Version 1.0, December 16, 2024

Integrating Datacake with OpenAI

Executive Summary

Hitechdb LLC designs and deploys IoT solutions. In the summer of 2024, Hitechdb implemented a proof-of-concept project to integrate a Datacake dashboard with a Large Language Model (LLM). This paper provides an overview of that project and consists of the following sections:

- Introduction to the Project
- Overview of the dashboard used to visualize the sensor and AI information
- Overview of the system components
- Project Learnings
- Potential Next Steps
- Conclusion

The result of the project can be seen in the following Datacake dashboard: Demo Dashboard with AI Integration.

Table of Contents

Contents

Project Introduction

The IoT dashboard, Demo Dashboard with AI Integration, is the output of a proof of concept project to integrate the Datacake, low-code IoT platform with the OpenAI developer platform. Project deliverables were as follows:

1. Integration of an IoT sensor with the Datacake platform
2. Construction of a Datacake dashboard to visualize information from the sensor and the Large Language Model (LLM)
3. Construction of an LLM query using sensor information in the Datacake platform
4. Sending the query to an OpenAI LLM
5. Receiving the query result and AI usage information from the LLM
6. Displaying the LLM query results and usage information in the Datacake dashboard

For the last 4 years Hitechdb has been constructing IoT dashboards to visualize data from various IoT sensors, primarily for building environmental monitoring and smart farming use cases. Many of these dashboards use Datacake's rules engine to trigger actions when specific conditions are met, for example sending an email or text message when the presence of water is detected or when a temperature is above or below a certain value or if air quality in a room exceeds a certain CO2 or VOC level.

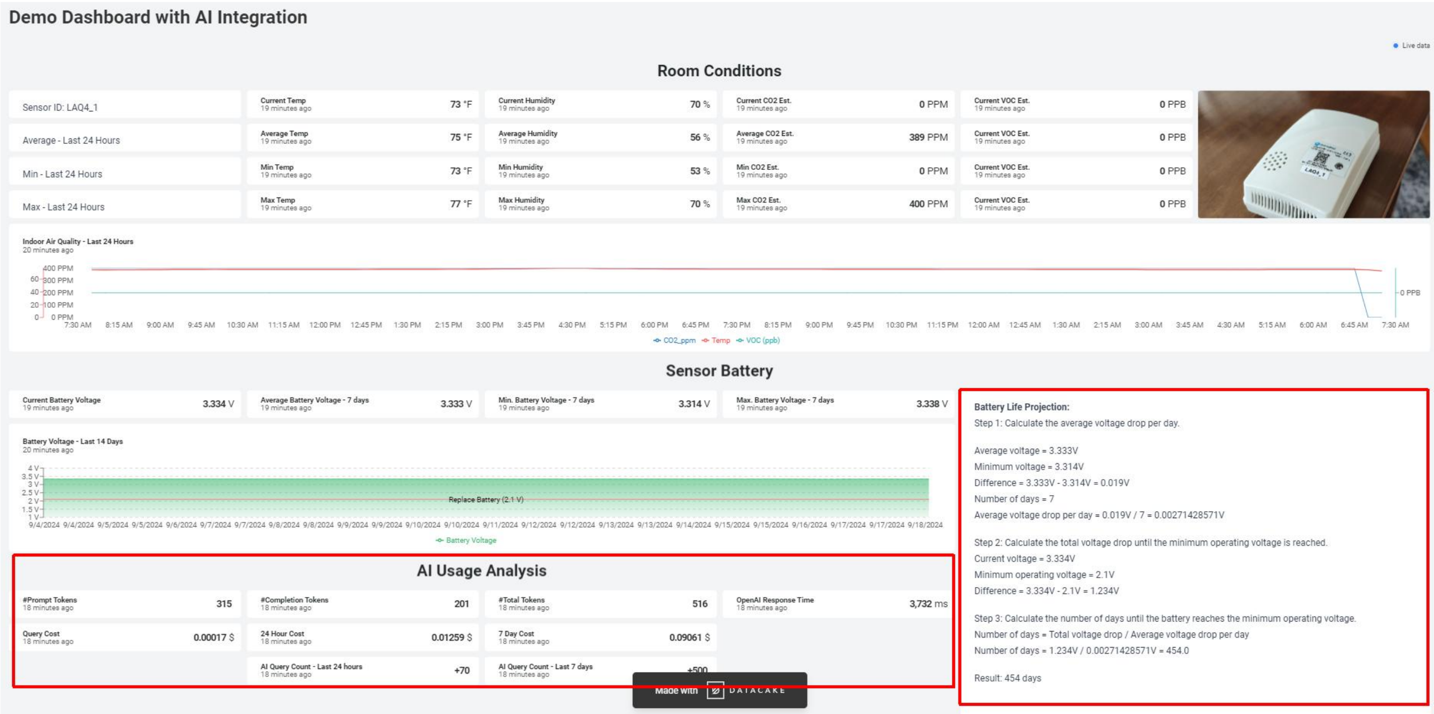
With the recent growth of Generative AI and Large Language models, there is a potential to provide additional useful knowledge and business value with the data in IoT platforms.

The objective of this project was to learn the plumbing aspects of integrating an IoT platform with an LLM. In other words, the project goal was to learn how to get data from the IoT platform into a LLM and the results back to the IoT platform.

The objective of this project was **not** to come up with a business use case for incorporating LLM capability into an IoT solution. That is a project for a different day. One reason for bringing up this point is because by the time I reached the end of the project, I could have implemented the battery life projection functionality performed by the LLM and displayed in the Demo Dashboard with AI Integration without using an LLM all.

