



AN INFLUXDATA CASE STUDY

How MOXIE IoT Uses Swift, MQTT and InfluxDB to Create a Modern iOS IIoT Monitoring Solution

Austin Gurley
CTO, MOXIE IoT



NOVEMBER 2020 (REVISION 1)



Company in brief

MOXIE IoT are the creators of the MoxieWORLD platform which collects, stores and analyzes industrial sensor data and visualizes it in real time in their iOS app. Their solution tracks the movement and activity of factory assets, and their goal is to improve performance and safety. These assets include overhead cranes, forklifts, pallets and people. Their iOS app provides their clients with real-time and historical data at their fingertips; the app provides their customers with intuitive overhead mapping of their factories. MOXIE IoT's industrial indoor tracking solution was created using Python, Swift, MQTT, InfluxDB and AWS EC2.

Case overview

MOXIE IoT has created a modern IIoT monitoring platform by using the InfluxDB platform. By adopting industry standards and cutting-edge technology, they have created MoxieWORLD, a modern iOS app, to provide manufacturing companies with a single source of truth. The iPad app enables their clients to visualize and analyze industrial factory data in real time. MOXIE's team has built its solution on InfluxDB and various other technologies. They are also using MQTT, Python, Swift, AWS EC2 and Ultra-Wideband technology to accurately measure and track any object, device, personnel or machinery. MOXIE IoT are the creators of an app-centric industrial IoT monitoring platform, and they create digital twins for their clients. MOXIE has built an industrial monitoring tool on InfluxDB.

The business challenge

MOXIE IoT likes to segment their industrial clients' teams into three categories: operations, technology and maintenance. MOXIE aims to provide solutions that benefit all three teams, which makes it easier to integrate across teams and to get an entire team to adopt new technologies. The team wanted to ensure that their platform addresses the needs of all the customer stakeholders to ensure optimal adoption:

- A manufacturer's operations team is interested in enhancing utilization, reducing downtime and improving safety as well as management of personnel and resources.
- Their technology team is keen on strengthening their predictive maintenance and providing better business insights. This team is the one using the IoT sensor data to create dashboards for the

operations to demonstrate the best ways to become more efficient.

- Lastly, maintenance teams are interested in better understanding service intervals, distance traveled and overload events all with the end goal of keeping the equipment running.

The maintenance and operations teams are both interested in minimizing downtime; however, the data they use and their approaches are different.

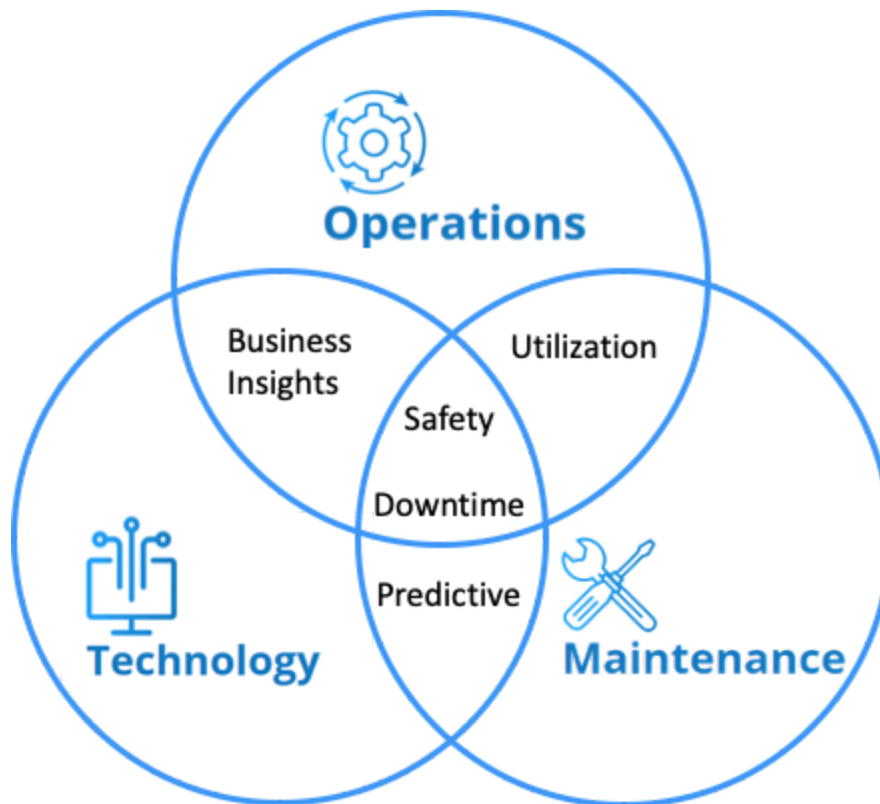


Figure 1: MOXIE's approach to industrial sensor monitoring

The MOXIE team desires to provide its customers with stellar 3D visualizations of their factory floors with low latency (2-10 seconds). The solution needed to be easily configurable so that it can help all teams. Additionally, their clients needed to be able to create and manage projects within the app. Moxie aims to provide the following to their clients:

- Access to historical and real-time data
- An intuitive interface to be used by the entire team to understand system performance

- Ability to understand and compare performance
- Better insights to all areas of their business.

