermostat data. As Cosmos DB is a no SQL database so you don't have to worry about Transforming the data into a specific format. The data that Cosmos DB SQL API receive; Cosmos DB saves the data in the same format as the data is receiving.



Data Template

The Thermostat sensor device will have the following data:

```
{
SensorDeviceName:String,
CurrentTemperature:Int,
DateTime:DateTime,
WaterLevel:Int
}
```

We will use above data for the implementation of our conceptual IOT Application. We will get the Sensor Device Name, Current Temperature and Date Time etc. attributes.

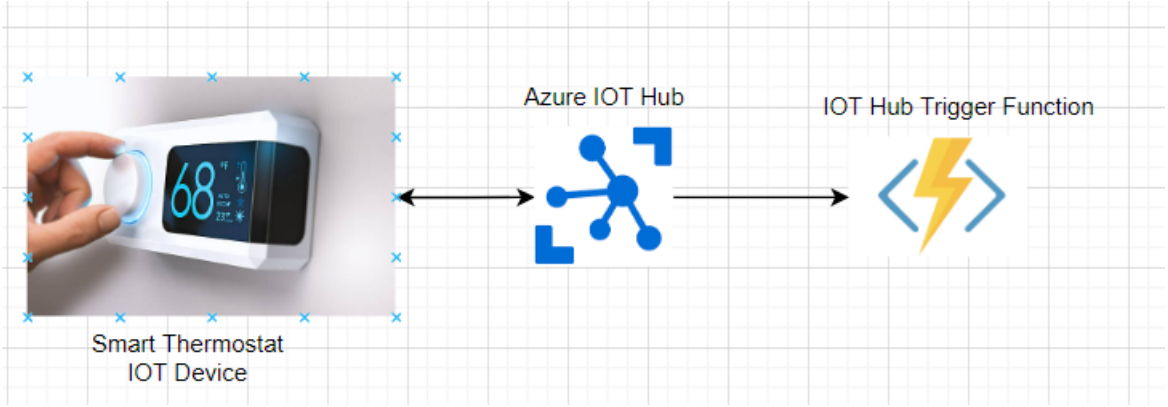
3.3 Azure Functions

Azure Functions is a serverless concept with a cloud-native design that allows you to deploy and run code without lacking server infrastructure, web servers, or another configuration. Azure Functions are written in multiple languages, including TypeScript, C#, Java, JavaScript and Python. Azure Functions are scalable. As execution demand increases, more resources are automatically allocated to the provider, and as demand decreases, all large resources and application instances are automatically reduced. Azure skills are of acceptable quality as small apps may run independently from various websites. Common Azure functions include sending emails, initiating backups, processing orders, project planning including database clean-up, sending notifications, messaging, and processing IoT statistics (Guay Paz, 2018).

