Integration of IoT sensor data to XR campus map application

Abstract— **Integration of Real time data to Extended Reality is an essential part in the AR/VR space. With the rise of XR HMD devices (Oculus Quest Pro, Apple Vision Pro) having XR applications that integrate real time data would be beneficial for users. Unfortunately, there is little implementation done in this area of XR because the AR/VR industry is relatively new in Computer Science field. The purpose of this study is to integrate real-time data collection from IoT sensor to the Extended Reality (XR) 3D campus map application. The application will be built for HMD devices such as Oculus Quest Pro and Apple Vision Pro headset which are accessible to the users. The study will showcase six steps in designing and integrating real time data to the XR campus map application. The first step is the design of the front-end of the application, the 3D model of the CSUDH campus map. The 3D Map will include all the CSUDH building locations inspired from the ToroGis Map. The second step is to install the Cisco IoT sensors in campus buildings (Campus Library, Parking Lot, and Cafeteria) to measure the environmental conditions such as temperature and humidity inside the campus buildings and measure the occupancy data of each campus building using the campus Wi-Fi network. The third step is to set up a simple server and store the data in a database in which an API will be created. In the next step we call the API data in Unity3D software which the application will be built to the HMD headset so that when the user selects a building, the fetched data (temperature, humidity and occupancy) will be displayed. The final step is to test the MR application via Headset with users for accuracy, latency and user experience. In this paper, we will section the paper based on the Front-End of the application (Step 1), the Back-End of the application (Step 2,3,4), and the Building and Testing phase (Step 5, 6 and 7)**

***Keywords*: XR, Sensor, IoT Dashboard, Azure, Integration, Temperature, LoRaWAN, Humidity, headset**

I. INTRODUCTION

1.1 What is XR

Extended Reality is a field that encompass all forms of immersive technology, including Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR). VR is completely immersive, transporting users to a different environment; AR overlays digital information on the real-world environment; and MR is a combination of Virtual and Augmented Reality that enables users to interact with digital objects (Virtual) in a physical environment (Augmented).

XR is a term used to encompass all forms of immersive technology, including Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR). VR is completely immersive, transporting users to a different environment; AR overlays digital information on the real-world environment; and MR blends real and virtual worlds. XR: Extended Reality is an umbrella term that includes Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR). It represents a wide spectrum of hardware and software,

including all types of computer-altered reality, from fully immersive VR to enhanced AR.

MR is a combination of Virtual and Augmented Reality that enables users to interact with digital objects (Virtual) in a physical environment (Augmented).

1.1 Application of XR in Industry

In the industry, MR allows employees and students to gain access to valuable knowledge of how real-world facilities and plants work [4]. Implementing MR eliminates the need to be physically present at a factory. It allows high-risk tasks to be replicated with ease and possibly no injuries, therefore proving the effectiveness of MR applications in the industry. For instance, Sims [5]

introduced Boeing VR/AR technology for aircraft design and manufacturing, where engineers in helmet-mounted displays can explore the aircraft long before any production begins.

1.1 Overview of IoT

IoT: Internet of Things refers to the network of physical devices embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet. IoT devices collect data from the physical world. This data can include temperature, motion, light, sound, and more. The data is then transmitted to a processing unit.

1.2 HMD devices

1.2.1 Meta Quest 3

Meta Quest 3 is a headset developed by Reality Labs, a division of Meta Platforms. It was released on October 10 as a successor to the Quest 2. From the below Figure, the face of the headset is decked with three "pills" containing sensors and cameras; the two outer pills each contain two cameras: a monochrome camera used for positional tracking, and a color camera used for mixed reality passthrough. The center pill contains a depth sensor, which is used in combination with other sensors to sense the user's surroundings for boundaries and mixed reality experiences.

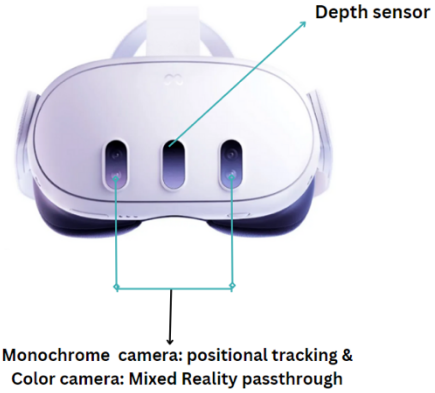


Figure 1: Front view of Meta Quest 3 headset

The headset features updated hardware with elements of the Quest Pro, including a thinner form factor and lenses, and additional

sensors and color passthrough cameras intended for mixed reality software.

1.2.2 Apple Vision Pro

Apple's AR Headset is an exceptional platform for this endeavor due to its cutting-edge AR capabilities, which go far beyond existing systems. Its highly sophisticated lidar sensors, depth perception, and image recognition capabilities make it a leading platform in AR technology. This provides a unique opportunity to explore groundbreaking applications and contribute to the evolution of AR.

1.3 Architecture Diagram

The study will showcase six steps in designing and integrating real time data to the XR campus map application. The architecture of the six steps of the process is shown below Figure 2.

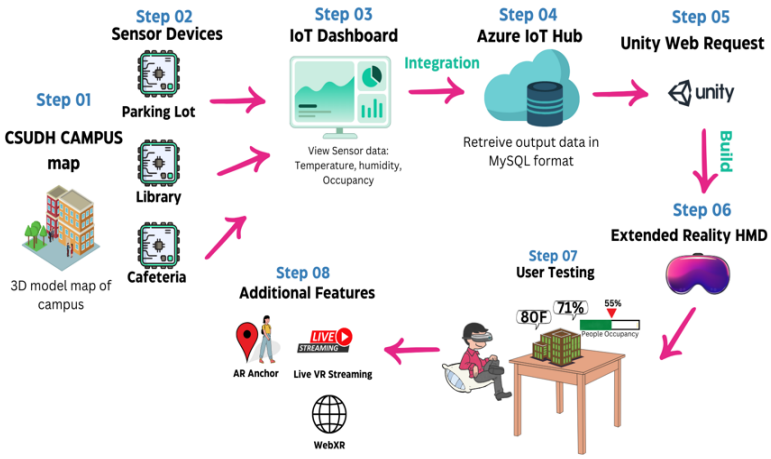


Figure 2: The overall architecture consisting of all the steps of the project.

