***Smart Home Assistant***

Contents

[1. Introduction 3](#_Toc536130860)

[2. Context 3](#_Toc536130861)

[3. Functional Overview 4](#_Toc536130862)

[4. Quality Attributes 5](#_Toc536130863)

[5. Constraints 5](#_Toc536130864)

[6. Principles 6](#_Toc536130865)

[6.1 Layering 6](#_Toc536130866)

[6.2 Componentization and Containerization 6](#_Toc536130867)

[6.3 Protocols and Message Formats 6](#_Toc536130868)

[6.4 Business Rules 7](#_Toc536130869)

[6.5 General Development 7](#_Toc536130870)

[7. Software Architecture 7](#_Toc536130871)

[7.1 Application Layer 7](#_Toc536130872)

[7.2 Data & Analytics Layer 8](#_Toc536130873)

[7.3 Edge & Connectivity Layer 8](#_Toc536130874)

[8. Smart Home Assistant 8](#_Toc536130875)

[8.1 Business Rules and Application Services Management 8](#_Toc536130876)

[8.2 Rules Inference and Execution 9](#_Toc536130877)

[9. Smart Home Assistant Platform 11](#_Toc536130878)

[10. Improvement Opportunities 12](#_Toc536130879)

[11. Scenarios used 12](#_Toc536130880)

[12. Next Steps 12](#_Toc536130881)

# Introduction

This document attempts to offer a high level overview and explains the proposed architecture for the “Smart Home Assistant” system.

It describes only a provisional system considering few initial imaginary scenarios. Thus, it can serve as a tool to drive a purpose-driven discussion that helps to evolve the concept further during discussion.

# Context

A human user uses many home appliances (e.g. air-conditioner, geyser, entertainment systems) and personal devices (e.g. smartphone, tablet, wearable, personal computer). Many times, the user observes patterns in using these devices to fulfill daily routines in life, like the sequence in which certain devices get used or the way in which the behaviors of these devices get combined and correlated.

The user wants to improve quality of life being away from such mundane tasks by orchestrating repeatedly occurring patterns of sequences (workflows) & correlated behaviors and automating them. For example:

1. The clock communicates with the watering system to water the garden daily at a regular time
2. The personal calendar communicates with the local weather network and the combination of these two information can predict the chances of rain interfering an outdoor party scheduled two days away

Home Appliance / Device

User

1. Configure device

2. Registration

4. Sensor Data & Status Data

4. Actuator Commands

Smartphone / Tablet / PC

1. Configure device

5. Statuses

3. Configure correlations

1. Configure device

3. Configure correlations

6. Dashboard – Visualize & Interact

Smart Home Assistant

Smart Home Assistant System

Smart Home Assistant (SHA) system aims to offer a platform for home appliances and personal devices within home environment to connect, interact and collaborate with each other to offer a unified and connected experience for humans, e.g. automation, entertainment, wellness and etc. It offers ways to:

1. discover and configure devices within home environment
2. configure connection and interaction between devices
3. configure correlations in capabilities and behaviors as offered by devices
4. orchestrate and automate required workflows
5. observe and monitor the status of complete activities in home environment
6. visualize trends and patterns for any further inferences and automation opportunities

# Functional Overview

In this system, SHA acts as a central hub. It allows all devices and vendor services to interconnect with each other in automating the intended workflows.

1. Thing – every device / appliance in home environment is referred to as a “Physical Thing”. It consists of:
   1. Sensor – gathers information, converts it into digital format and sends it to IoT module
   2. Actuator – receives commands from IoT module, converts it into analog physical action
   3. IoT module – collects information from associated sensors, processes it and sends appropriate commands back to associated actuators
2. IoT module – an IoT module is essentially a microcontroller. It consists of:
   1. Microprocessor core – it processes local data and makes appropriate decisions
   2. Memory
      1. ROM – to store the program for microprocessor
      2. RAM – to receive and store data including data for microprocessor processing
   3. Input ports – to collect data from sensors
   4. Output ports – to effect necessary actuations
   5. Controller interface – to communicate with other Things in the system