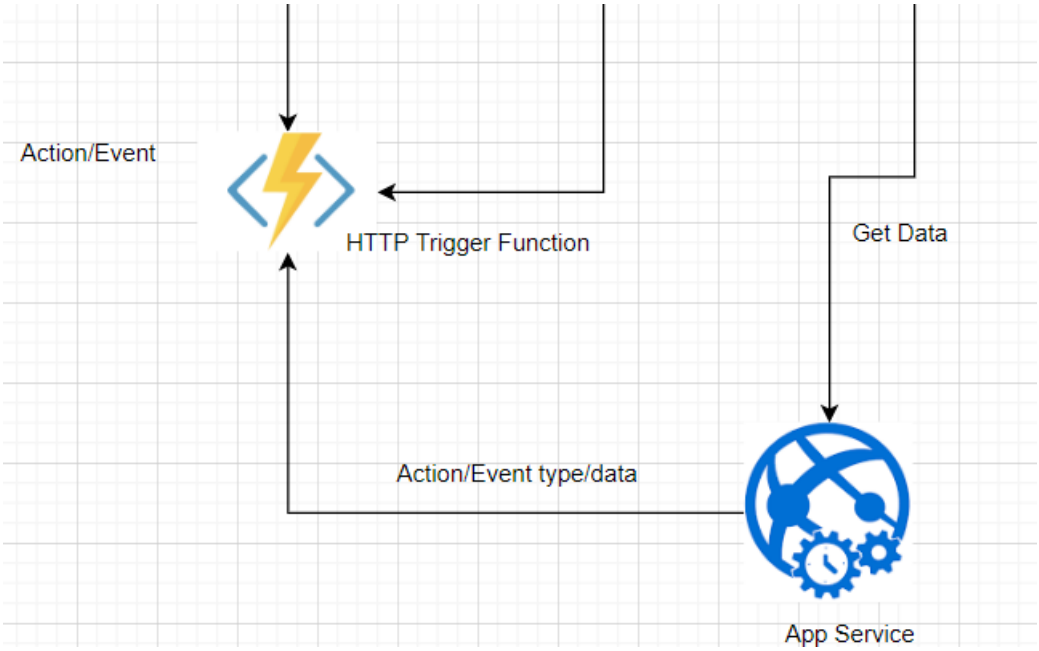
Usage

As shown in the Design Diagram Section 2, We are two types of Azure Functions that we are using for the implementation of this Conceptual IOT Application.

- **IOT Hub Trigger Azure Function**, We will use this function to get the data from the Azure IOT Hub in a real-time trigger when Thermostat device receives the data. It will receive the data and send the data to the Cosmos DB to save it for future use and for prediction purposes.



- **HTTP Trigger Azure Function**, It is a type of Azure Function that will trigger when an HTTP request is called. We are using it to get the data from the Cosmos DB and for getting the events and actions from the app control device and automatic events call. All the events call we be gone to the IOT Hub through this HTTP Trigger Azure Function.



3.4 Azure App Service

Azure App Carrier uses our own cloud services to rapidly build apps, quickly and easily build organization-centric web and mobile apps for any platform and tool, and make them scalable and reliable. It helps you deploy and install on high-performance cloud infrastructures. Azure Web Apps is a cloud computing-based website hosting platform built and operated by Microsoft. It's a platform-as-a-service that runs in multiple frameworks and lets you publish web apps written in a variety of programming languages, including Microsoft's own and third-party languages (Chilberto et al., 2020).

Usage

We will be using Azure App Service Plan to build a very beautiful web page that will show the data into different sections as bar charts and attractive UI's.

The Web page also have some buttons to control the Thermostat device behavior. Some actions are triggered automatically and some user

can call by just pressing the buttons on the UI's side of web site. We will use the Angular Framework for the implementation of this Application.

4 Security Considerations

The Security considerations that we will use during the implementation of this conceptual IOT Application are as follows:

1. Efficient Access Control
2. Use of updated software's
3. Data Encryption
4. Handling over to Application vulnerabilities
5. Physical security

Explanation

4.1 Efficient Access Control

Only the owner of the application and others trusted persons should have access to the services that this Conceptual IoT Application uses. IoT devices can trust on the local network to the point that they don't need to get any further authentication. The devices on the same local network are also trusted. The device should be connected to the internet. The device's capability may be accessed by all over the world (Harnal and Chauhan, 2018). So, we have to restrict and allow the owner of the IOT Device to use it and block any other access.

4.2 Use of Updated Software's

Updated software comes with after the removal of all the latest security threats, so by just using the latest software the changes of security breach will least a lot. So, it's always a good idea to use the updated software's. As this is a very big issue, maybe there is an issue in the last update and the issue is fixed in the newer release so we have to use the latest software that is up to date. A proper updating of software's also can eliminate security risks that are often happened by the use of outdated software's (Mugarza, Parra and Jacob, 2017).

4.3 Data Encryption

Encryption is the process of scrambling information. This process transforms the original representation of information, called plaintext, into another form called ciphertext.

As data is everything in the digital era. So also have to secure our data that no one can have access to this data. Some data encryption techniques are used to encrypt the data like SHA256 and MD5 etc. So, we encrypt the data before it can send over a network. Let's assume an encrypted data is traversing over the internet and an attacker attacks the data and gain access to view the data, but as data is encrypted by encryption algorithm so only encrypted data will be visible by the attacker. Attacker not able to decrypt the data as data is encrypted by using an encryption key that owns the application owner (Data Encryption Algorithms, 2016). So, by encrypting the data we can increase the security of our application.