The technical challenge

Prior to selecting InfluxData's InfluxDB platform, MOXIE IoT tried the entire AWS suite of products. As the background of MOXIE IoT CTO Austin Gurley, as well as his team's, is in mechanical engineering, they were comfortable with embedded devices. MOXIE was initially attracted to AWS via Google ads, and the team used the Amazon suite for about a year. They started with AWS's IOT system and DynamoDB. The challenge with the AWS IoT platform is that it is only great for the hardware devices they support. It is end-to-end and secure, and AWS helped them along the way. However, MOXIE realized that sometimes AWS was almost too helpful. Gurley points out that the computer processing power Amazon expected from IoT devices is quite high. He further explains that if you're looking for something with the power of a Raspberry Pi or an embedded computer, Amazon's IoT solutions might be appropriate, but not for smaller, lower-power devices. In MOXIE's case, they were interested in extracting data from embedded microcontrollers systems — they barely need network access. For example, MOXIE wasn't looking to run a full Linux computer with AWS. MOXIE acknowledged that AWS is great for certain applications; however, it became clear they needed a different solution due to the power of their IoT devices.

DynamoDB seemed like a good solution as it seemed like it could handle lots of small data points. However, MOXIE IoT's team realized it was built to query in a sorted manner. Once again, it wasn't best suited for their industrial IoT use case. It didn't work great for data that's spread equally across time, and it didn't provide them with the ability to drill down into specific time intervals. MOXIE couldn't look back historically at their data, look at certain segments of time, or look at the data for a specific time range or device.

In addition to being able to provide 3D visualization of their clients' sites, MOXIE needed to be able to export data out to other solutions. For example, Power BI is popular with larger industrial companies. As Austin Gurley pointed out: "Every industrial business of a certain size will use Power BI". MOXIE was aware they needed to extract data from their app in various ways, to address the needs of their customers. MOXIE wants their customers to be able to use their platform as a single point of reference of how things look; their solution should be a single pane of glass of their facilities. Once the data is all in one spot, being able to address specific teams' needs (i.e. Power BI) is important for MOXIE as well.

The solution

MOXIE's platform consists of four primary technologies that make up the network flow and starts with ultra-wideband (UWB). They rely on UWB indoor positioning, MQTT for real-time data streaming, InfluxDB for historical time series data collection and analysis, and their app-centric interface which was programmed using Swift.

This is used for indoor positioning; it's like GPS but for indoors. It consists of two pieces: mobile "tags" and stationary "anchors". The tags are the IoT devices which are attached to mobile equipment and act as its own self-gateway. The sensors transmit data across Wi-Fi or cellular. The devices start by transmitting its position and then send other associated data. All of this data is directly associated with a physical device that moves around a client's facility; some of this metadata includes:

- Motor spindle load
- Temperature
- Humidity
- Air volatilities

A typical GPS' accuracy is about 1-2 meters, providing excellent asset tracking in outdoor environments. However, as soon as a GPS system is taken indoors, its accuracy drops due to interference; the system's signals are blocked from the satellites. UWB is the solution to GPS shortages. As Austin Gurley points out, they use ultra-wideband technology to create their own "constellation of references for the tags to use for positioning".