In this thesis, we will showcase the implementation of the front end, back end and then the build and test of the XR application which includes from Step 1 to 9 of the architecture diagram (refer Figure 2 and Figure 3). The objective is to get the real time data from the campus buildings and present in our XR application. The sensor data we want to measure are temperature, humidity, and people occupancy of the selected campus building. In order to achieve this we need a sensor device and transfer the data to the cloud which can be web requested in the Unity3D application.

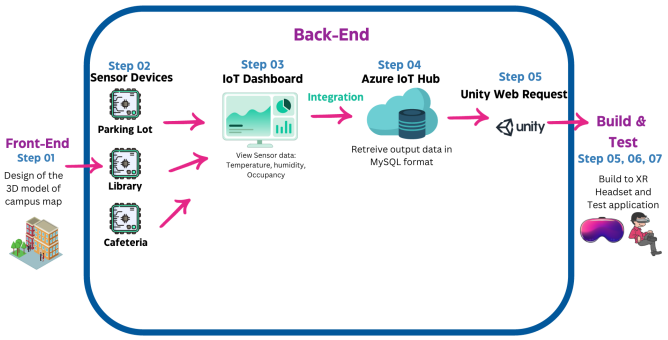


Figure 3: Mobile AR campus map representation

II. DESIGN OF 3D CAMPUS MODEL MAP

2.1 Design of Campus Map

The first step was to build the front end of the application, which is the 3D model of the CSUDH campus map. We want the 3D model to be a similar representation of the building layout, building locations, building architecture, roadways, parking lots, and other key features building. The 3D campus model was designed using Unity3D, which is a real-time 3D software to build games, apps, and experiences.

2.2 Campus Interactive Map

To design a similar campus layout, a detailed and accurate representation we used the CSUDH interactive Campus Map (Figure 2) and ToroGIS (Figure 3) as sources. The campus map was developed by ArcGIS, an online geographic information which provides contextual tools and services for mapping and spatial analysis which is system software developed by Esri. The campus map shows building information, points of interest, and rooms of the campus. It is mainly used for navigation of different rooms and buildings. This information was utilized as a foundational base to develop a XR 3D model of the CSUDH campus. We want to ensure the virtual representation closely mirrors the real-world campus environment.

2.3 3D Modelling

We want to ensure the virtual environment accurately reflected the real-world campus. The detailed campus layouts was retrieved from the CSUDH interactive Campus Map [1] and ToroGIS [2] based on previous Figure 2(a) and 2(b). The task was to use this information as a base to design the dynamic 3D campus model using Unity3D software.

In order to construct the 3D model we have to begin with the creation of the terrain to match the geographic contours of the campus, I used the Unity's Terrain tool for sculpting of the campus grounds, ensuring the elevation and landscape features were represented true to life. Based on the Figure 3 below, can see in the inspector panel, the Terrain Mesh Filter is added. The 3D campus model would be positioned in the middle of the terrain. The box collider is added to the terrain for future user interaction and to prevent the campus building from falling down.

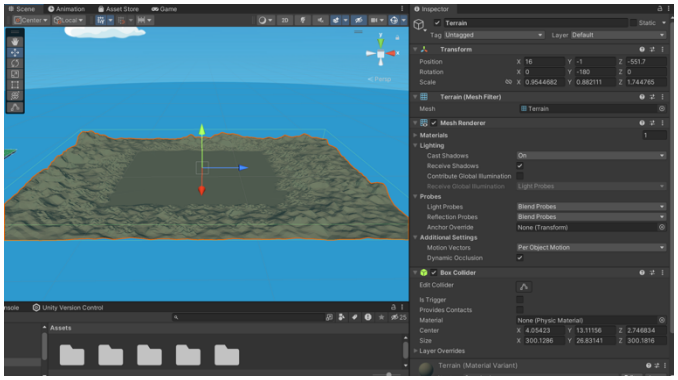
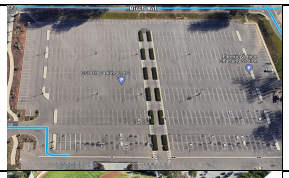
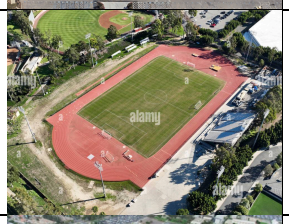
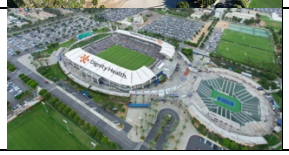
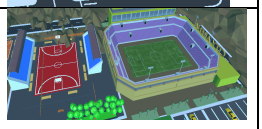


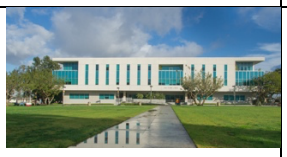
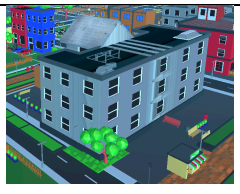
Figure 3: Terrain 3D structure

Parking		
Stadium		
Dignity		

2.4 3D Building Representation

After the groundwork was laid, the focus shifted to the building architectural elements. I had to ensure each building was modeled to reflect its real-life counterpart's dimensions, architectural style, and color scheme. The online building models were retrieved from online 3D open-source websites. The 3D models were fbx format which is a format that Unity uses. The roadways, parking lots, and walking paths were, with attention to their connectivity and layout, ensuring that virtual navigation would be intuitive for users familiar with the campus. Have to ensure the texture of the building surfaces required a balance between visual fidelity and performance, using tiling textures to create realistic surfaces without overburdening the
The Table 1 below shows the 3D model and its counterparts of the real-physical building.