4.4 Handling over to Application vulnerabilities

Proper software testing does not mean that it is totally error free or it not have any issue in future. As the technology increase, the hackers are also the parts of this digital era. They always found the ways to attack the application and we have to project that way, so complete Black and while box testing is the solution to overcome this threat.

Security flaws are inevitable while creating software, just like any other kind of bug. There are many solutions to overcome the vulnerabilities to overcome the risks of becoming vulnerable. Best practices for preventing application vulnerabilities are included in this (Lepofsky, 2014). We will use consistent testing on our Conceptual IOT application to overcome this security threat.

4.5 Physical Security

An attacker can open a gadget and attack the hardware if they have physical access to it. For instance, by accessing the component's contents directly, he can get beyond protection software in memory. Debug pins that can be accessed after the device is opened may also be present, giving an attacker further options.

- Physical attacks need physical contact and only impact one device.

- If a device key shared by all devices of the same model is revealed and many devices are hacked, a physical attack might be successful. Physical security is a concern in this situation, but I think sharing keys across all devices is more crucial.
- So, the Smart Thermostat security should be increased that only the owner and the authentic of software company members can physically touch the device.

5. DevOps Testing Strategies

Below is the list of DevOps testing strategies that we will use for the testing of this IOT application.

1. Application Compatibility Testing
2. Usability Testing
3. Pilot Testing
4. Load Testing
5. Security testing

5.1 Compatibility System Testing

It is a type of testing in which we test our application parts to check whether they are compatible with each other or not. Some Sensor devices can only interact with Azure etc (us Saqib and Shahzad, 2018). So, we will select those devices and services that will be compatible with each other.

5.2 Usability Testing

The purpose of usability testing is to verify that the product's features, functionality, and overall purpose plan match what users want by observing how they use the product in real life. Check it out. Usability testing lets you know in advance about user behavior, needs, and expectations (Lewis, 2006).

So, we have to build the conceptual IOT Application by keeping in mind the user needs and behavior.

5.3 Pilot Testing

Pilot testing is a type of software testing that validates the system under real-time operating conditions. The purpose of pilot testing is to assess the feasibility, time, cost, risk, and performance of a research project.

Pilot testing is performed by a group of end users that can test the whole application on different parameters. Before publishing the Conceptual IOT Application, our application will be tested by a group of professionals and end users that will test the whole application in detail (In, 2017).

5.4 Load Testing

Load testing is a type of testing in which we test, how the system behaves under normal and heavy loads and determines whether the system can handle heavy loads when end-user demands are high (Menasce, 2002).

We will test whether our sensor device can generate the data every 24/7 under every condition or not and we also test whether IOT Hub can receive the data 24 hours a day or not.

5.5 Security Testing

The primary goals of security testing are to identify threats in your system, measure potential vulnerabilities in your applications, detect threats, and prevent your system from crumbling or being exploited (Felderer et al., 2016).

We will test the application security to check whether there is some loop whole or not from where an attacker can attack the application. We will eliminate all the security threats that can help an attacker to attack on the application.

6. References

Ayan, O. and Turkay, B. (2018). *Smart Thermostats for Home Automation Systems and Energy Savings from Smart Thermostats*. [online] IEEE Xplore. doi:10.1109/CEIT.2018.8751790.

Chilberto, J., Zaal, S., Aroraa, G. and Price, E. (2020). Azure Web Apps. *Cloud Debugging and Profiling in Microsoft Azure*, pp.147–175. doi:10.1007/978-1-4842-5437-0_5.

Data Encryption Algorithms. (2016). *Introduction to Network Security*, pp.45–92. doi:10.1002/9781119113102.ch2.

Familiar, B. and Barnes, J. (2017). Device Management Using IoT Hub. *Business in Real-Time Using Azure IoT and Cortana Intelligence Suite*, pp.95–126. doi:10.1007/978-1-4842-2650-6_3.