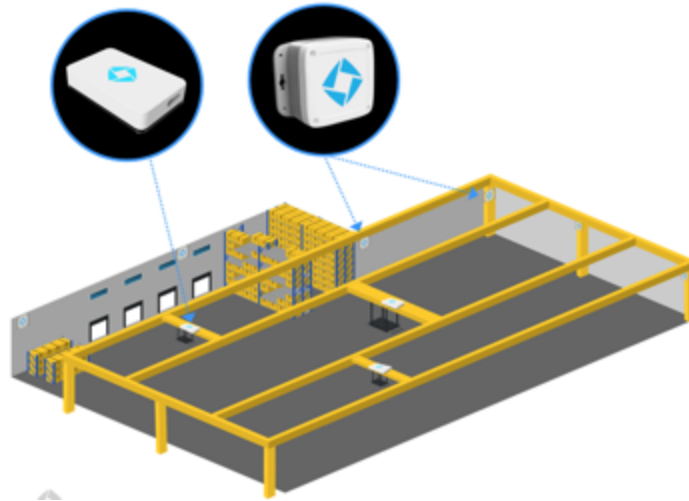


Figure 2: Demonstrating the location of sensors within a facility



Anchor
Fixed Reference Point



Tag
Mobile Tracker & Gateway

Figure 3: Examples of anchors and tags

These reference points that make up the map or “constellation” are called anchors. They are scattered around their clients’ sites at 50-100 meter intervals around the edges of the walls. The tags can then find the locations of all the anchors. MOXIE’s solution uses a combination of UWB and GPS to track the movement of all machinery — like forklifts, etc. Their platform can track these items as they move throughout a building, outside and back into the facility; this is all done seamlessly. Many of their clients have one or two buildings, and MOXIE enables them to track the workflow of their entire facility — indoor and outdoor. MOXIE’s solution of tags and anchors provides their clients with indoor positioning with 10 cm of accuracy within a 50+ meter range. An unlimited number of anchors can be added to extend the size of the workspace.

MOXIE’s devices are mounted to the top of their customers’ cranes, forklifts, etc. The sensors have UWB tracking hardware, application-specific sensors, and act as a self-gateway or a hub. IoT data for each individual device is transmitted to the internet using LTE or Wi-Fi and is sent for longer-term data storage. To form a useful Digital Twin for IoT Data, location is just as important as timestamps — MOXIE wants to ensure their users can visualize their factories as a place where they can see movement, how things are moving, where they are going. It isn’t just charts showing changes over time. They have data that has independent access to time, position and dependent data. Independent data includes the time and position. Some examples of dependent data are: speed, motor load, voltage sensors toggling on and off, etc. Countless other things can be measured and collected from sensor data. MOXIE wants to ensure they can pick a time range and plot their dependent data against time and then against the position in the facility.



Figure 4: Screenshot of MOXIEworld - MOXIE IoT's iOS app

Why MQTT?

MOXIE uses MQTT messaging protocol to collect sensor data from its IoT devices. MQTT is a publish-subscribe message system that allows a single-broker computer server to act as an interface between sensors and endpoint devices. MQTT is beneficial for IoT metrics as it's low power in computing, battery life and internet bandwidth. Many telecom carriers, like AT&T, offer Cat-M1 (which is a LTE category and can transmit IoT data at low bandwidths). Cat-M1 cell plans are generous, and although there is a maximum amount of data that can be sent per second, Moxie's IoT devices typically stay well within Cat-M1 data rate limits.

A common solution to addressing battery life issues on IoT devices is to log a day's worth of data and dump it all at once into a historical database or a historian. However, the new Cat-M1 plans accommodate sending a consistent trickle of data. MQTT is a great solution for this. MOXIE uses MQTT to enable streaming of their sensor data; this provides access to their data in real time, and they are able to run this system at low power.

MQTT is great at handling IoT devices for special features like "keep alive" timing for when a sensor dies. This provides an interesting challenge as it is nice to know when a sensor dies and what happened, but of

course the sensor isn't transmitting because it died. The MQTT broker can send a message to all its subscribers in the event that the signal is lost or something else has disconnected. Another benefit of using the MQTT protocol is that a single broker can handle thousands of devices and can be easily partitioned to have multiple customers and multiple projects all in one broker. The topic filters enable MOXIE to safely isolate users. Once MOXIE is happy with their MQTT setup, they can easily partition it and access their preferred programming language; it's very easy to customize the broker so that your devices can use this protocol. For example, there were drivers which made it easy for MOXIE to code in Python, C++ and Java.

