IoT example

# Business Context

We wish to design a system to manage an arbitrary smart home. A smart home consists of HVAC systems (heating, Ventilation, Air Conditioning) as well as smart appliances, security systems, and other items that are managed by IoT (Internet of Things) devices.

IoT devices use a communication protocol called MQTT (a messaging service protocol) at the application layer. IoT device manufacturers certify their devices using MATTER. MATTER is a connectivity standard for IoT devices. MATTER has a certification process for devices and MATTER certified deices can interoperate securely with different manufacturers.

# Requirements

Use Cases

1. A building manager can add/remove a device to their building.
2. The building manager can send commands to devices in their building.
3. The system can send commands to devices in real time based on data collected at the edge.
4. Device manufacturers can send updates to their devices.

Constraint

1. The system should support Matter certified devices

QAs

1. Availability: The system should ensure that safety critical devices (e.g. fire alarm) are always available.
2. Extensibility: The system should support the addition of new types of devices with a month of development time
3. Scalability: The system should support hundreds of thousands of users in different geographical regions. Each user is estimated to have an average of 10 devices
4. Performance: The system can process sensor input and corresponding commands can be sent to devices within 10 ms.
5. Security: The user should be able to securely connect to their own devices

# Context diagram

Three external types of entities interact with the system: a building manager, the devices being managed by the system, and device manufacturers. The context diagram is shown in Figure 1.

The device manufacturer certifies the device as conforming to the MATTER standard and received certificate attesting to that certification.

Figure 1: Context diagram for envisioned system.

A person in a tie and a computer

Description automatically generated with medium confidence

# Select component to decompose

We begin by choosing the total system. We add a column to Table 1 in method description to annotate the iteration number.

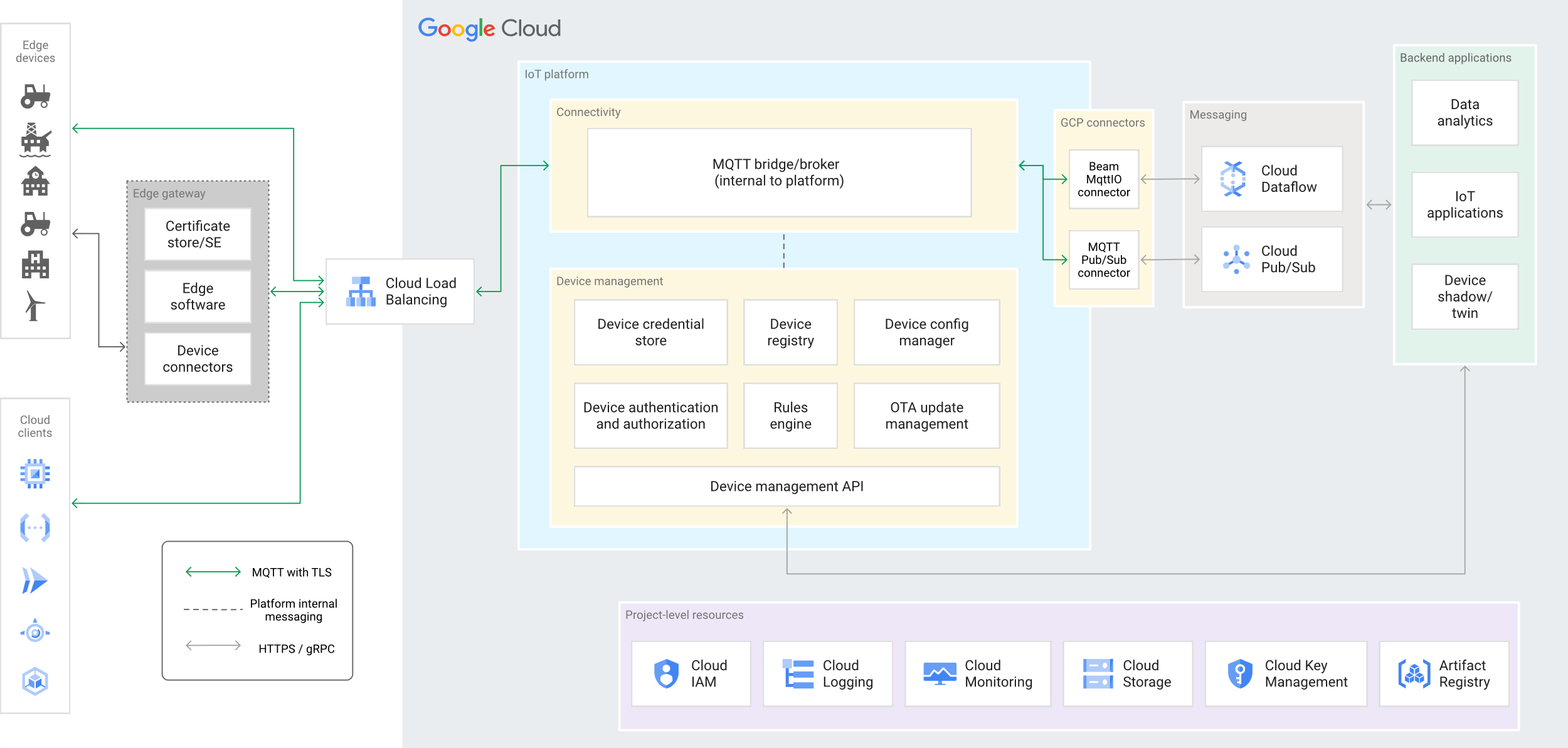
*Example table 1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Iteration number** | **Component** | **Description of activities** | **Quality attribute requirements** | **Components or external entities with which it communicates** | **Allocated to.** |
| 1 | System | Use cases 1-4 | QA 1-4 | Building manager, devices, device manufactures |  |