Table 1: Comparison of realistic 3D campus with Unity 3D

Building Name	Realistic 3D model	Unity 3D Model
I&I		
Welch Hall		
LSU		
Library		

Directional light and street light were added to bring the campus to life. Dynamic lighting was configured to simulate different times of day, enhancing the realism of the virtual campus and allowing for the observation of the model under various lighting conditions.

Fig. 4 (a) and (b) represent the comparison of perspective view of 3D campus map with Unity 3D model map

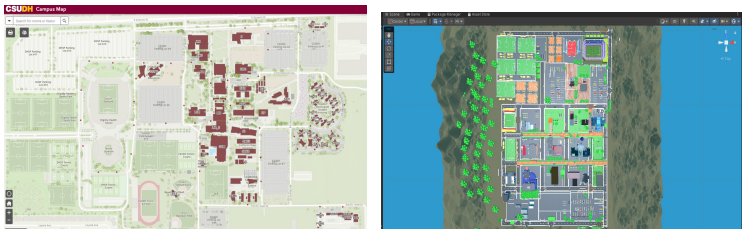
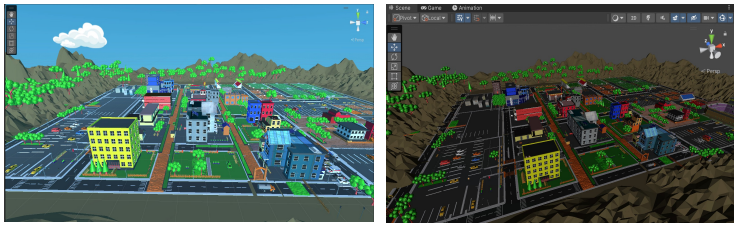
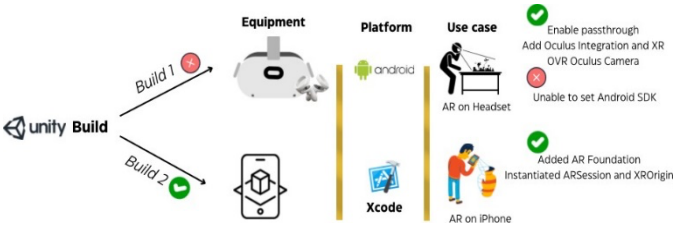


Fig. 5 (a) and (b) represent the comparison of top view of 2D campus map with Unity 3D model map

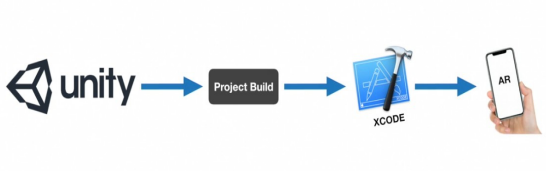


2.5 Test Build of the 3D model

In order to get a general idea of how the application would be in XR. We have to test build the 3D model of the application to the smartphone and the Oculus headset.
The below figure shows the equipment, platform of the two processes to build the application to headset and smartphone



The 3D campus model was initially tested on the iPhone mobile device to ensure compatibility and performance. The Figure 6 below shows the schematic process to build a Unity3D application to iPhone. This requires a Mac machine or a Virtual Mac machine on Windows since the build solution is in XCode.



Unity technologies created a framework known as AR Foundation, which unifies mobile AR development by offering a common solution which supports core functionality for both ARCore and ARKit. Since in this case we are building for the iPhone, therefore we would select the ARKit in out Startup (refer Figure 7 below).

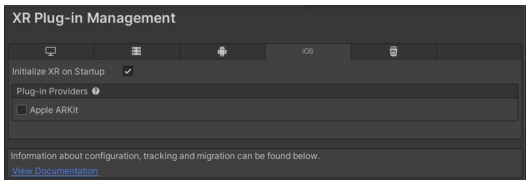


Figure 7: Apple ARKit selected under XR Plug-in

The advantage of using Unity3D is the different platforms you can build the application to. In this case since we are building on an iPhone, therefore we will use the iOS platform (refer Figure 8 below). Before building the project, we need to select the scene we want to build. If multiple scenes are present, for example a multi-level game, we would have to select multiple scenes. In this case, we are using only a single scene. If we want to build it for the Oculus headset, we will use the Android platform since Oculus is based on Android source code.

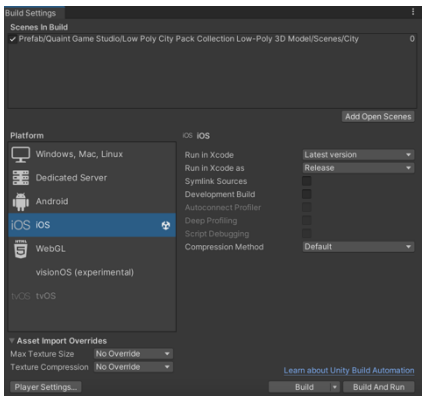
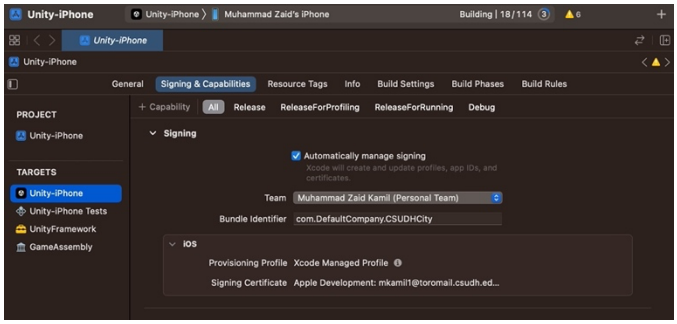


Figure 8: Unity build platforms on scene

Once the scene is built in Unity3d, the build solution is an XCode file. Since the iPhone can only read XCode file. To transfer the build application to the smartphone, we need to connect a USB lightning cable from the phone to the Mac machine to transfer the data to the phone. Once connected the XCode will show as below Figure 9. Before building, have to make an Apple developer account and check the signing capabilities. The Apple developer account allows three application build on the iPhone.