Dashboard Overview

The screenshot below is taken from Demo Dashboard with AI Integration viewed in a desktop browser. Viewed in a mobile browser, the dashboard will look a little bit different, but all the same informational elements will be present.



Screenshot – Demo Dashboard with AI Integration

The information in the 2 sections of the dashboard outlined in red, **AI Usage Analysis** and **Battery Life Projection**, comes from an OpenAI LLM. The information in the other sections of the dashboard comes from the Indoor Air Quality sensor, LAQ4_1.

The battery life projection information, both the steps and the result, are the AI model's response to a query built from sensor data.

The AI Usage Analysis section of the dashboard shows LLM usage information to give a sense of the cost of the integration.

System Overview

The components used to build the Demo Dashboard with AI Integration are shown below in Figure 1, *Datacake to OpenAI System Diagram* .

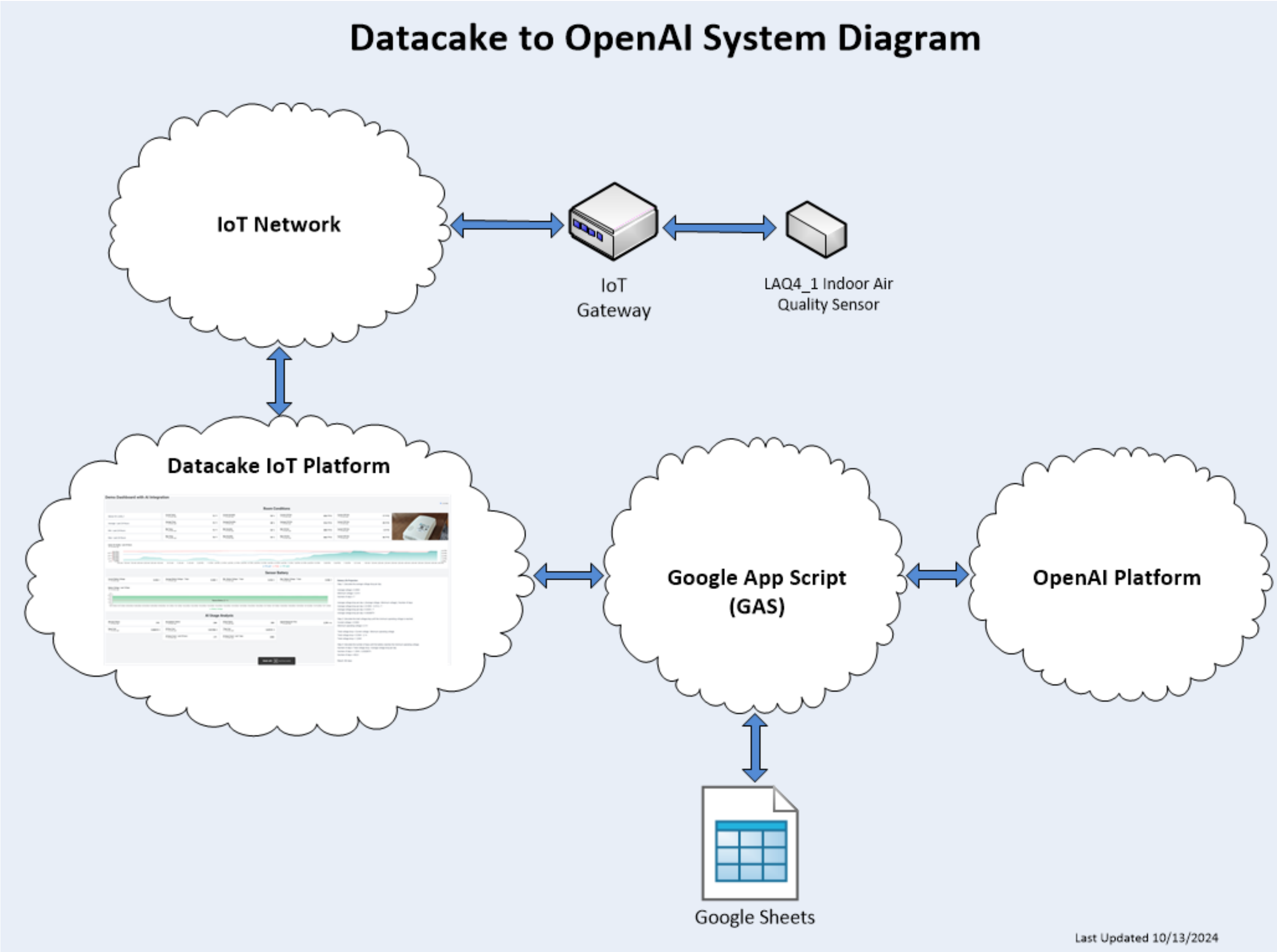


Figure 1 – Datacake to OpenAI System Diagram

In the system, sensor data from sensor, LAQ4_1, flows through the IoT network to the Datacake IoT platform where the sensor data is then displayed in the dashboard. Sensor data from the Datacake platform is pushed via a webhook to a Google App Script (GAS), which constructs the AI query, sends the query to the OpenAI platform, receives the AI response, then sends the AI response back to the Datacake platform where the information in the AI response is populated in the dashboard. Google Apps Scripts are also used to log data to Google Sheets.

The capabilities of the Datacake platform along with the business logic contained in the Google Apps Scripts to utilize the OpenAI capabilities is where the magic happens.

A brief description of each system component is provided below.

