Real-Time NASA Asteroid & Weather Monitoring Using Azure IoT and Streamlit

Brindavan Samuthira Pandian Ganesan

X23430320@student.ncirl.ie

Repository Link

Video Link

# Abstract

This is a real-time asteroid and weather monitoring dashboard using the Azure IoT and Streamlit. It consumes telemetry data provided by a fake Internet of Things device that is an asteroid tracking system and environmental sensors proposed by NASA. The system also visualizes major indicators including proximity of the asteroids, their velocity, temperature and humidity and issues warnings of potential dangers. This project is inspired by satellite-aided research and space-based IoT networks and investigates the usability of edge to cloud communications as critical space situational awareness R&D.

# Introduction

The Internet of Things (IoT) has changed the world of data capture and tracking in industries, and its merger with space application is becoming one of the key frontiers. Space-based IoT allows instant contact with the satellites, ground sensors, and mission control to build a network of connected devices to be utilized in case of critical instances. This project describes how to develop a real-time monitoring dashboard, provided using Azure IoT to monitor near-Earth objects (NEOs) and environmental telemetry. The use of weather data gives a synergetic aspect to monitoring the environment and space objects as well as discussing the opportunities and challenges of space habitats and satellite aided computing of the Internet of Things in space. These issues of the low delay data provision, efficient communication, and warning provision are addressed by the project through the mixed requirements of real-time telemetry ingest and visualization. They are important factors to consider in planetary defense as well as space living environments with enhancements of their digitally.

# System Architecture

The proposed NASA Asteroid and Weather Monitoring System complies with the cloud-integrated architecture of the Internet of Things (IoT), which incorporates real-time ingestion, processing, and visualization of the data. General architecture consists of four main layers, namely, data acquisition, cloud integration, stream processing, and visualization.

1. **Data acquisition layer**

Telemetry is acquired by the layer in the form of asteroid and weather telemetry. The asteroid information is pulled through NASA Near Earth Object (NEO) application programming interface (API) and it contains information about the nearest passing asteroids, their distances, velocities, and level of riskiness. At the same time, environmental sensors based on IoT technology record weather telemetry, i.e., monitoring temperature and humidity readings in real-time. The raw data is processed by a Python-based telemetry module (telemetry.py) into JSON objects and a timestamp is appended to each object to synchronize them.

1. **Cloud Integration Layer**

The telemetry data is relayed with integrity to the Azure IoT Hub after going through preprocessing stages as the central binary for information ingestion between the device and cloud. The IoT Hub provides high-assurance message-delivery, scalability and device identity solutions. Messages are sent to the Azure Event Hub via the IoT Hub acting as the backbone of the high-throughput data streaming so that it can consider the telemetry by several consumers at once.

1. **Stream Processing Layer**

The Event Hub Listener (iot\_listener.py) subscribes to the dataset in the Event Hub asynchronously and reads the consumption of the telemetry data in near real-time. It decodes single-object messages (weather info) and batch messages (asteroid data) and puts them in a shared in-memory telemetry queue. The ingestion of data is decoupled to dashboard in this design; it makes latency short, and it does not lose any data during high-frequency streaming.

1. **Visualization Layer**

The user interface (dashboard.py) is an implementation of Streamlit that consumes a queue of telemetry and updates in real-time. Asteroid telemetry is presented graphically by KPIs, the velocities between asteroids and the earth, distance meter and a table ranking the top five nearest asteroids. The data of the weather telemetry is realized through temperature and humidity charts, real-time key performance indicators, and alerts. Dark-themed interface guarantees easier readability, when the data is regularly monitored.

There are three main components of the architecture:

* Azure IoT Device 01 - Simulated device receiving telemetry about asteroids and weather.
* Azure IoT Hub - this is just a messaging broker of the device and processing layer.
* Streamlit Asteroid and Weather Dashboard - Visualization of asteroid data and weather data in real-time.

The responsible process routes Telemetry messages sent by the IoT device to the Azure IoT Hub and consumed by the dashboard and stored there in the form of buffering, processing and visualization. The dashboard has a capability to monitor both the proximity of the asteroids and the environmental conditions and raise alerts once the set limits are reached.

Its architecture enables modular implementation of other data sources and follows the best practices of IoT-based structure in terms of fault resilience and scalability. The system uses the managed services of Azure to make the delivery and transfer of the information low latency with reliable real-time monitoring of both near-Earth objects and environmental conditions.

# Implementation

NASA Asteroid and Weather Monitoring System was implemented with the help of a mixture of Python-based IoT modules, the Azure IoT hub, and a real-time visualization dashboard developed with Streamlit.

1. **Telemetry module**

There are two important functions carried out by the telemetry module: (a) it accesses asteroid data retrieved over the NASA Near-Earth Object API and (b) it gathers environmental conditions telemetry via local weather sensors. The response obtained with the help of NASA API is parsed to retrieve the five nearest asteroids in relation to its missing distance and the pertinent parameters, the distance, velocity, and alert status are arranged as JSON objects. In the case of weather data, the climate records of temperature and humidity are shipped with timestamps to provide synchronization. The message in the required form accordingly is sent to the Azure IoT Hub through the IoTHubDeviceClient of the Azure IoT Python SDK.