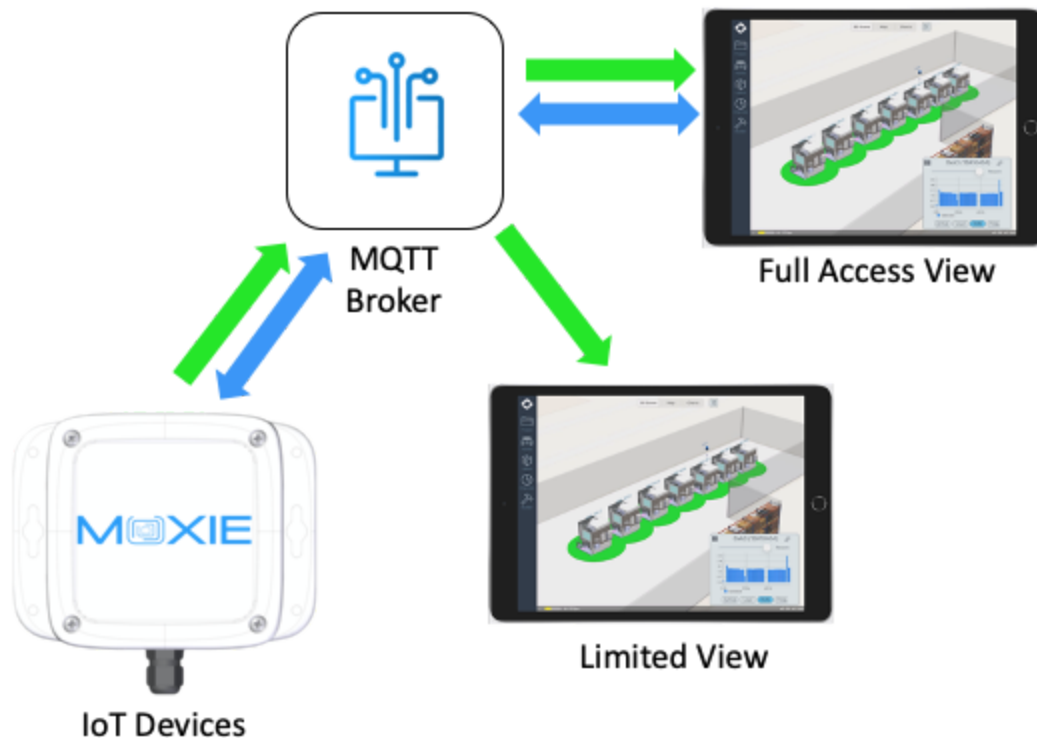


Figure 5: Demonstrating MOXIE IoT's usage of the MQTT broker

There isn't an industry standard for how MQTT data should be organized. Typically, MQTT data packets have two primary elements: topic structure and payload. The topic structure can be described like a file system: there's a slash-divided string that is like a file folder where a single file resides in. The file is called a payload. If an IoT device has permission, at any time, it can put a single small payload file anywhere in the file system it wants. The next time anything is written to the same location, it will be overwritten. As any device can overwrite another one, if given permission to do so, the data structure can become quite deep. The data is organized based on the topic, and the payload contains the information being carried from one place to another. Moxie's MQTT structure allows them to stream data from a customer's facility to their app

and keep it secure and partitioned from other customers.

Their technique for organizing IoT data across multiple customers is detailed in Figure 6.

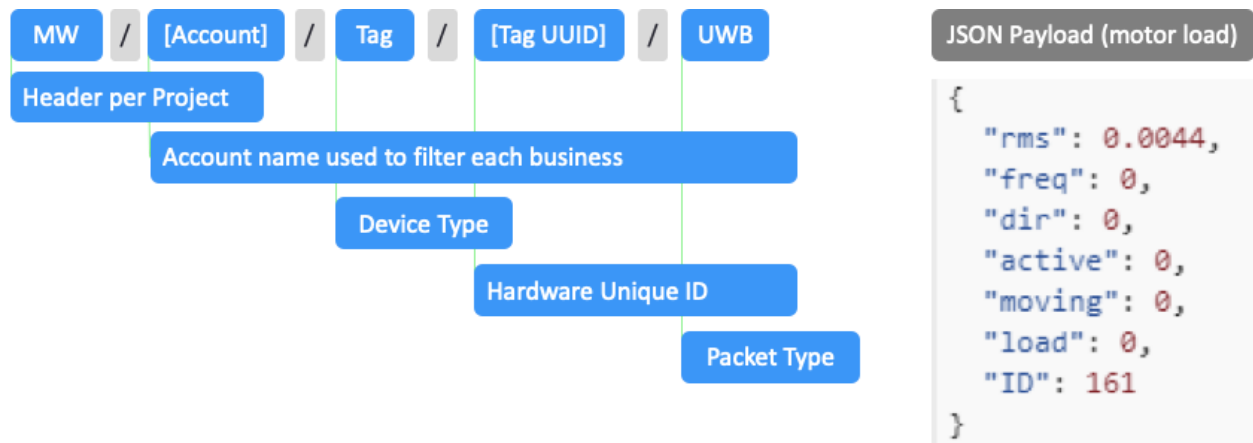


Figure 6: MOXIE's MQTT Stream

MQTT stream levels:

- **Header:** MOXIE chooses to start just a header as it lets them have multiple projects on a single MQTT broker. It might not be necessary, but it's a nice way to organize.
- **Account:** The account name is the next level in the topic, and this is how permissions are set for the stream. MOXIE's clients will be given access to all the topics below **MW / [Account]**. This is a great way for MOXIE to partition everything based on customer logins and manage permission and accessibility.
- **Tag:** The next level is device type, and MOXIE has a few of these. For example for UWB, their tags, the movie devices and anchors will have stationary information about their location within the facility.
- **Tag UUID:** This level consists of a serial number of unique identifiers for the tag. Serial numbers enable MOXIE customers to rename devices, set custom configurations, etc. However, a serial number is a perfect way to track all the devices, regardless of what changes through the course of the sensor's "life".
- **Payload Type:** Since different tags may have multiple sources of data, the payload type is denoted as the last element of the Topic to identify how it can be decoded.
- **The JSON Payload** contains all sensor data for each timestamp. For instance, in the example above the JSON Payload is for motor load. This is used to understand torque, spinning speed, etc. These data packets are easily sent to InfluxDB for storage.