LAQ4_1

LAQ4_1 is a battery powered [Dragino LAQ4 LoRaWAN Indoor Air Quality Sensor] that measures Total Volatile Organic Compound, equivalent CO2, temperature, and relative air humidity. “LAQ4_1” is the name I have assigned to this device to distinguish it from other LAQ4 devices in my systems.

One nice feature about the LAQ4 is that it has a port on the side that can be used to trigger the sending of measurements (including battery voltage) to the IoT network. This on demand trigger feature was useful when developing the solution. Not all sensors have this trigger on demand capability.

The screenshots below show a top and side view of the device. The hole in the side view is the port that can be used to trigger the sending of a message by using a pen tip or paperclip to depress the button inside the port for 1 – 3 seconds.

(above) Top and side views of a Dragino LAQ4 sensor

LoRaWAN is a low-power, wide area networking protocol built on top of the LoRa radio modulation technique. LoRaWAN wirelessly connects devices to the internet and manages communication between end-node devices and network gateways. LoRaWAN is a long-range, bi-directional communication protocol with very low power consumption. Depending on the application and the specific device, a device can run for multiple years on a single small battery. LoRa uses the unlicensed ISM (Industrial, Scientific, Medical) radio bands for network deployments.

LoRa frequency bands vary by region, with the most common being: EU868 (863-870 MHz) in Europe, US902-928 MHz in North America, and AU915/AS923-1 (915-928 MHz) in regions like Australia and Asia. When buying a LoRaWAN device, it is important to select the model for the region where the device will be used.

IoT Gateway

The IoT Gateway is a device that connects IoT devices such as LAQ4_1 to an IoT network.

There are many LoRaWAN IoT gateway vendors. The gateway used for this project is the RAK7268C by RAKwireless. This gateway can connect to the internet via ethernet cable, Wi-Fi, or cellular.

IoT Network

The IoT Network is used to manage IoT devices. A key component of an IoT network is the network server. An IoT network server enables connectivity, management, and monitoring of devices, gateways and end-user applications.

For the IoT Network and network server, I am using The Things Network (TTN). The RAK7268C has been provisioned on the TTN Network server with Datacake as the endpoint. The LAQ4_1 sensor has also been provisioned on the TTN network server. The result of this configuration work is that data from the LAQ4_1 sensor gets routed to my Datacake account.

Configuring IoT gateways can be quite involved. Fortunately, Datacake has an option that lets you add a gateway to your Datacake account, hiding most of the LoRaWAN details. See more about this here. Included in the Datacake documentation are detailed instructions for setting up the RAK7268 gateway on Datacake.

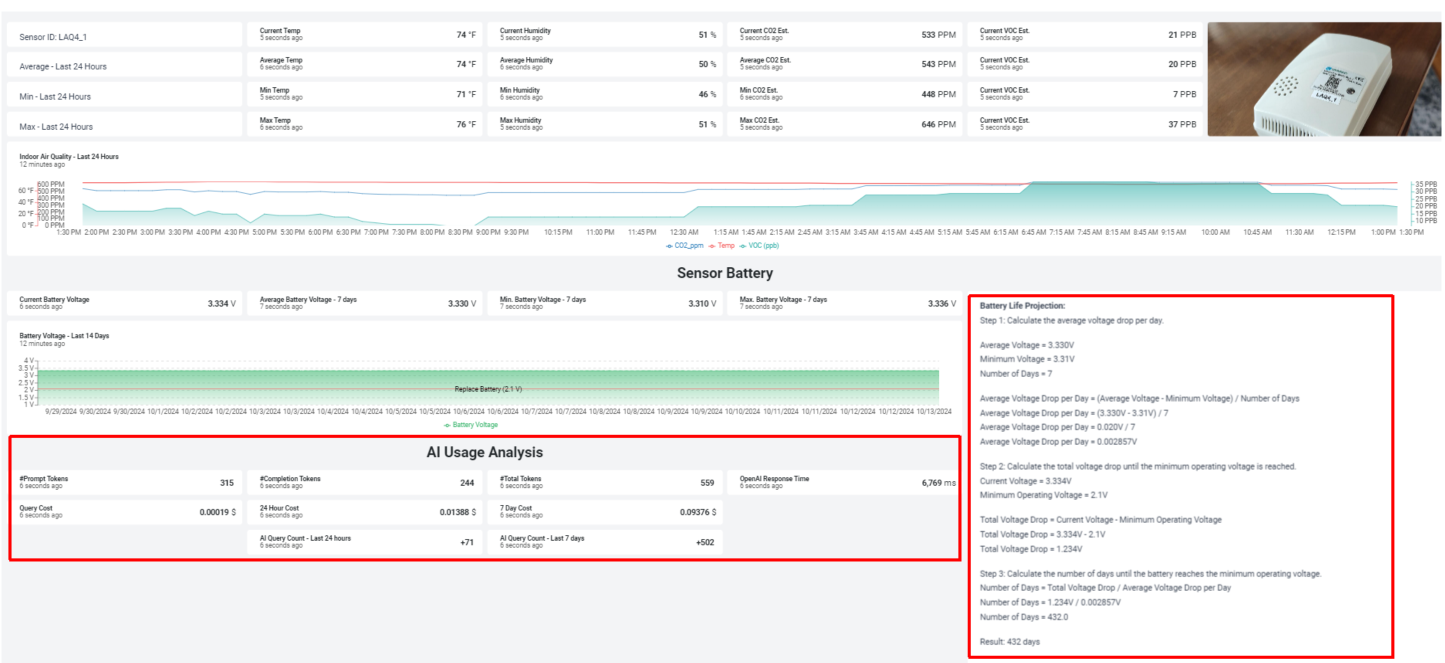
Datacake IoT Platform

Datacake is a low-code IoT platform, which means that it is a platform for building deployable IoT solutions without needing computer coding skills. The Demo Dashboard with AI Integration is built with Datacake. One of the nice things about Datacake is that while it's fairly easy to get started with IoT using Datacake, the platform also has powerful capabilities to build and deploy complex IoT solutions and systems. Another nice thing about Datacake is that the first 5 devices you put on Datacake are free, so it's an inexpensive way to start learning IoT.