Felderer, M., Büchler, M., Johns, M., Brucker, A.D., Breu, R. and Pretschner, A. (2016). Security Testing. *Advances in Computers*, pp.1–51. doi:10.1016/bs.adcom.2015.11.003.

Guay Paz, J.R. (2018). Working with an Azure Cosmos DB Database. *Microsoft Azure Cosmos DB Revealed*, pp.61–123. doi:10.1007/978-1-4842-3351-1_3.

Halabi, W.H., Smith, D.N., Hill, J.C., Anderson, J.W., Kennedy, K.E., Posey, B.M., Ngo, L.B. and Apon, A.W. (2021). Viability of Azure IoT Hub for Processing High Velocity Large Scale IoT Data. *Companion of the ACM/SPEC International Conference on Performance Engineering*. doi:10.1145/3447545.3451187.

Harnal, S. and Chauhan, R.K. (2018). Efficient and Flexible Role-Based Access Control (EFRBAC) Mechanism for Cloud. *ICST Transactions on Scalable Information Systems*, 0(0), p.161438. doi:10.4108/eai.13-7-2018.161438.

In, J. (2017). Introduction of a pilot study. *Korean Journal of Anesthesiology*, [online] 70(6), p.601. doi:10.4097/kjae.2017.70.6.601.

Klein, S. (2017). *IoT Solutions in Microsoft's Azure IoT Suite*. [online] Berkeley, CA: Apress. doi:10.1007/978-1-4842-2143-3.

Kull, T.M., Penu, K.-R., Thalfeldt, M. and Kurnitski, J. (2020). Energy saving potential with smart thermostats in low-energy homes in cold climate. *E3S Web of Conferences*, 172, p.09009. doi:10.1051/e3sconf/202017209009.

Lepofsky, R. (2014). Web Application Vulnerabilities and the Damage They Can Cause. *The Manager's Guide to Web Application Security:*, pp.21–46. doi:10.1007/978-1-4842-0148-0_3.

Lewis, J.R. (2006). Usability Testing. *Handbook of Human Factors and Ergonomics*, pp.1275–1316. doi:10.1002/0470048204.ch49.

Mandlem, K., Gopalakrishnan, B., Nimbarte, A., Mostafa, R. and Das, R. (2020). Energy Efficiency Effectiveness of Smart Thermostat Based BEMS. *Energy Engineering*, 117(4), pp.165–183. doi:10.32604/ee.2020.011406.

Marantos, C., Lamprakos, C.P., Tsoutsouras, V., Siozios, K. and Soudris, D. (2018). Towards plug&play smart thermostats inspired by reinforcement learning. *Proceedings of the Workshop on INTelligent Embedded Systems Architectures and Applications*. doi:10.1145/3285017.3285024.

Menasce, D.A. (2002). Load testing of Web sites. *IEEE Internet Computing*, 6(4), pp.70–74. doi:10.1109/mic.2002.1020328.

Mugarza, I., Parra, J. and Jacob, E. (2017). Software Updates in Safety and Security Co-engineering. *Lecture Notes in Computer Science*, pp.199–210. doi:10.1007/978-3-319-66284-8_17.

Ozgur, L., Akram, V.K., Challenger, M. and Dagdeviren, O. (2018). An IoT based smart thermostat. *2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*. doi:10.1109/iceee2.2018.8391341.

Rau, P.-L., Gong, Y., Huang, H.-J. and Wen, J. (2016). A Systematic Study for Smart Residential Thermostats: User Needs for the Input, Output, and Intelligence Level. *Buildings*, 6(2), p.19. doi:10.3390/buildings6020019.

Sahay, R. (2020). Cosmos DB. *Microsoft Azure Architect Technologies Study Companion*, pp.699–716. doi:10.1007/978-1-4842-6200-9_20.

us Saqib, N. and Shahzad, S. (2018). Functionality, Performance, and Compatibility Testing: A Model Based Approach. *2018 International Conference on Frontiers of Information Technology (FIT)*. doi:10.1109/fit.2018.00037.

Yang, R. and Newman, M.W. (2012). Living with an intelligent thermostat. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*. doi:10.1145/2370216.2370449.