Smart Home Assistant

***IoT***

Mobile Device

Advanced End-user

End-user

Mobile Device

Vendor Services

Thing

Actuator Commands

Sensor & Status Data

Service Registration

Event & Data Stream

Configure Device & Register Service

Configure Rules

Configure Device

Register Device & Service

Configure Rules

Trends & Analytics

Visualize & Interact

1. Vendor Services – vendor services available locally or on the internet can provide interesting information necessary to fulfill the intended workflow for the user
2. Advanced End-user – configures devices and vendor services with all settings configuration, authentication and authorization details necessary to connect and exchange information with them
3. Advanced End-user – knows intimate details about:
   1. what external events and information come from various IoT sensors and vendor services
   2. how to correlate these events to extract sensible interesting patterns out of them
   3. how to process these correlations and patterns to produce resulting outcomes
   4. how to send these resulting outcomes as commands back to IoT actuators
   5. how to orchestrate intended workflows out of these event triggers, correlations and actions
4. Advanced End-user – orchestrates intended workflows in the form of “Rules”. A rule consists of:
   1. Trigger – when an external event or information is received and it satisfies the necessary correlation or pattern, the rule gets executed
   2. Action – when a rule gets executed, it processes received information and produces outcomes
5. SHA – it has the ability to:
   1. continuously receive streams of events and information from all sensors and services around
   2. continuously evaluate the conditions on incoming events
   3. automatically trigger appropriate rules when necessary conditions are satisfied
   4. appropriately execute actions to process received information and to produce results
   5. automatically pass the results as commands back to actuators

Since the SHA acts as a central hub, everything flows through and happens in it.

# Quality Attributes

SHA needs to fulfill the following quality expectations from the user to offer a flawless and peaceful experience:

* Availability – system should be available 24x7 to take care of the automated activities. A small amount of downtime (less than 30 minutes per day) can be tolerated.
* Security – system should be available only to authorized users reaching through home network. Only “Advanced End-user” are allowed to create or modify the business rules inside SHA.
* Scalability – system should support exchange of data between all certified compatible devices within home network. It is expected to support exchange of 5000 messages per minute now and it is expected to increase 50%. The impact when number of devices increase within home network is expected to be negligible.
* Reliability – the data traffic from all participating devices should be 100% reliably communicated
* Auditing and Backup – audit logs for actions performed within system are categorized in terms of levels of importance. “High” level logs are for statutory purposes and must be kept for a duration of one year; “Medium” level logs are for product support purposes and should be kept for a duration of one week; “Low” level logs are for product diagnostic purposes and could be rolled over on need. Appropriate alerts should be raised in advance for the user to backup “High” level logs when storage is running out.
* Usability – user should have a uniform and unified experience across products

Any other attributes can be evaluated periodically and decided based on more understanding of requirements.

# Constraints

Following constraints are considered in developing this system:

* Hardware – this system concerns only with the software and communication aspects; it does not deal with any hardware, chip, embedded software design or firmware and their updates
* Connectivity – connectivity options like Wi-Fi, Bluetooth, ZigBee, Thread, Cellular, Gateway and etc. are out-of-scope for this system because it depends on required business factors like range, size and budget.
* Time, Budget and Resources shall be discussed based on more information about requirements and will have a significant influence
* Tentative skill profile of software development team based on present understanding on requirements
  + Hardware interfacing developer – knows how sensors and actuators work
  + Networking developer – knows communication protocols, connectivity solutions and messaging models
  + Application designer and developer – knows products and applications development including UI
  + Business Intelligence and Data Analytics developer – knows how to reliably and quickly ingest, store, process and query vast amount of heterogeneous data originating from devices
  + Security developer – knows authentication models, how to securely exchange sensitive data with right encryption models including latest trends like blockchain
  + Machine Learning and AI developer – knows how to infer new patterns emerging from the data collected so far and to take these inferences further to realize new automation opportunities

# Principles

The system would generally be developed considering the principles and guidelines outlined here.