Two of the more powerful Datacake capabilities used in this project are Outbound Webhooks and API devices. Outbound Webhooks allow you to send device data contained in the Datacake platform to other applications. In this project I use an Outbound Webhook to send LAQ4_1 data from Datacake to the Google Apps Script that builds and sends the AI query to the OpenAI platform. I also use another Outbound Webhook to send device data to a different Google Apps Script that logs data from each device type to a different Google Sheet device log. The file, *Datacake Outbound Webhook Log – LAQ4.xlsx*, found in the same repository as this document, is a copy of the device log for LAQ4 devices in my systems.

A Datacake API device is a means to get data from other applications into Datacake. A Datacake API device is like a sensor device in Datacake, except that instead of the data coming from a real sensor device, the data is coming from another program. Just like with a sensor device, data for an API device can be put into variables that can be visualized in dashboards. I use an API device to get the query results from OpenAI back into Datacake so that the results can be shown in the Demo Dashboard with AI Integration.

In the Demo Dashboard with AI Integration screenshot shown below, the information in the red outlined boxes is received in Datacake by a single API device. The data source for the API device is the same Google Apps Script that received the LAQ4_1 data and queried the OpenAI platform.



Screenshot – Demo Dashboard with AI Integration

Google App Script

Google Apps Script (GAS) is a rapid application development platform that can be used to create business applications that integrate with Google Workspace. GAS uses JavaScript as the programming language. With GAS, there is nothing to install on your computer; coding, compiling, and program deployment is done through a browser. A Google (i.e. Gmail) account is required to use GAS.

For this project I used 2 Google App Scripts; one script to write sensor data to a Google Sheet device log file. The other script to receive LAQ4_1 data from the Datacake platform, construct the AI query, send the query to the OpenAI platform, receive the AI response back from the OpenAI platform, then finally to send the AI response and other AI-related information back to an API device in Datacake. This second GAS also logs the AI data to a separate Google Sheet log file, *Log_ai_laq4.xlsx*, to aid in debugging. A copy of this log file is contained in the same repository as this document.

OpenAI Platform

The OpenAI Developer Platform is OpenAI's tool for integrating with OpenAI's APIs. When you create an account on the OpenAI Developer Platform, you set up an entity called an organization. In the organization you configure members, billing, model usage limits and data controls. Within the organization you can also create and configure different projects. A project configuration defines how the OpenAI APIs are used for a particular solution. Each project has a unique key used by the application calling the OpenAI APIs. Some of the parameters that can be configured on a per project basis include which model(s) to use, users, and billing limits.

For this project, I configured the OpenAI project to use the gpt-3.5-turbo and gpt-4o-mini models, seeded the account with $20.00 and set a monthly budget of $20.00. At the time I started the project, I had no idea how much it would cost, so I started with a small amount. That turned out to be a good amount as it ended up costing a couple of dollars per month to run and maintain the proof-of-concept system.

Project Learnings

This section captures some of the key learnings from this project.

- **Datacake Webhooks** to **Google Apps Scripts** (or other similar platform), then back to Datacake via **Datacake API variables** is a useful and powerful pattern to extend the Datacake platform
  - Being able to get data out of Datacake and into a software development platform where more complicated business logic can be implemented, then getting the results of the business logic back into Datacake for visualization of the information – is powerful system.
- Datacake webhooks work only on a device's primary workspace
  - Datacake uses the construct of a workspace to organize and group devices. Datacake allows devices to be inherited (or shared) from one workspace into other workspaces.
  - If a device exists in multiple workspaces, you can use Outbound Webhooks to export that device's information only from the device's primary workspace, i.e. the workspace where the device was originally set up in Datacake
  - In this project, for the **Datacake Webhook** to **GAS** to **Datacake API Variable** pattern to work, the webhook is assigned to the LAQ4_1's primary workspace and the API variable used to receive LLM data is set up in a secondary workspace. The LAQ4_1 device is inherited into that same secondary workspace. The Demo Dashboard with AI Integration, which shows data from the LAQ4_1 device and from the LLM, is built on the secondary workspace.

- It's hard to get historical data out of the Datacake platform
    - For the AI query, I wanted to include historical information about the average, minimum, and maximum battery voltage for the past 7 days.
    - In the Datacake platform, it's easy (i.e. no coding is necessary) to show historical information in a dashboard using a Value widget. The screenshot below illustrates this.