Why Swift?

Swift is the general-purpose programming language for macOS, iOS, watchOS and tvOS. Prior to selecting Swift, MOXIE considered React Native or other cross-platform tools. While these have a place and are powerful, Gurley and his team decided they weren't best suited for connecting to real-world devices. They needed a solution that could connect to physical hardware with Bluetooth. They discovered that once set up, their React Native solution quickly falls apart in the real world. They were having to make individual input and output drivers for their hardware for every platform they were trying to support.

As a startup, they were keen to stick to one platform. They'd rather be rockstar iOS Swift programmers rather than average programmers across multiple languages. They also discovered that once they were in the Swift language, there were so many other additional tools available in iOS including powerful 3D visualization tools for maps and map tiling. Additionally the inputs and outputs were robust; there are Apple and third-party drivers to support Bluetooth Low Energy, MQTT, etc.

As MOXIE IoT's solution is an iOS native app, using Swift was a natural decision. They are streaming sensor data from their clients' factory using MQTT to store it. They are either displaying the data live or using it for social analysis — all within their iPad iOS app.

```
// query using the Flux API
// https://v2.docs.influxdata.com/v2.0/query-data/execute-queries/
func query(_ query: String, token: String, completion: @escaping ((Data?,Error?)->Void)){

    let dbUrl = "https://us-west-2-1.aws.cloud2.influxdata.com"

    var userRequest = URLRequest(url: URL.init(string: "\(dbUrl)/api/v2/query?org=\(organization)"))
    userRequest.httpMethod = "POST"
    userRequest.setValue("Token \(token)", forHTTPHeaderField: "Authorization")
    userRequest.setValue("application/vnd.flux", forHTTPHeaderField: "Content-type")
    userRequest.setValue("application/csv", forHTTPHeaderField: "Accept")
    userRequest.httpBody = query.data(using: .utf8)

    let task = URLSession.shared.dataTask(with: userRequest) { data, response, error in
        guard error == nil else {
            print("error: \(error!.localizedDescription)")
            completion(nil,error!)
            return
        }
        guard let data = data else {
            print("no data returned")
            completion(nil,nil)
            return
        }
        completion(data,nil)
    }

    task.resume()
}
```

Figure 7: Example of InfluxDB query in Swift

Why InfluxDB Cloud?

InfluxDB was the ideal solution for MOXIE IoT as it is [purpose-built](#) for time series data. MOXIE quickly realized that InfluxDB is designed specifically for collecting time-stamped data and enabling historical analysis. Being able to query their data within specific time ranges and selecting specific data points were benefits of switching to InfluxDB. Gurley points out that the MOXIE team liked that they can “offload the work of filtering the data in normal signal processing concepts, moving average filters, sorting, max and min”. They appreciate that they can offload this to the queries they’ve made and receive the data in a way that’s easy to understand. Being able to push the data to their app and visualize their data however they see fit was a big draw for MOXIE. MOXIE IoT is very happy with how InfluxDB handles IoT data.

MOXIE also uses InfluxDB Cloud as it is easy for them to estimate the cost. It was difficult with AWS to determine their spend. It was easy for MOXIE to determine how much it would cost to host x number of devices, stream data and x samples per minute with InfluxDB.

MOXIE uses Fluxlang to query their historical data. Additionally, they are able to provide their customers with direct access to their InfluxDB buckets. Their clients can pull data out via three ways: through the MoxieWORLD app, directly through InfluxDB buckets, and their clients can access the live data through the MQTT stream.

MOXIE stores data in InfluxDB every 10 seconds. This is great for them as once the data is in the time series database, they can query it. These queries don’t overload their devices as they don’t have to download a lot of data and create filters or summaries. They can design a Flux query themselves and with any desired output.

“It was easy for us to estimate the cost of using InfluxDB Cloud based on the number of sensors.”