## Layering

Every piece of code should honor the layering abstraction:

* Adherence to “Dependency Inversion Principle”:
  + High level layer depends on low level layer; low level layer does not depend on high level layer
  + High level layer consumes the behavior of low level layer through a well-defined interface; the interface definition is owned by high level layer
* Release management
  + A layer can evolve independently from another and thus has its own individual lifecycle
  + Broken changes on interfaces should make it first obsolete, then deprecate and eventually remove.
* Testability of layers
  + Every exposed behavior from a certain layer to another layer is 100% unit tested
  + A layer is tested within its own boundary by mocking necessary behaviors of low lever layer’s interface

## Componentization and Containerization

When a set of code offers related services and capabilities, it should be packaged as a service (or component):

* Adherence to microservice architectural style:
  + A service adheres to Single Responsibility Principle catering to only one responsibility
  + A service is an individually deployable unit and can evolve independently from another
  + A service exposes its behaviors through well-defined interfaces
  + A service is stateless in its communication with other services
* Testability of components:
  + Every behavior exposed from a component through interface is 100% unit tested

## Protocols and Message Formats

* Use standard IoT specific lightweight protocols like MQTT and CoAP for communication with edge devices.
* Use standardized efficient messaging serialization formats like JSON, Protocol Buffers and Flat Buffers for exchange of data across microservice and system boundaries.
* Use APIs with agreed interfaces for communication between web services and application clients.

## Business Rules

* When writing business rules, actively look for any inadvertently resultant system-wide conflicts which can potentially result in system-wide buggy behaviors, for e.g.
  + two rules with conflicting trigger conditions but with same action
  + two rules with same trigger condition but with conflicting actions
  + recursion resulting from dependencies between rules, including from transitive dependencies

## General Development

* Preferably use a rich domain model
* Do not write business logic in views and do not access database from views
* Preferably use a system-wide Dependency Injection and IoC container
* Preferably use a system-wide unified logging framework

# Software Architecture



## Application Layer

This is responsible for defining and managing required workflows using application focused microservices.

### Rule Engine Management

A flexible rule engine that allows user to configure and modify business rules. Rules try to externalize the business logic through “constraints” (event-trigger-conditions) and “actions”.

### Application Services Management

It allows user to import business logic modules. These modules are typically web services running inside SHA system that register to the vendor services and subscribe to their event / data streams.

### API Management

It defines APIs for user to manage devices, workflow rules and other business logic modules, to receive data analytics and associated visualizations.

### User interface framework

This framework helps in providing uniform experience for the user in interaction and visualization abilities with the system. This framework is commonly used across the whole application.

This framework defines necessary user interfaces like controls, widgets, charts, reports, icons and etc.

## Data & Analytics Layer

This is responsible for gathering, processing, storing and serving data.

### Data Ingestion

* Listen and capture message stream from devices
* Type system for messages – determine type of each message and forward for further processing

### Data Processing

* Performing quantitative analysis like categorizing and aggregating data
* Performing qualitative analytics like correlation, statistical analysis, trend analysis
* Parallel computing to boost performance of calculations and query response

### Data Models and Storage

* Support for multiple data models – relational and document
* Management of data relationships – dependencies and contracts
* Version management – mechanisms to handle obsolete and deprecated interfaces
* Scaling and Retention – scaling of storage with low-cost storage mechanisms and needs-driven retention

## Edge & Connectivity Layer

This is responsible for managing devices and their integration.

### Edge Device Management

* Device discovery and resolution
* Device configuration, connection and availability

### Security Management

* Security at edge level with authentication or security token mechanisms

### Connection Management

* Edge devices offer multiple ways of data connections based on individual hardware architectures, for e.g. always-connected stream data, periodically-connected batch data

### Message Formats and Protocols

* Support for message encodings (text, binary, json, protocol buffers), communication protocols (MQTT, CoAP) and their messaging models like publish/subscribe, request/response, multicasting

# Smart Home Assistant

This section explains two majorly important use cases of the SHA system.

## Business Rules and Application Services Management

The system allows the user to configure and modify business rules. The user can perform these activities from a laptop by launching the web application running inside SHA.

The system allows the user to import an application service (business logic) that subscribes to a vendor service and acts as a “Sensor” data provider for the system. The user can perform this activity from a laptop by deploying a web service directly into SHA.

The system allows the user to start and stop the rules for an interim period as necessary. The user can perform these activities from a laptop or mobile device by launching the web or mobile application running inside SHA.

Smart Home Assistant

Advanced End-user

**Web Application**

[Container: .Net MVC]

Delivers the static content and SHA single page application

**Web Service**

[Container: .Net WCF]

Business logic to register with and receive stream of data from vendor service

**Single Page Application**

[Container: JavaScript and Angular]

Provides all functionalities of SHA business rules management to customers via their web browser

**Mobile App**

[Container: Xamarin]

Provides Start Stop functionality of SHA business rules management to customers via their mobile device

Delivers

Uses

[HTTPS]

Uses

Uses

Deploys

**Database**

[Container: Relational Database Schema]

Stores user authorization, access logs, audit logs, etc.