Sensor Battery

Current Battery Voltage 17 minutes ago	Average Battery Voltage (1 wk) 17 minutes ago	Min. Battery Voltage (1 wk) 17 minutes ago	Max. Battery Voltage (1 wk) 17 minutes ago
3.330 V	3.328 V	3.310 V	3.334 V

Screenshot – Datacake value widgets showing the Average, Min, and Max battery voltage for the past 7 days

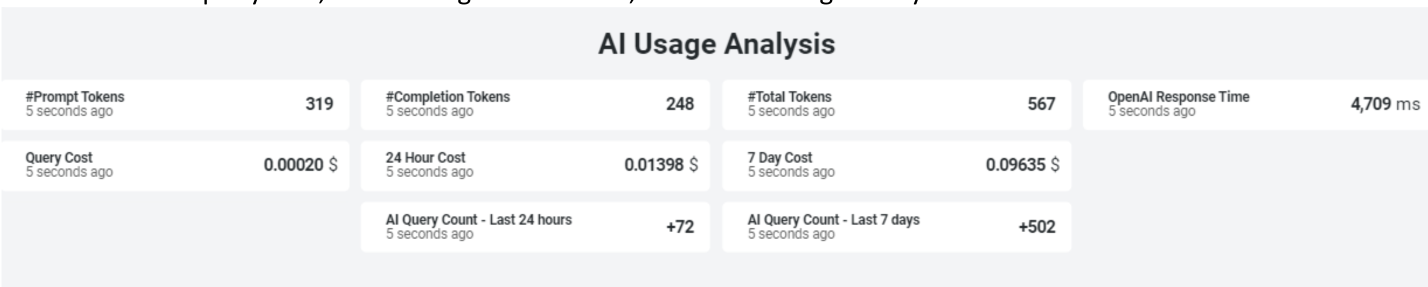
-
    - Unfortunately, I was unable to extract historical battery voltage information from the Datacake platform for use in my Google Apps Script.
        - One approach I tried was to add logic to the device decoder to extract the min, max, and average battery voltage into variables that would be exported through the webhook, but I was unable to get that to work.
        - Another approach I tried was to use a GraphQL API call from the Google Apps Script back into Datacake to get the information, but I was unable to find enough documentation to figure out how to make that work.
        - What I did manage to do was to add logic to the Google Apps Script, to access my LAQ4_1 Google Sheet log and calculate the 7-day min, max, and average battery voltage from the information in the log.
- How OpenAI API Usage Costs are Determined
    - OpenAI's API usage cost is determined by the number of tokens used in the input (the query) and in the output (the response).
    - A token is a piece of a word. A token is about 4 characters.
    - Each OpenAI model has a set per 1000 tokens input and output pricing. You can see the pricing here: https://openai.com/api/pricing/
    - At the time I am writing this document, the pricing for the gpt-4o-mini model, which is the model being used for this project, is as follows
        - $0.000150 / 1K input tokens
        - $$0.00060 / 1K output tokens
    - The screenshot below, taken from Log_ai_laq4, shows the query and response size and cost calculations. The cost for the single query and response was $0.0002 or .02 cents.

E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
AIQUERY	AIRESPONSE	OpenAI API Finish Reason	GASMESSAGE	DATACAKE RESPONSE	OpenAI API Response Time (millseconds)	GAS Execution Time (millseconds)	AI Query Length (characters)	AI Response Length (characters)	Prompt Tokens	Completion Tokens	Total Tokens	Query Cost ($)	Response Cost ($)	Total Cost ($)
I have a Dragino LAQ4 indoor air quality device with a 4000mAh Li-SOCl2 (Lithium Thionyl Chloride) battery. The ambient temperature is 66.56 °F and the current battery voltage is 3.327. The minimum operating voltage of the device is 2.1 V. The sensor transmits a message every 20 minutes. Over the past 7 days, the average, minimum, and maximum voltage readings have been 3.328V, 3.31V, and 3.334V respectively. Using the supplied current voltage, 7-day average voltage, and 7-day minimum voltage, calculate the number of days until the battery voltage will hit the minimum operating voltage level using the following 3 steps: first calculate the average voltage drop per day by finding the difference between the average and minimum voltage, then dividing that by the number of days specified Second, calculate the total voltage drop until the minimum operating voltage is reached by finding the difference between current average voltage and the minimum operating voltage. Third, calculate the number of days until the battery reaches the minimum operating voltage by dividing the result of step 2 by the result of step 1. Show the calculation and result for each step, but do not include any additional commentary, rounding, or changes to the result in any way.	Step 1: Calculate the average voltage drop per day. Average voltage = 3.328V Minimum voltage = 3.31V Number of days = 7 Average voltage drop per day = (Average voltage - Minimum voltage) / Number of days Average voltage drop per day = (3.328V - 3.31V) / 7 Average voltage drop per day = 0.018V / 7 Average voltage drop per day = 0.00257142857V Step 2: Calculate the total voltage drop until the minimum operating voltage is reached. Current voltage = 3.327V Minimum operating voltage = 2.1V Total voltage drop = Current voltage - Minimum operating voltage Total voltage drop = 3.327V - 2.1V Total voltage drop = 1.227V Step 3: Calculate the number of days until the battery reaches the minimum operating voltage. Number of days = Total voltage drop / Average voltage drop per day Number of days = 1.227V / 0.00257142857V Number of days = 477.0 Result: 477 days	stop	GAS message placeholder	thanks	5082	10675	1291	901	315	248	563	0.00005	0.00015	0.0002

Screenshot – entry from *Log_ai_laq4* showing cost calculation for an LLM query and response