1. **Cloud Integration via Azure IoT Hub**

IoT Hub is the safe entry point of the device to cloud communication. It validates the device through connection strings, manages messages and pipes the telemetry to an Azure Event Hub. Event Hub is used as the high throughput streaming backbone to stream real time data to be ingested by the various consumer applications.

1. **Event Listener Module**

An asynchronous listener is a special communication model that uses the AMQP-over-WebSocket transport to subscribe to the Event Hub by creating an instance of the EventHubConsumerClient class. Parsed telemetry is pushed into a shared in-memory queue (telemetry\_queue) with timestamps attached and the downstream of the post-processing of the data. The listener is to accept individual weather telemetry packets as well as lists of asteroids telemetry.

1. **Real-Time Dashboard**

The dashboard is also developed with Streamlit with plotted visualizations. It reads the telemetry queue every few seconds and displays the user interface being refreshed. The asteroid information is presented in a graphical form, utilizing KPIs on the most up-to-date object, real-time velocity pattern line graph, gauge of the closest approach distance and a tabular list of top five asteroids. Temperature Weather data is presented with KPIs in a temperature graph, humidity graph and an alert arrest KPI. There is also a history graph of the last 50 readings.

1. **Logging and Fault Tolerance**

Telemetry is all logged into a CSV file so that it may be analyzed offline. The device client contains a reconnection system to cope with interruptions of the network and guarantee the transfer of data. The data buffers are stored in the dashboard, which ensures that they do not interfere with the UI at any time.

# Evaluation

The system evaluation is based on three main major metrics: latency, data accuracy, and system scalability.

1. **Latency Analysis**

With the regular operation, the end-to-end latency, which is defined as the data acquisition time on the device to time of visualization on the dashboard, averaged 2.3 seconds. Latency was minimal when it came to delivery of the messages, because of utilization of Azure IoT Hub and Event Hub, as such, the system is applicable in near real-time monitor systems.

1. **Data Accuracy**

NASA NEO API furnished detailed information about asteroids, and the weather telemetry was checked against the reference sensor data. The numerical precision of the distances and the velocities was preserved in the JSON based message formatting. Absence of error was confirmed against the entries in the CSV that were logged against the API responses achieving more than 99 percent compatibilities.

1. **Scalability and Throughput**

The architecture endured stress tests of simulated high-frequency telemetry (running up to 500 messages a minute). The buffered in-memory queue and the scale of Azure Event Hub provided the support of the load without packet losses. With modular architecture placeholders, other IoT devices or APIs can be added without much architectural intervention.

1. **Visualization**

Streamlit dashboard was able to keep an update interval of less than 3 seconds notwithstanding 500 recent records already in the buffer. Streamlit gauge and charts were smoothly rendered with simultaneous updates, which makes them practical to continuous monitoring cases.

The project was carried out in Python with Streamlit as a front-end rendering tool and Azure IoT as a tool used to connect tenant devices to the cloud. A lightweight device simulated by railing published telemetry such as asteroid distance (km), velocity (km/h), object name, temperature (degree Celsius), humidity (percent), and the level of alerts. Dynamic gauges and visualization of trends were done in Streamlit.

Important points of implementation:

* Multi-threaded listener that could process incoming telemetry in Azure IoT.
* Asteroid and weather KPI real-time dashboard.
* Alert system issuing warning level of asteroids less than 3 million km and critical level less than 1 million km.
* Plots of historical trends of the velocity, temperature and humidity of an asteroid.

They tested the dashboard responsiveness and the way accuracy of real-time information is represented. This system had a low-latency pipeline, and messages did not take more than a few seconds to be processed and showed in the graph. Scalability testing also proved that the dashboard could hold 500 recent telemetry messages within a unifying performance.

# Conclusion

This project shows a complete real-time monitoring system based on the Internet of Things (IoT) which combines space telemetry (asteroid information) and ground weather telemetry with Azure IoT Hub and Streamlit visualization dashboard. Aggregated with NASA Near-Earth Object API and weather sensor data, the system provides a low latency and low failure point (not amounts to high reliability) view of both the sky and the atmosphere, in real time, on a unified view. This architecture and implications are quite coincident with the overall vision outlined in the paper, Internet of Things in Space: A Review of Opportunities and Challenges, yet it focuses on the combination of satellite-aided computing and digitally augmented space living. Project gives real-life demonstrations of how IoT space systems may interconnect with cloud infrastructures to provide insights needed to support missions, with this type of integration underscoring the expanding synergy between the terrestrial IoT and space-based data platforms. The next steps that are planned include the incorporation of machine learning-based predictive analytics to predict the asteroid trajectories and weather anomalies and the expansion of the system to work with satellite-based IoT infrastructures, the addition of which will help bridge the gap between ground-based sensors and future space-enabled IoT systems.

# Screenshots

A screenshot of a computer

AI-generated content may be incorrect.

# A screenshot of a computer AI-generated content may be incorrect.

# *Max paper length is 7 pages in this format (without including references).*

# *Do not exceed the limit and/or modify the font size. Do not add header pages.*

# References

**References should be formatted using APA or Harvard style as detailed in NCI Library Referencing Guide available at** [**https://libguides.ncirl.ie/referencing**](https://libguides.ncirl.ie/referencing)

Beloglazov, A. and Buyya, R. (2015). Openstack neat: a framework for dynamic and energy-eﬃcient consolidation of virtual machines in openstack clouds, *Concurrency and Computation: Practice and Experience* 27(5): 1310–1333.