# Generate decomposition hypothesis.

Initial hypothesis based on reference architecture from Google.

*Figure 1: Itération 0 hypothèsis[[1]](#endnote-1)*



A screenshot of a computer

Description automatically generated

We use the component categories from the reference architecture in this iteration for several reasons:

1. We are not yet ready to commit to the individual Google components.
2. For the initial iteration, we wish to identify missing categories.
3. We can add more detail on subsequent iterations.

Component categories:

* Connectivity
* Device Management
* Project level resources
* Connectors
* Edge gateway
* Messaging

# Analyze Requirement

**Use case 1 for iteration 1: Building manager can add/remove device to their building.**

1. Building manager views available devices and selects the device to add.

**Sequence of responsibilities**

1. Precondition: IoT device is connected to the building network (this is outside of the scope of the system being constructed)
2. Building manager logs into the building management system
3. Building manager asks for a list of connected devices.
4. System generates a list of connected devices with those already registered highlighted.
5. Building manager selects new device and requests that it be registered.
6. System queries device to determine type (using MQTT)
7. System verifies that there is a MATTER certificate for that device type.
8. System adds
   * device to list of managed devices,
   * System adds device to list of devices whose status is periodically obtained and checked against rules.
9. System highlights added device in list seen by building manager

**Actions on responsibilities**:

|  |  |
| --- | --- |
| Responsibility | ACTION |
| Building manager logs into the building management system | Place in unallocated responsibilities |
| Building manager asks for list of connected devices | Place in unallocated responsibilities |
| System generates list of currently registered devices | Place in Device management |
| System generates list of connected but unregistered devices | Place in Device management |
| System displays both lists with registered devices highlighted | Place in unallocated responsibilities |
| Building manager selects new device and asks that it be registered | Place in unallocated responsibilities |
| System queries device to determine type | Place in Device management |
| System verifies that there is a MATTER certificate for that device type | Place in Device management and Edge gateway |
| System adds device to list of managed devices, | Place in Device management |
| System highlights added device in list seen by building manager | Place in unallocated responsibilities |

**Communication needs**

Several communication requirements are identified by this use case

1. System must determine connected but unregistered devices.
2. System must communicate with component that manages list seen by building manager.
3. Device management must communicate with Edge gateway.
4. Edge gateway must communicate with component that interacts with building manager.
5. System must query device

**Added to process table.**

Several items must be determined and should be added to the process table.

1. How does the system determine connected but unregistered devices?
2. What does it mean for a device to be registered?
3. What happens if a communication fails?

**Reflections on use case 1**

This use case demonstrates

* the addition of information to the design (assigning activities and communication to components).
* Identification of unallocated activities.
* Identification of unknowns (added to process table).

Do not attempt to add detail too rapidly. The method allows for the recording of unknown information to be determined later. More detail of the decomposition can be added in subsequent iterations.

**Use case 2 for iteration 1: The building owner can send commands to devices in their building.**

**Sequence of responsibilities**

1. Preconditions:
   1. IoT device is connected to and registered with the building
   2. Building owner is logged into the application with the appropriate access
2. Building owner views a list of registered devices and selects the desired device
3. Building owner views a list of available commands that can be performed on a selected device
4. Building owner sends a command to selected device
5. System verifies that the command can be safely performed on the device
6. System sends the command to the device
7. Device performs action based on the command
8. Device returns result of command to the system, and subsequently to the building owner