Once the application was built and transferred to the iPhone. In the iPhone, the application is a Unity application. When starting the application, it will ask to enable camera. Once the camera permission is enabled. The application will be super-imposed into the real world surrounding. Since there are no UI features added, therefore the AR would be just the campus map.

Figure 9: Mobile AR campus map representation

III. SENSOR DEVICES

3.0 Selection of the sensor devices

When selecting the sensor devices we need to take into consideration the portability (battery powered or not), accuracy of data, data acquisition rate, integration of data. The below Table 2 gives a comparison of the different IoT sensor devices. The table highlights the differences between the two types of IoT sensor devices when utilized for sensor data collection in the CSUDH campus environment.

Table 2: Comparison of Raspberry Pi and LoRaWAN Sensor devices

Criteria	Raspberry Pi-based Sensors	LoRaWAN Sensor Devices
Connectivity	Wi-Fi, Ethernet, Bluetooth (with additional modules)	LoRaWAN (Long-Range Low-Power Wireless)
Power Consumption	Higher, requires a stable power supply or battery management	Low, optimized for battery operation
Range	Limited by Wi-Fi/Bluetooth range (tens of meters)	Long-range (up to 15 km in rural areas)
Data Transmission Frequency	High-frequency updates possible	Lower-frequency updates (data acquisition rate ~ 15 mins)
Cost of Components	Lower cost for the base unit, but additional sensors/modules needed	Higher cost with the gateway, but integrated with necessary sensors
Setup Complexity	Moderate to High (requires OS setup, programming skills)	Low to Moderate (plug-and-play sensors, pre-configured gateways)
Network Infrastructure	Requires Wi-Fi/Internet infrastructure	Requires a LoRaWAN gateway, but fewer needed due to range
Scalability	Scalable with some complexity in network management	Highly scalable with simple network management
Data Security	Depends on the security measures implemented	Generally high with built-in encryption and secure protocols
Community Support	Extensive community and resources available	Growing community, especially in IoT sectors

Raspberry Pi-based sensors offer high customizability and can support high-frequency data updates, making them suitable for complex projects that may require additional computing power or peripheral integration. LoRaWAN sensor devices, on the other hand, are optimized for long-range, low-power operation, making them ideal for deployments where battery life and coverage area are critical. Based on the above Table 2 and research, the sensors we selected the LoRaWAN Sensor Devices.

Initially the Raspberry pi sensor was taken into consideration, however, Power consumption: Raspberry Pi consumes more power than Arduino, making it unsuitable for battery-powered applications.
Complexity: Raspberry Pi is more complex than Arduino and requires some programming skills to use effectively

3.1 LoRaWAN Sensor Device

LoRaWAN (Long Range Wide Area Network) network device, often used for connecting sensors in a wide-range IoT (Internet of Things) setup.

We used the Cisco Industrial sensors to provide telemetry data such as temperature and humidity. The Cisco AV201: Indoor Temperature/Humidity Sensor were used. The LoRaWAN network gateway was used for long-range connectivity with the sensors.

The data collected by these sensors was transmitted via a LoRaWAN network gateway. This technology was chosen for its long-range connectivity (up to 10km) and low power consumption (5-year battery life), which is ideal for the extensive campus setting such as CSUDH. The gateway's robust coverage allowed for the seamless collection of telemetry data from even the most remote sensors on campus, ensuring no area was left unmonitored.

The AV201 is a LoRaWAN sensor is a wireless communication device which detects ambient air temperature and humidity and transmits the data to the gateway through the LoRaWAN wireless network. It is battery powered which is intended for use for Indoor Temperature and Humidity Monitoring. Mounting options are available, including using its magnetic base to attach to a ferrous objects.

The Figure 11 below shows the components of the AV201 Sensor device.
Function button: To turn on the sensor, we have to press and hold the function button for 3-5 seconds.
LED indicator: if the LED illuminates solid for few seconds, Sensor has been turned on. If the LED blinks 20 times:
Sensor has been turned off.
QR Code: It is used for onboarding the sensor device to the Cisco IoT Dashboard.

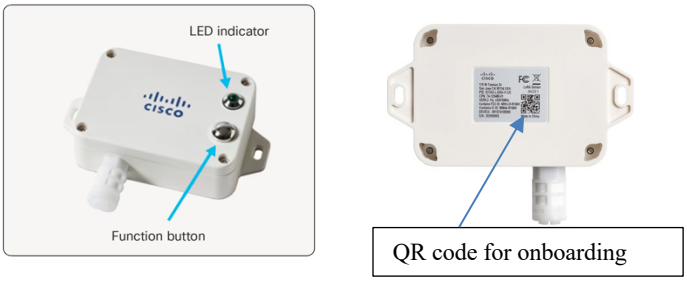


Figure 11: AV201 Sensor devices to measure indoor temperature and humidity.

The Figure 12 below shows the components of the LoRaWAN network gateway device:

LED: Usually serves as a status indicator that provides visual feedback about the device's operation. Green LED indicate power on. Blinking green LED may represent network activity, or error states.

RF Antenna Connectors: These are used to connect the Radio Frequency antennas that enable wireless communication over long distances. The device has two connectors, allowing for diversity in reception and transmission, which can enhance the signal quality and reliability.

There are also mounting option to Mount the gateway: the gateway is able to receive sensor data.