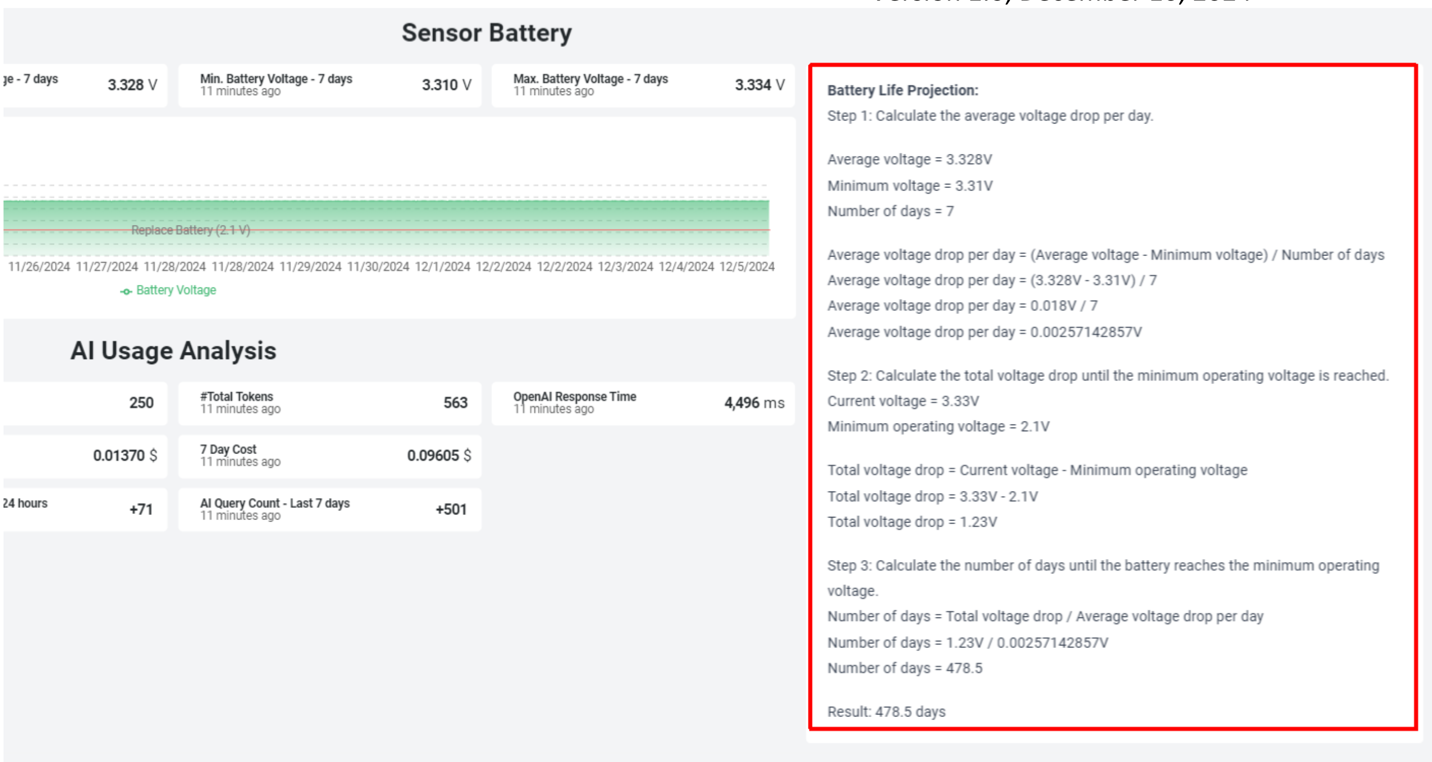
- It's pretty cheap to use the OpenAI API: ~10 cents per week for [Demo Dashboard with AI Integration]()

- When I started the project, I had no idea how much the LLM integration would cost. Once I got the system working, it turned out that the cost is approximately 1.4 cents/day or 10 cents/week.
    - The LAQ4_1 sensor sends an update every 20 minutes, or 3*24=72 updates per day
    - 72 updates/day * $0.0002/update ~= $.0144/day
- The AI Usage Analysis section of the dashboard shown in the screenshot below shows the most recent query cost, the running 24-hour cost, and the running weekly cost.



Screenshot – AI Usage Analysis information from the [Demo Dashboard with AI Integration]

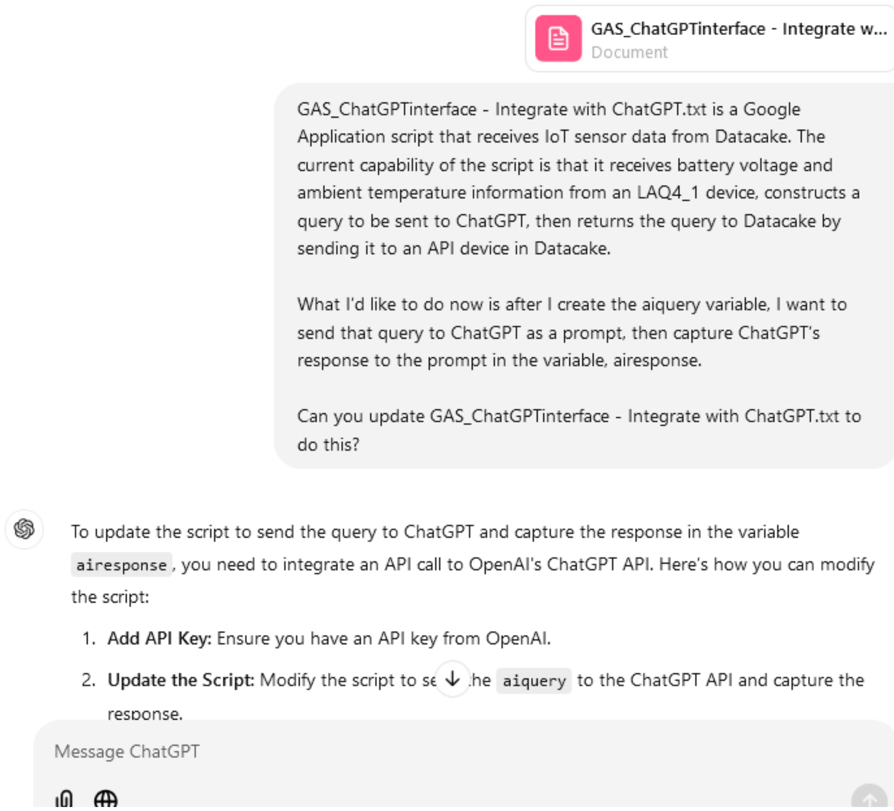
- LLM models are advancing quickly
    - Between late June when I started the LLM integration work and late august, when I finished the LLM integration work, a better and cheaper LLM was released
    - When I began the LLM integration in late June, the recommended OpenAI model was gpt-3.5-turbo with the following pricing:
        - Input: $0.0005 / 1K tokens
        - Output $0.0015 / 1K tokens
    - By the time I finished the AI integration work in late August, the recommended OpenAI model was gpt-4o-mini with the following pricing
        - Input: $0.00015 / 1K tokens
        - Output $0.0006 / 1K tokens
- Getting an LLM to format the output in a specific way can be hard
    - In the dashboard, the wording in the Battery Projection box is taken directly from the AI response.