Austin Gurley, Founder and CEO, MOXIE IoT

Technical architecture

At the lowest level, there is the embedded system which consists of one of MOXIE's devices — one of the self-gateways. It connects to the internet using Wi-Fi or LTE Cat-M1; the only access point to the internet is over MQTT. Gurley points out that it's common practice to send live data over MQTT and send historical data directly to the time series database, but MOXIE has chosen to architect their solution a little differently. They want to minimize the bandwidth and computing needed on their IoT devices, so it's only connected over cellular or Wi-Fi.

As shown in the diagram below, MOXIE has three data access points for their customers:

1. MQTT stream is for live and data streaming. The bandwidth for MQTT is 240 bytes per update at a rate of 1 Hz. This means it averages out to 622 MB per month per device.
2. InfluxDB is for historical data and querying. The bandwidth for InfluxDB is 80 bytes per update at a rate of .1 Hz. This equates to 20.7 MB per month per device.
3. App-interface is for digital twins and dashboards.

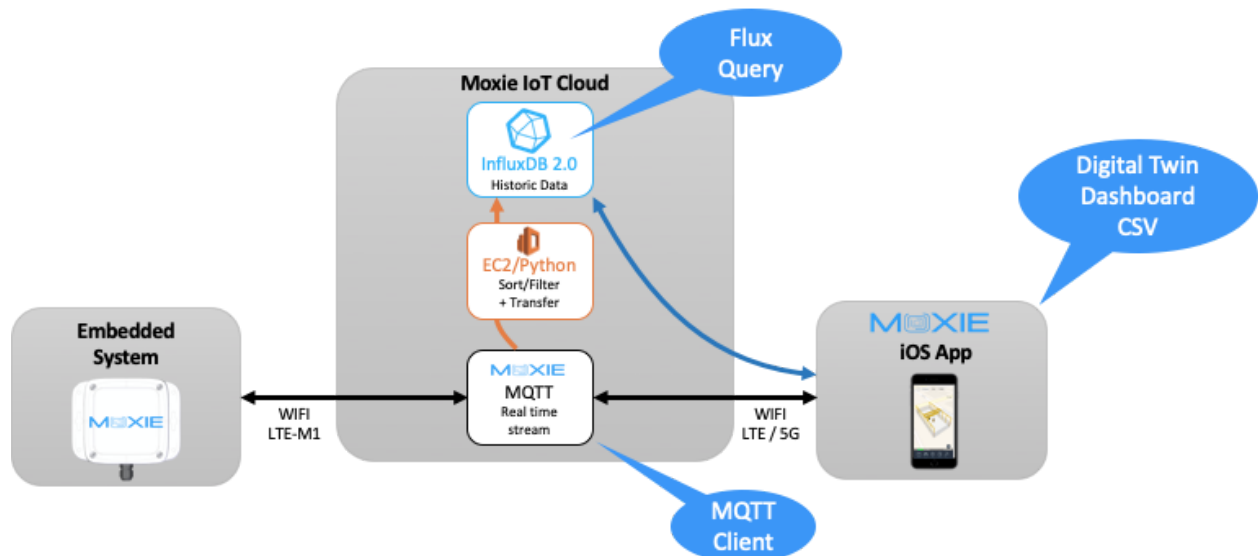


Figure 8: Overall network layout of MoxieWORLD

Gurley points out that the typical data rate for an IoT device is about one per second. For a typical system, the payload includes the position and 12 other data points. These data points include position, speed,

accuracy of the signal, vibration level and temperature. All of this adds up to about 240 bytes per payload that MOXIE sends, and it's updated at about 1 Hz for most of their systems. This is more than enough data to result in nice 3D visualizations; the resolution is great for display and is useful for looking back at the last hour or day's worth of data. This granularity meets their customers' needs; it adds up to about 600 MB per month per device. This is well within the limits and as long as the data stream is a continuous trickle, it is within the limits of more Cat-M1 plans without occurring overcharges for data.

MOXIE doesn't store every data point as it adds up pretty quickly for a single sensor, and this becomes an issue if they want to store it all. Instead, they split it up. The MQTT broker plus an AWS EC2 instance runs Python which is able to collect the data, sort it based on customers and filter it however needed. They store metrics that are collected every 10 seconds; these data points are stored in InfluxDB. They had tried using Lambda but didn't find it efficient. MOXIE likes to run on EC2 with Python because it enables them to create custom filtering. For those who want to use MQTT and Python, Gurley explains they should be using the Paho MQTT client. He points out that InfluxDB client libraries are great for reading and writing data in and out.

MOXIE creates buckets in InfluxDB for every account. This makes it easy for MOXIE to provide their clients with a single token which provides them with read access to the bucket. While the data coming through the MQTT stream might change, the unique identifier (UUID) will not. Even if MOXIE adds more sensors or reorganizes the JSON structure, the UUID remains the same. In accordance with InfluxDB's schema, UUIDs are measurements.