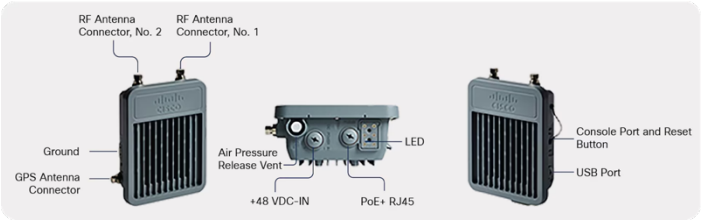


Figure 12: LoRaWAN Network Device including all the components.

We use the Mobile application to Onboard sensors and the network gateway device by scanning the QR code in the back of the device. The network gateway is able to withstand harsh enviroment and can cover several kms. The Figure below shows the simple architecture of the connectivity between LoRaWAN, sensor and IoT Dashboard.

Figure 13: Connection of sensor

Eventually the IoT sensors in campus buildings (Campus Library, Parking Lot, and Cafeteria) to measure the environmental conditions such as temperature and humidity inside the campus buildings and measure the occupancy data of each campus building using the campus Wi-Fi network.

IV. IoT DASHBOARD

Once the sensors were onboarded, the telemetry data (Humidity and temperature) were displaying in the Cisco IoT Dashboard, which is a cloud-based application to view the sensor data, location, battery life and online status. By using the dashboard, we are able to monitor all the IoT Devices & Backend Services in One Place w/ ASingle Lightweight Agent.

4.0 IoT dashboard navigation

The Figure 14 below shows the landing page of the IoT dashboard which shows the map-based monitoring dashboard and the status of the sensor devices. The IoT Operations Dashboard provides a central monitoring interface for managing and observing sensor status. This intuitive dashboard offers a comprehensive view of the sensors deployed across various environments.

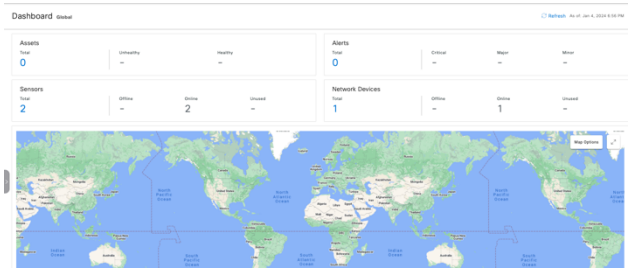


Figure 14: Dashboard Landing Page which shows the location of the sensor and network devices

The Figure 15 below shows the sensors that are registered within the system. It also displays the total number of active alerts, which means if the temperature goes below a specific certain value, it will send an alert. The 'Asset Type' is 'Indoor Environment', which implies it's used for monitoring conditions within buildings, such as temperature and humidity levels. The 'Control Status' is 'Healthy', indicating the sensor is operational and no issues have been detected.

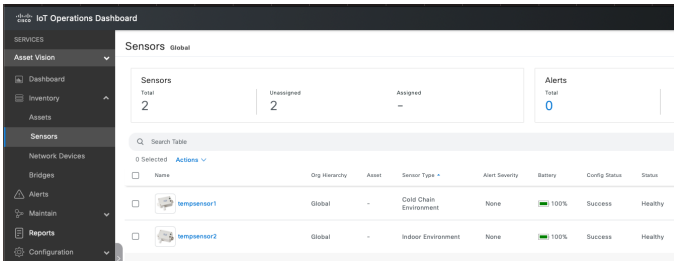


Figure 15: Dashboard Landing Page which shows the location of the sensor and network devices

Figure 16 below shows the registered LoRaWAN network in the system. Cisco's LoRaWAN-compliant solution connects IoT sensors and endpoints across both cities and rural areas, at low cost. Low power consumption extends battery life up to several years without being replaced.

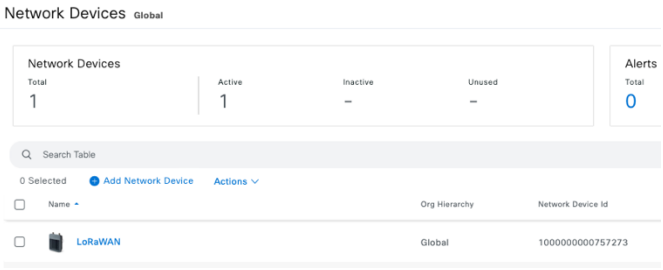


Figure 16: Overview of Network Devices registered in the Dashboard system.

The Figure 17 below shows the data monitoring interface of one of the sensors, shows the recorded temperature and humidity data over time. The top chart displays the humidity levels recorded by the sensor. The bottom chart displays the temperature data over time with a data acquisition rate of 15 mins.

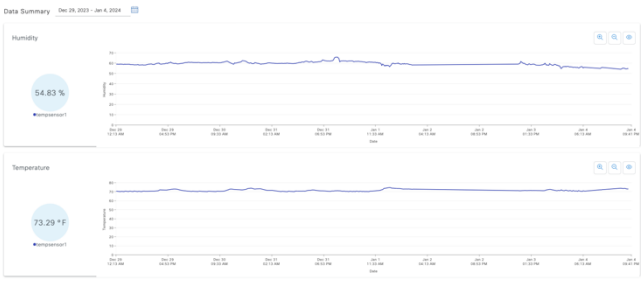


Figure 17: Humidity and Temperature data monitoring

4.1 Sensor connection interface

The below Figure 18 shows the overview of the connection of the Interface with the Sensor devices and the Dashboard. The type of data we are going to collect are the temperature, humidity and occupancy inside buildings. The sensor devices we are going to use are the Cisco AV201 for Temperature and Humidity data monitoring, Vision AI Sensor for building occupancy monitoring, and Milesight for classroom occupancy monitoring.