Screenshot – Showing the Battery Life Projection portion of [Demo Dashboard with AI Integration]

- What I had wanted to do was to have the Result information (e.g. 478.5 days) presented at the very top. I spent a week trying to change the query to show the result information at the beginning of the response, but every time I modified the query to do that, it broke the result calculation. I finally gave up.

- Using an LLM such as ChatGPT to help with the coding sped up the coding development significantly – at least by a factor of 4 as a rough estimate
    - It's been quite a few years since I was a software engineer and I was never a JavaScript expert, but because I had ChatGPT to help write the code, my rustiness and lack of knowledge didn't matter.
    - Much of the work for this project was developing the Google Apps Script. The script has to do things such as
        - Look into a Google Sheet log file of LAQ4_1 sensor messages to calculate the min, max, and average battery voltage for the past 7 days
        - Construct a query that inserts sensor measurements received via the Datacake webhook and calculated from the LAQ4_1 log file
        - Make the query call to the OpenAI platform
        - Receive the response back from the OpenAI platform and place the response and response parameters into a data element to be pushed back to the Datacake platform
        - Construct and send an HTTP message back to the Datacake platform
    - I used ChatGPT to help write all the functionality listed above. I also used ChatGPT to debug and fix errors. Many times, I would upload my latest working version of the script to ChatGPT and then tell the ChatGPT the additional functionality I wanted in the script or the error that was occurring – and ChatGPT would add the requested functionality or code fix. The screenshots below show a couple of the queries I used.



GAS_ChatGPTInterface - Integrate w...
Document

GAS_ChatGPTInterface - Integrate with ChatGPT.txt is a Google Application script that receives IoT sensor data from Datacake. The current capability of the script is that it receives battery voltage and ambient temperature information from an LAQ4_1 device, constructs a query to be sent to ChatGPT, then returns the query to Datacake by sending it to an API device in Datacake.

What I'd like to do now is after I create the aiquery variable, I want to send that query to ChatGPT as a prompt, then capture ChatGPT's response to the prompt in the variable, airesponse.

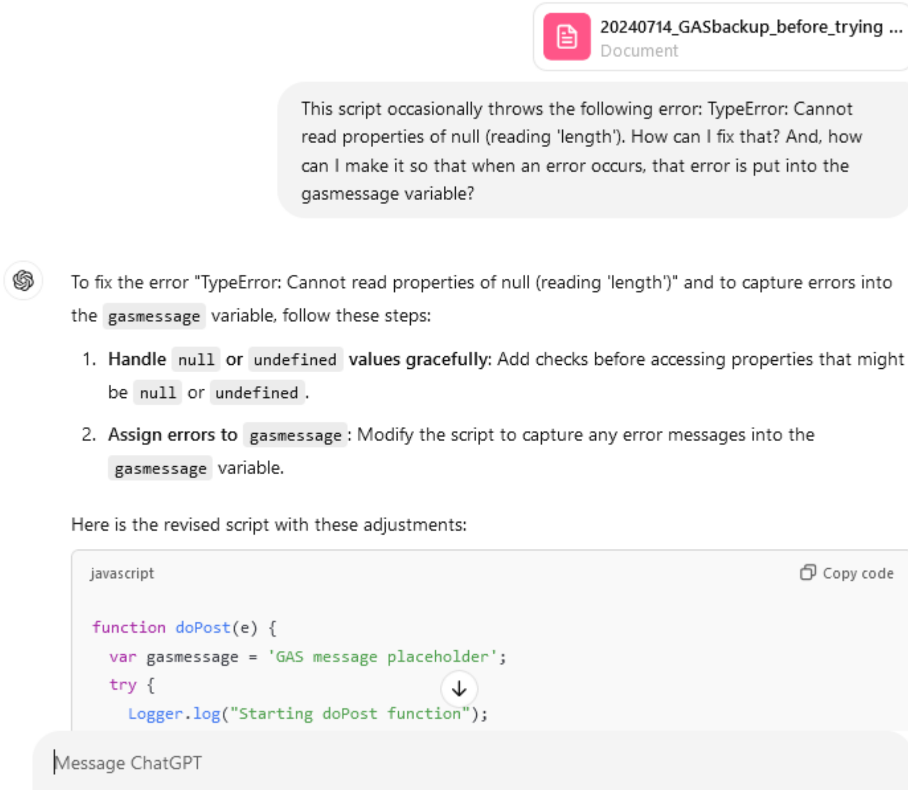
Can you update GAS_ChatGPTInterface - Integrate with ChatGPT.txt to do this?