**Actions on responsibility**:

|  |  |
| --- | --- |
| Responsibility | ACTION |
| 0a. Ensure IoT device is connected to and registered with the building | Add as responsibility for device management |
| 0b. Ensure building owner is logged into the application with the appropriate access | Place in unallocated responsibilities |
| 1. Building owner views a list of registered devices and selects the desired device | * Add retrieving list of registered devices as responsibility for Device management. * Unallocated – Add the ability for the building owner to view and select devices. (need to create an interface) |
| 2. Building owner views a list of available commands that can be performed on a selected device | * Add retrieving list of available commands as responsibility of Device Config Manager (in Device Management) * How are commands associated with devices? add as process step |
| 3. Building owner sends a command to selected device | * Add responsibility to enable building owner to send commands to the interface added in activity 1 * Unallocated - Add responsibility to receive available commands from device * Determine the format of the “command” - add to process step |
| 4. System verifies that the command can be safely performed on the device | * Unallocated - Add activity of verifying the safety of a command * We need to determine the conditions that are checked for a command to be safe. How are these conditions added, stored, and processed? - add to process step * What do we do if command cannot be safely performed? - add to process steps |
| 5. System sends the command to the device | * Add responsibility to MQTT broker to accept command and send to device * Add responsibility to MQTT pub/sub connector to act as messaging queue for commands to be sent to MQTT broker * Unallocated - Add responsibility of sending command to MQTT pub/sub connector * How does MQTT broker send command to device? - add to process step * What does the system do when the device is offline? - add to process step |
| 6. Device performs action based on the command | * Add responsibility to edge software to receive command from MQTT broker * How does device receive command from MQTT broker? - add to process step |
| 7. Device returns result of command to the system, and subsequently to the building owner | * Add responsibility to device to return result of command to MQTT broker * Add responsibility to MQTT broker to receive command from device and send to pub/sub connector * Add responsibility to pub/sub connector to send results of command to service that will process it * Unallocated - Add activity to receive results of command from pub/sub connector and send to user |

**Added to process table.**

Several unknowns were discovered while analyzing this use case. Let’s add these to the process steps

1. How are commands associated with devices?
2. Determine the format of the “command”
3. We need to determine the conditions that are checked for a command to be safe. How are these conditions added, stored, and processed?
4. What do we do if command cannot be safely performed?
5. How does MQTT broker send command to device?
6. What does the system do when the device is offline?
7. How does device receive command from MQTT broker?

**Reflections on use case 2**

This use case helps to demonstrate 2 important aspects:

**Dealing with unknowns**  
This use case shows us that there may be many unknowns while thinking about the steps required to satisfice a use case. It is important to track the questions that are generated during this process, as they help to inform the overall design of the system.

**Adding new components for unallocated activities**

Sometimes, the unallocated activities in a use case, guide us towards the creation of a new component in the system. This use case has two examples of that.

1. There are multiple unallocated activities that require the building owner to interact with some interface to perform operations. This should inform a design decision to provide such an interface. We do not need to make the decision of whether this will be a mobile app, or website, or simply APIs at this stage. However, it is important to add a new component to represent these discovered responsibilities.   
   We will add a building owner application to represent these responsibilities for future enhancements of the design. The specifics of the application can be determined when we have more information about the system.

*Add component to current design: Building owner application*

1. We have another list of related responsibilities that are unallocated at the end of this use case. These include:
   1. Add responsibility to receive available commands from device (from activity 3)
   2. Add activity of verifying the safety of a command (from activity 4)
   3. Add responsibility of sending command to MQTT pub/sub connector (from activity 5)
   4. Add activity to receive results of command from pub/sub connector and send to user (from activity 7)

Clearly, these are related responsibilities, and we can either add them to an existing component like device management, or create a new component that encapsulates these responsibilities

*Add component to current design: Command Handler*

Once we introduce such new components into the system, it is essential to review, all the other activities in the system to make sure there are no inconsistencies. For example, creating Command Handler might cause us to reassign the responsibility of retrieving list of available commands (from activity 2), to command handler.

In some cases, it is also important to review previously analyzed use cases with this new information. The iteration, help in incorporating some of these refinements into the design.

**Use case 3 for iteration 1: The system can send commands to devices in real time based on data collected at the edge**

**Sequence of activities**

1. Preconditions: The device is constantly streaming data to the system
2. The system processes the data being streamed from the device
3. The system sends command to device if needed