Results

Customers are able to access this data via **MoxieWORLD**; the iOS app generates real-time visualizations using the MQTT stream of historical data. The app makes queries to InfluxDB. It is through **MoxieWORLD** that clients are able to access the digital twin of their facility. Clients can generate a digital twin — a layout of their entire facility, to create charts and dashboards, and to export the data as CSV files.

As they want to make it as user-friendly as possible, MOXIE understands their users don't want to see the backend queries but want to understand their devices better within desired time ranges. Through their iOS app, customers can select which IoT device they're interested in, and the time interval, directly through the UI. Clients can select one specific sensor or a group of sensors — for example, they can select all of the sensors tied to a trolley. Clients can control combinations of sensors within the app. All the querying and

filtering is done automatically for their customers based on the sensors and time intervals selected.



Figure 9: Screenshot of MoxieWORLD iOS app

One of MOXIE's customers turned to them to better understand material placement within a facility. This customer uses MoxieWORLD to optimize bridge crane placement based on how far the cranes have to move for unloading and loading. The client can now better place the materials closer to the loading zones for when trucks arrive. This way, they no longer have to move cranes hundreds of yards just to pick up one piece of metal. They have also been able to adjust truck arrival times to balance shift workloads. The bar charts below represent the weekly pace and productivity of the facility. Thanks to MoxieWORLD, the customer determined that the first shift is consistently more active than the other two shifts. This provides them a great opportunity to better optimize operations (preloading, loading, and when the trucks are scheduled to arrive).

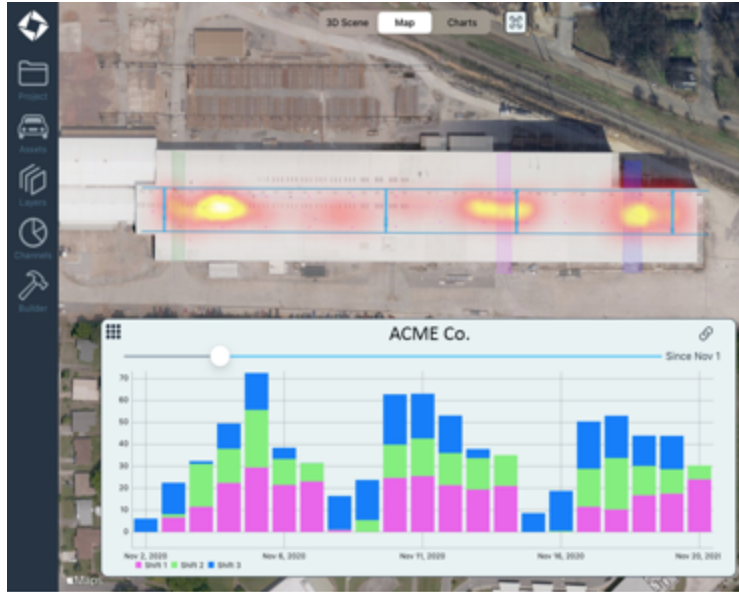


Figure 10: Example of MOXIE's customer - bridge crane at metal distributor

Another client was interested in understanding spindle loads at a production machine shop. MOXIE turned their attention to motor current, electric motor loads, CNC machines, spindles, pumps and compressors. They were able to determine that the spindles were overloaded in the morning as the grease had cooled; running the spindles after the grease had cooled resulted in higher electrical loads on the spindle. The solution was to run the machines for a while longer so they could warm up more before being used. It appeared as though the machines were humming along and working as fast and efficiently as possible; MOXIE helped them improve operational efficiency by reducing air time by 22%.



Figure 11: Example of MOXIE's customer - CNC spindle load at a production machine shop

"MoxieWORLD has made a client's operations 70-80% more efficient."