To update the script to send the query to ChatGPT and capture the response in the variable `airesponse`, you need to integrate an API call to OpenAI's ChatGPT API. Here's how you can modify the script:

1. **Add API Key:** Ensure you have an API key from OpenAI.
2. **Update the Script:** Modify the script to se[…]he `aiquery` to the ChatGPT API and capture the response.

Message ChatGPT

Screenshot - ChatGPT session to add functionality to make the API call to the OpenAI platform



Screenshot - ChatGPT session to fix an occasional script error

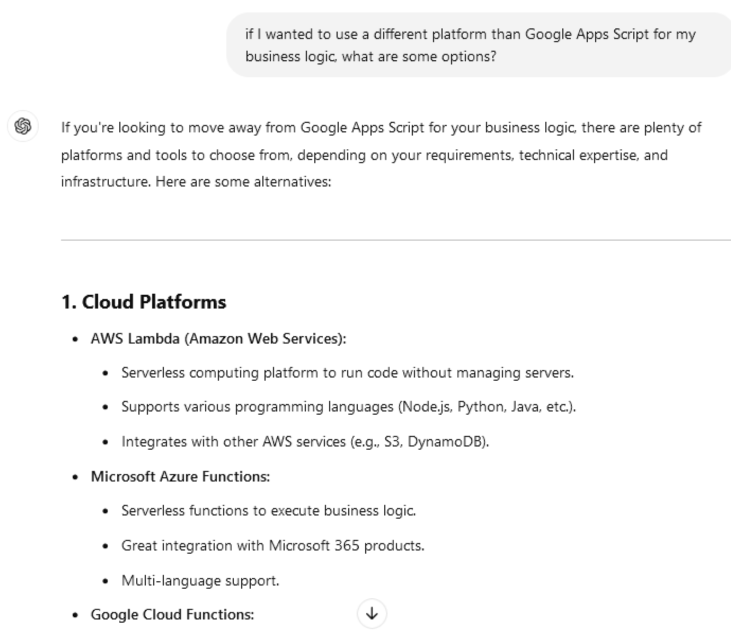
- The battery life estimator that I ended up with doesn't need an LLM
    - There is no need to integrate my Datacake data with an LLM to get a usable battery life estimator
    - At the start of the LLM query development portion of this project, my plan was to build a query with as much information as I thought the LLM would need to come up with a good estimate. That information included the sensor type, battery type, room temperature, current battery voltage, and 7-day minimum, maximum, and average battery voltages.
    - Where things ended up was that the most reasonable looking estimates came from using a simple formula, so the query evolved into directing the LLM to perform the formula calculation.
    - Looking back, a faster and cheaper solution would have been to use the LLM to come up with a good estimation equation, then to simply embed that equation into the Google Apps Script instead of integrating the Google Apps Script with the LLM.
- Finding business value from integrating an IoT platform with an LLM is not elementary
    - When I say, business value, I mean 'enough usefulness that the amount customers are willing to pay is enough for the entity providing the capability to continue providing that capability'.
    - This proof-of-concept project shows that it is possible to query an LLM with IoT sensor data and then to display the query results in an IoT dashboard.
    - This proof-of-concept project does **not** show that using an LLM to implement a battery life projection provides business value – at least for the way the projection is being calculated in this project.
    - If it was determined that the information provided by the battery life projection formula did provide business value, then that capability could be implemented without an LLM.
    - It may be possible to use an LLM to come up with a **better-enough** battery life projection than the formula used in this project such that there is business value in an LLM-based projection, but that will require a better understanding of battery performance characteristics as well as the use of a purpose-built LLM.

  - Or thinking bigger, perhaps business value can be found in monitoring the characteristics of a battery system such as the battery system in an electric car or a solar-powered home or even an industrial scale battery system. But that also will require understanding of how these systems are used and function and most certainly will require a purpose-built LLM.
- Beyond the technical learnings from this project, another key takeaway is a reminder of the challenges that many companies around the world are working hard to overcome:
  - Finding business value from an IoT solution
  - Finding business value from an LLM solution

Potential Next Steps

Listed below are some follow-up areas to explore if one wanted to extend this project.

- Perform the battery projection calculation in the Google Apps Script instead of through the LLM integration
- Go further with battery life modeling. Use battery voltage data in the device logs to train the LLM on battery usage instead of using a formula to estimate remaining battery life
- Integrate the Datacake platform with an LLM from an LLM provider other than OpenAI
- Replace GAS with a different platform



Screenshot – ChatGPT query asking for GAS alternatives

- Figure out how to scale the proof-of-concept model to support multiple devices or multiple workspaces within Datacake
- Make the system more secure, especially in the area of OpenAI key management
- Optimize (shorten) the query to see how low you can get the cost with the same query result
- Instead of using the LLM to predict sensor battery life, use the LLM to do something useful with sensor measurements (temperature, humidity, VOC, CO2)
- Replace the LAQ4 sensor with a different sensor type

Conclusion

The ability to get data out of the Datacake platform and then back into the platform after performing business logic calculation with the data (e.g. using an LLM on the data) is a powerful way to give end users more useful information from the data being collected by the IoT system.

Now that the basic plumbing aspects of integrating Datacake with an LLM are understood, the focus can turn to finding IoT-LLM solutions that provide business value.

You can reach me at mikeb@hitechdb.com.

Resources

Listed below are the resources referenced throughout the document.

- https://github.com/mikedsp/datacake-openai-integration - GitHub repository where this paper can be found
- Demo Dashboard with AI Integration - Public, live dashboard
- Datacake, low-code IoT platform
- OpenAI developer platform
- Google Apps Script