**Actions on activities**:

|  |  |
| --- | --- |
| ACTIVITY | ACTION |
| 0. Ensure device is constantly streaming data to the system | * How can we receive a constant stream of data from the device? - add to process step * How can we detect that a device has stopped streaming data? - add to process step |
| 1. The system processes the data being streamed from the device | * Add responsibility to MQTT broker to publish data stream to device specific topics on the pub/sub connector * Add responsibility to Rules engine to receive data stream from pub/sub connector by subscribing to the device’s topic * Add responsibility to Rules engine to retrieve rules based on device for which command must be sent   + How are these rules stored? - add to process table   + How are these rules represented? - add to process table   + How are these rules added to the system? - add to process table * Add responsibility to Rules engine to process data items based on the retrieved rules and generate command * Add responsibility to Rules engine to re-process failed data points where appropriate. - this creates a detailed workflow that we will handle separately based on priorities |
| 2. The system sends command to device if needed | * Add responsibility to Rules engine to send commands to Command handler * Add responsibility to Command Handler to send command to device. (This responsibility was added in a previous iteration. We do not need to break down the details here) * Are there any special cases when commands are triggered by devices, instead of by building owners? - add to process table |

**Added to process table.**

* How can we receive a constant stream of data from the device?
* How can we detect that a device has stopped streaming data?
* How are these rules stored?
* How are these rules represented?
* How are these rules added to the system?
* Are there any special cases when commands are triggered by devices, instead of by building owners?

**Reflections on use case 3**

This use case had simple looking responsibilities that were required for its satisfaction. However, many components of the system were involved in satisficing this use case, and these need to be carefully thought out while assigning responsibilities to components.

The three primary things that this use case demonstrates are:

* Even if the activities seem simple, oftentimes the complexity lies in how the components of the system can be used to satisfy the use case
* While analyzing responsibilities, we often discover hidden requirements of the system. For example, we discovered that to ensure we are always processing data, we need to store the records to enable retries, and traceability. This can either be delegated to a process step, or detailed while analyzing the use case
* Oftentimes, different use cases have common activities. For example, the activity of sending a command to the device was needed by use case 2, as well as use case 3. We can use our previous analysis, to simplify the design. However, we must think about the specifics, of each use case carefully, while doing this. For example, if a customer generated command fails to reach device, we can simply show the failed status to the customer. This approach is not feasible for use case 3, where a failed command, must be automatically retried.

**Quality attribute requirement 1 for iteration 1: The system should ensure that safety critical devices (e.g. fire alarm) are always available.**

**Key questions for availability:**

* Failure of which components will result in a violation of the requirement?

In this case, failure of a safety critical device will result in violation.

Also, failure of the building management system will result in a violation.

* Is there redundancy in the design for those components?

In a typical safety critical device such as a fire alarm, there is no redundancy. The devices themselves are outside of the scope of the building management system. What the building management system can do is check for operation of the safety critical device.

*Add* *responsibility to device manager*: periodically check operation of safety critical devices and set alarm if a failure is detected.

The other possibility to cause violation of this requirement is a failure of the building management system. Specifically, the device management component.

*Add component to current design:* back up copy of device management component.

*Add responsibility to new component:* synchronize with primary device management data stores.

**Reflections on QA 1 for iteration 1**

Use the key questions for the quality attribute as given in the method description to drive the analysis of whether a design satisfies a quality attribute requirement.

… analysis of other use cases and quality attribute requirements is not included in this example…

# Prepare for next iteration

As a result of the analysis thus far, there are responsibilities in the unassigned responsibility table. Examination of these responsibilities results in the addition of a new component to the design: edge component. It will communicate with the edge gateway and be allocated on an edge device. Its responsibilities will be:

* Building manager logs into the building management system
* Building manager asks for list of connected devices
* Display registered devices and unregistered devices with registered devices highlighted
* Building manager selects new device and asks that it be registered.
* System highlights added device in list seen by building manager.

All other responsibilities discovered by the use case analysis are accomplished by Device Management.

**Process table**

The process table contains items of different types.

* **Items requiring research.** The item “How does the system determine connected but unregistered devices?” can be resolved by doing research into the operating system capabilities.
* **Items requiring design.** The item “What does it mean for a device to be registered?” will require a deeper understanding of how the system treats registered devices. The device will have a number of settings defined by MATTER with recommended values provided by the manufacturer. The settings for the devices in the building must be determined. There are also activities that will be performed on registered devices such as health checks and status checks. Decisions must be made about how these settings are performed and the results stored and made available. Registered devices are managed by the Device management component. So these decisions will be deferred until the Device Management component is decomposed in the next iteration.
* **Items requiring additional activities**. The item “What happens if a communication fails?” will require retries, possibly marking a device as failed. The activity “manage failure of device communication” is assigned to the Device Management component.

# Begin iteration 2

With the analysis we have performed in this example, only two components have lists of activities: Device Management and Edge. The use cases and the quality attribute requirements are unchanged. Choose, for example, Device Management and repeat the process of generating a hypothesis and analyzing the hypothesis.

1. https://cloud.google.com/architecture/connected-devices/iot-platform-product-architecture [↑](#endnote-ref-1)