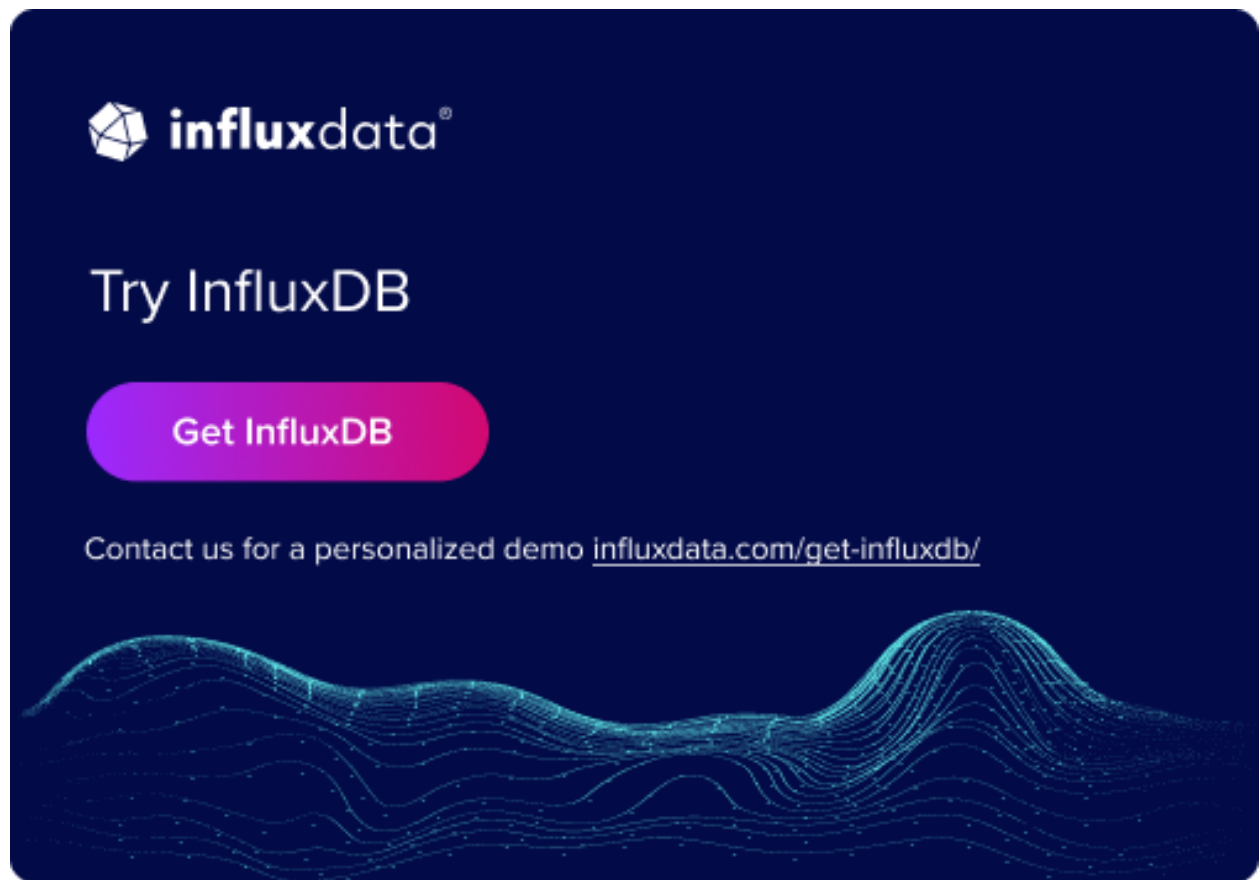
Austin Gurley, Founder and CEO, MOXIE IoT


What's next

In the future, MOXIE IoT would like to set up user-defined alerts in InfluxDB Cloud. This will help improve the usability of their iOS app for their clients. They are also interested in rescaling the data so it's better suited for long-term storage. Gurley points out they are already thinking about the next iterations for MoxieWORLD. For future versions of their solution, MOXIE wants to provide LIDAR scanning for CAD models. They want to extend their ultra-wideband functionality for iPhones. Longer-term, MOXIE would like to provide their customers with cross-facility tracking. For example, they want to enable their clients to track items port to port. Lastly, they want to enable their app to interface with automated guided vehicles (AGV's) and robots.

About InfluxData

InfluxData is the creator of InfluxDB, the leading time series platform. We empower developers and organizations, such as Cisco, IBM, Lego, Siemens, and Tesla, to build transformative IoT, analytics and monitoring applications. Our technology is purpose-built to handle the massive volumes of time-stamped data produced by sensors, applications and computer infrastructure. Easy to start and scale, InfluxDB gives developers time to focus on the features and functionalities that give their apps a competitive edge. InfluxData is headquartered in San Francisco, with a workforce distributed throughout the U.S. and across Europe. For more information, visit influxdata.com and follow us [@InfluxDB](https://twitter.com/InfluxDB).

A promotional banner for InfluxData. It features a dark blue background with a glowing, wavy line pattern at the bottom. The InfluxData logo is in the top left. The text 'Try InfluxDB' is centered. Below it is a pink button with the text 'Get InfluxDB'. At the bottom, it says 'Contact us for a personalized demo' followed by a link.

 **influxdata**[®]

Try InfluxDB

[Get InfluxDB](#)

Contact us for a personalized demo influxdata.com/get-influxdb/

548 Market St. PMB 77953, San Francisco, CA 94104