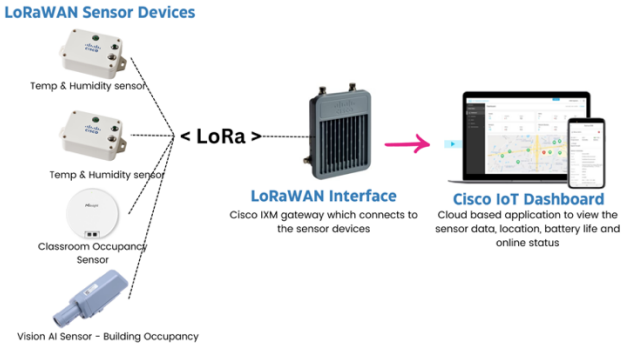


Figure 18: Humidity and Temperature data monitoring

4.3 Integration of sensor data

After successfully accessing the real-time temperature and humidity data on the IoT Dashboard. The next step is to integrate this data to the Azure Cloud. There are two Integration types: one is from the MQTT Server or directly through Microsoft Azure IoT. MQTT, a lightweight messaging protocol for small sensors and mobile devices, is well-suited for the Internet of Things and can be used to forward data to various endpoints. However, for streamlined operations and advanced cloud capabilities, we chose the Microsoft Azure IoT integration. With Azure, we can utilize tools such as Azure Stream Analytics to gain real-time insights and Azure Machine Learning for predictive analytics.

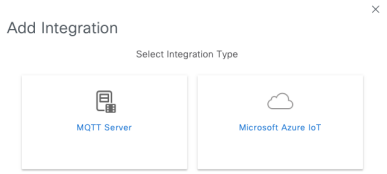


Figure 19: Integration Type of MQTT Server and Azure IoT

The figure below shows a configuration panel for integrating sensor data with Azure IoT, detailing the setup process for establishing a connection between an IoT device and Azure's cloud services. We have to make a device within the Azure IoT Hub to get a unique identifier.We used the shared access key type of credential used for authentication with Azure IoT Hub. We have to retrieve the Shared Access Signature key from the device with access to Azure IoT Hub. The certificate file used for establishing a secure connection ensures that the communication between the device and Azure IoT Hub is encrypted and authenticated.

• For creating a new Integration, the user should be aware of the Connection Details of the destination server, such as Host IP/name, Port number, user credentials, and Topic Pattern in case of an MQTT server.

4.3 Certificate file generation

This session is part of setting up a secure connection between a local environment and Azure IoT, which requires a trusted SSL/TLS certificate. The certificate details are crucial for establishing a secure, encrypted channel to ensure that data sent to and from Azure IoT is protected from interception or tampering as it ensures the identity of the server and the privacy of communicated data.

To secure the client connection to the Azure portal server we have to use the terminal session on Ubuntu, specifically within a command-line interface (CLI). the openssl command to initiate a TLS handshake with Azure's portal: We use the shell script:

openssl s_client -showcerts -connect portal.azure.com:443

The openssl s_client utility is a diagnostic tool for establishing a client connection to a remote secure server.

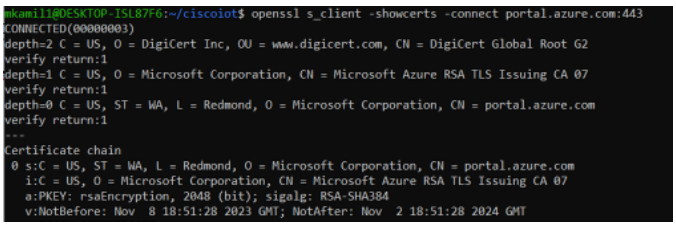


Figure 20: Command Line for Azure integration

The root CA certificate is in PEM format is shown, starting with -----BEGIN CERTIFICATE----- followed by the base64 encoded certificate data which ends with -----END CERTIFICATE----- lines. We have to save it as a new file with the .pem extension

Figure 21: Certificate generation

V. AZURE IOT HUB

We need to make an Azure account and subscription. We use the campus email address to create one. We first create the Azure IoT Hub. We use the subscription Azure for Students. We create a device under the Azure IoT hub. From the device we select the Symmetric Key authentication type when creating the device. This would generate a primary key.

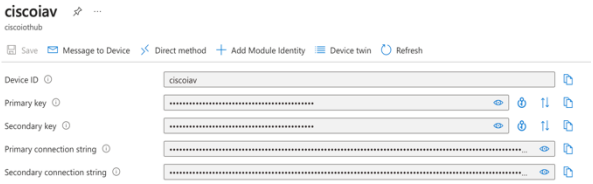


Figure 22: Device creation

5.1 IoT Sensor data process pipeline

The Figure below is a IoT data pipeline where raw data from sensors is ingested into a cloud platform, processed and analyzed, and then stored and presented in formats that are useful for end-users or other applications.

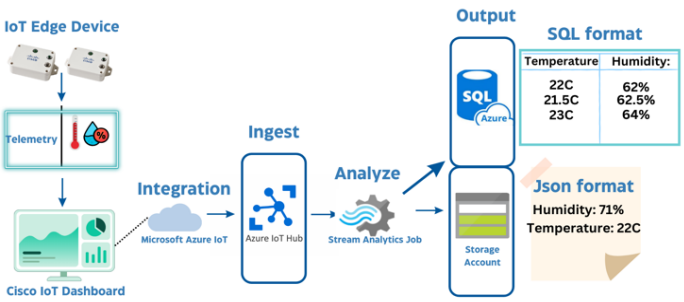


Figure 23: IoT data pipeline

The pipeline starts with telemetry data from IoT Edge devices, the Cisco IoT sensors collect the telemetry data. This data includes metrics like temperature, humidity. : The collected telemetry data is initially monitored and possibly managed through the Cisco IoT Dashboard, which provides a user interface to observe and control the IoT devices. The telemetry data from the Cisco IoT Dashboard is integrated with Microsoft Azure IoT services.

The integration is likely facilitated through Azure IoT Hub, which is a managed service that acts as a central message hub for bi-directional communication between IoT applications and the devices it manages. The Azure IoT Hub ingests the telemetry data from the IoT devices. The data is then ready to be processed and analyzed by Azure services. The Azure Stream Analytics Job is used to process the incoming telemetry data in real-time.

This service can filter, sort, aggregate, and analyze the data, turning it into meaningful insights or actionable information. After analysis, the processed data can be stored in an Azure Storage Account, which might be in the form of Azure Blob Storage.

This storage can be used for long-term data retention, further analysis, or as a data source for other applications or services. The results from the Stream Analytics Job are then outputted in two formats: SQL Format: Data is organized into a structured format with columns like Temperature and Humidity, making it suitable for storing in an Azure SQL Database. This structured format allows for easy querying and reporting and JSON Format: Data is also transformed into a JSON format, which is a flexible and lightweight format for storing and transporting data. This is useful for APIs, web services, or applications that consume JSON.

5.2 Azure Resource Utilization

The below Figure 24 shows a list of resources used within a Microsoft Azure environment. Each resource is listed with its name, type, and the last time it was viewed. Which includes the SQL Database to store the data, SQL Server which is based on the database, Stream Analytics for the Database job.

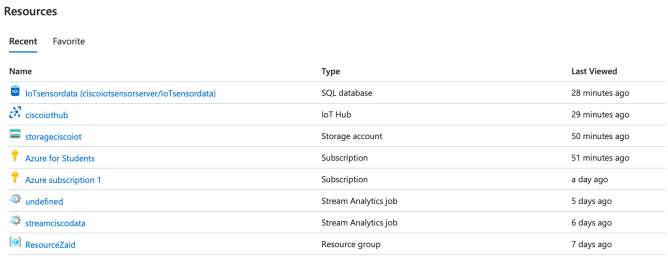


Figure 24: Resources used withing Microsoft Azure

From the below Figure we the following resource type utilized from the Azure portal and their function in the context of retrieving sensor/telemetry data:

We use the Azure IoT Hub which is hosted in the cloud, that acts as a central message hub for bi-directional communication between the Cisco IoT sensors and the devices. This enables to gather telemetry data from your devices, manage them, and integrate the data with other services like Stream Analytics.

The (Stream Analytics job): provides a real-time analytics to analyze and process the telemetry data as it arrives from the IoT hub. In this case there are two streaming analytics for each jobwhich is stored in Blob Containers or SQL Database.

The Azure Storage account contains all of your Azure Storage data objects, including blobs, file shares, queues

, tables. In this case, the storage account is used to store large amounts of unstructured data, such as the full payloads of telemetry data collected from sensors before or after it has been processed by Stream Analytics.
The SQL database is used to store, query, and manage data. When paired with Streaming Analytics, the processed telemetry/sensor data is stored in a structured table format, allowing for easy querying and reporting of temperature and humidity. The subscription under which all the Azure resources are billed. Multiple subscriptions can be used to manage costs and organize resources across different teams or projects. It enables students to access and learn Azure services without incurring high costs. A resource group is a container that holds related resources for an Azure solution. In this case, 'ResourceZaid' is the name given to the group that contains all the resources related to a specific IoT solution or set of solutions.

Each of the following resources plays a role in the process of collecting, processing, storing, and managing telemetry data from IoT devices.

5.3 Azure Cost Analysis

Based on the resource utilization within the Azure Portal we can get a breakdown of the costs associated with each Azure resource over a specified time period. The cost analysis indicate in terms of resource utilization based on sensor data integrated with sensor devices. The individual resources that incur the highest costs include Stream Analytics Job and SQL Database.
The stream analytics job processes large streams of real-time data. In the context of IoT, this could be processing telemetry data from sensors. The cost is likely based on the number of streaming units (SUs) used and the time the job has been running. The SQL Database stores processed data for querying and analysis. The cost would depend on the database's number of vCore, size, and how long it's been running. DTUs are more cost friendly in SQL database. The cost for each resource will vary depending on usage patterns and the specific Azure pricing model chosen for each service (e.g., pay-as-you-go, reserved instances). The cost breakdown helps you understand and manage these expenses as part of your overall Azure IoT solution.

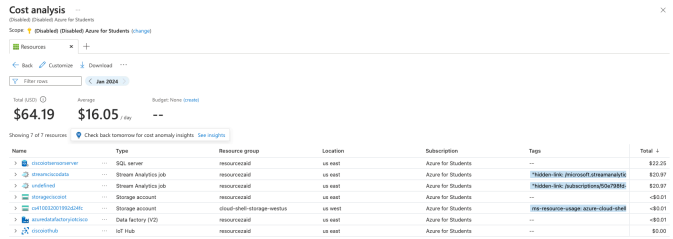


Figure 25: Individual resource cost analysis

We can also monitor the costs associated with the collection of sensor data which gives visual representation of spending trends over time, we can do a cost over time (line graph) which indicates the day-by-day expenditure. The cost by resource which breaks down costs by individual Azure resources. It shows that the "Stream Analytics Job" is the most significant cost driver, followed by "SQL Database", and negligible costs for "IoT Hub" and "Functions".

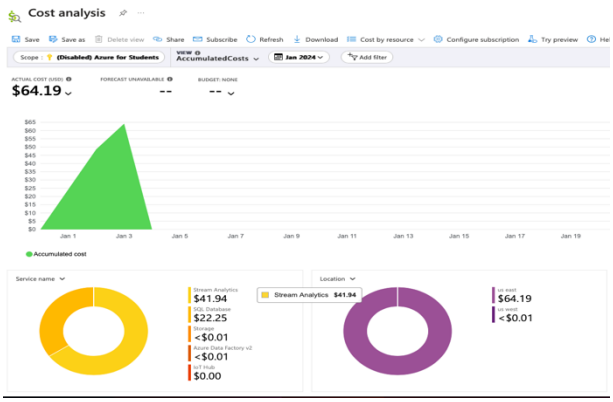


Figure 26: Visual resource cost analysis

5.4 Streaming Job process

The below Figure explains the data processing workflow that includes querying the input data from the Azure IoT Hub then storing it in Blob Storage and an SQL Database. The results are outputted into Blob Storage in the JSON and SQL Database. The JSON structure includes temperature and humidity objects, each with value and unit properties. The second output from second streaming analytic job is shown in a table in a more readable SQL table format with two columns, Temperature and Humidity.

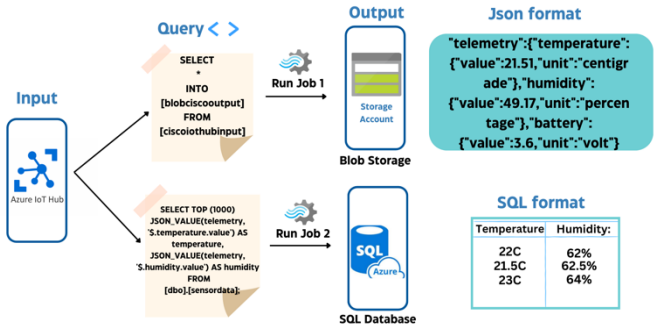


Figure 27: Overview of the Job Diagram

5.5 SQL Database

In order to view the data in SQL format, we created the SQL Database resource. Based on the query we can view the telemetry data. The data received in the cloud is in JSON format. Therefore, we only need to retrieve the temperature and humidity value (refer Figure below). Under the Query editor in the SQL Database, we run the following SQL command.

SELECT TOP (1000)
JSON_VALUE(telemetry, '$.temperature.value') AS temperature,
JSON_VALUE(telemetry, '$.humidity.value') AS humidity
FROM [dbo].[sensordata];

The above query selects the top 1000 entries, extracting temperature and humidity values from a JSON structure in the telemetry column of the [dbo].[sensordata] table. The values are being retrieved using the JSON_VALUE function, which extracts data from a JSON document. The actual data is stored in a single column in JSON format, and the query is used to transform that data into a tabular format for easier analysis and reading (refer Figure). The query successfully extracts the desired values from the JSON and presents them in a structured table as shown in below Figure.

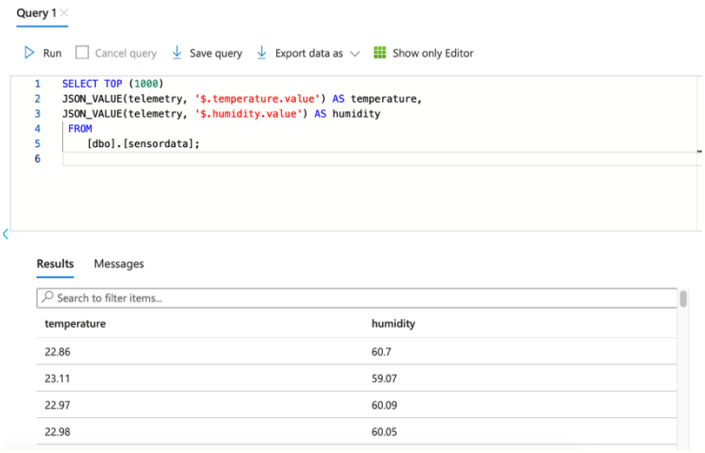


Figure 28: Temperature and Humidity in Table format

5.6 Blob Storage Container

The Figure 29 below shows a screenshot from the Azure Stream Analytics Query editor. The editor is used to write, test, and run Stream Analytics Query Language queries.

*SELECT * INTO [blobciscooutput] FROM [ciscoiothubinput]*

The above query is inserting all data from a source (ciscoiothubinput, possibly the Azure IoT Hub) into a destination (blobciscooutput, likely a container or folder in Azure Blob Storage).

The result of the Query gives the following:

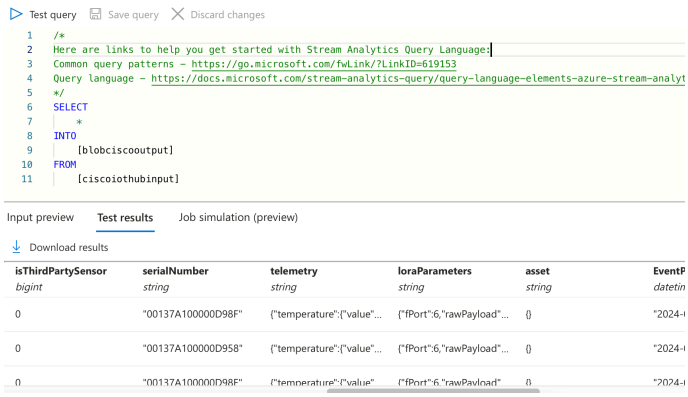


Figure 29: Result Azure Query in Blob Storage

The Figure below displays the interface of an Azure Blob Storage container, and how the data is organized within an Azure Blob Storage account, specifically showing the sensor data sorted by date for efficient access and management. This allows for the end-user to use this structure to find and access JSON files containing sensor data from specific days. We can access the Json file. Which consists of temperature and humidity on each day.

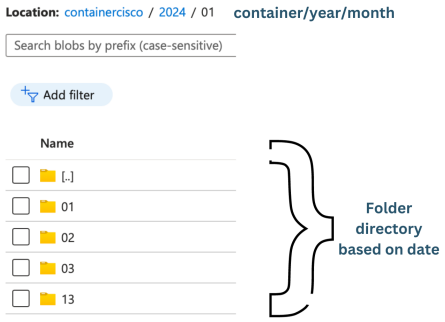


Figure 30: Container blob storage