**API Application**

[Container: .Net MVC]

Provides SHA business rules management functionality via JSON/HTTPS API

Reads from and Writes to

[ODBC]

**Database**

[Container: Document Database Schema]

Stores business rules in form of JSON documents

Vendor Services

Subscribes to event data

Receives event data

Reads from and Writes to

[ODBC]

Uses

[HTTPS]

Uses

[HTTPS]

## Rules Inference and Execution

The system allows the devices and SHA to exchange messages between them through listeners and publishers:

* The listeners and publishers are essentially “bindings” necessary to establish communication between two entities. A binding specifies following details: transport/protocol (e.g. MQTT, CoAP, UDP, TCP) and message encoding (e.g. text, binary).
* A transport/protocol supports one or more messaging communication model, for e.g. publish/subscribe and request/response.

When an event comes from a device or service or from another rule, the rule engine applies trigger-conditions of business rules onto the event. It executes the actions of all rules whose conditions are satisfied.

Smart Home Assistant

Thing

***IoT***

**Rule Engine**

[Multithreaded Inference Engine]

**Database**

[Container: Document Database Schema]

Stores business rules in form of JSON documents

Reads from [ODBC]

**Listener**

[MQTT, CoAP, UDP, TCP]

[Publish/Subscribe, Request/Response]

**Publisher**

[MQTT, CoAP, UDP, TCP]

[Publish/Subscribe, Request/Response]

**Rule**

[Written in JSON format]

**Trigger-Condition**

[Conditional, Logical operators]

**Action**

[Sequence of operations]

{Performs Arithmetic, Comparison, Logical, Temporal calculations}

{Measures correlations, statistics, trends}

Sends operation results to Publishers and Stores

**Database Store**

[Container: Relational or Document Database Schema]

Stores events from sensors and results from rules

**Memory Store**

[Queue, Blackboard]

**Log Store**

Inbound event message

Verifies trigger-conditions on messages

Executes action when trigger-conditions satisfied

**Cloud Store**

Result

Event messages

{Sensor Data, KeepAlive, Sync-request, etc.}

Thing

***IoT***

Event messages

{Actuator Command, Heartbeat, Sync-response, etc.}

There are many streaming and stream processing platforms available off-the-shelf like [Apache Kafka](https://kafka.apache.org/) and [Reactor](http://anywhichway.github.io/rule-reactor.html). It might be an over-engineered design to use for home automation needs based on cost constraints.

We can alternatively write a custom efficient and lean rule engine that works based on the [RETE](https://en.wikipedia.org/wiki/Rete_algorithm) algorithm. It decouples evaluation of trigger conditions (hypothesis) and execution of actions (sequencing and ordering). Thus, it can apply all trigger conditions onto the incoming event data simultaneously to verify which trigger conditions are satisfied. Once it knows the rules that satisfied the trigger conditions, it can simultaneously trigger the associated actions.

Any workflow can be captured using this “rule template”: {Listener, Trigger-Condition, Action, Publisher, Store}.

# Smart Home Assistant Platform

The platform includes both directly consumable components and extensible behaviors for many modules mentioned in the architecture. The application developers can write missing components and extend the behaviors to fulfill the needs of the applications they are developing.

1. Application Layer
   1. Rules management
      * The platform includes a rule engine that expects the workflow to be captured in terms of a much generic “rule template”.
      * An application developer needs to just focus on only capturing the business logic in terms of: trigger-conditions, domain calculations and necessary communication bindings.
      * *Bindings are ready-to-use components and rules are extension points offered by platform*.
   2. Application services management
      * The platform allows to deploy a web service that encapsulates the required business logic.
      * The sophisticated smart IDEs available nowadays capture away the boilerplate code sequences in writing web services and allow the application developer to focus only on capturing business logic.
      * *Web service management is an extension point offered by platform*.
   3. User interface framework
      * The platform comes with a UI framework that already adheres to the UX guidelines designed to offer a unified intuitive feel for the user to visualize and interact with data.
      * An application developer needs to just focus on what kind of visualization is necessary to show the data and what kind of interaction is necessary for the scenario
      * *Framework offers extension points for the application developer to supply data for visualization and to customize interaction behavior as necessary for the intended workflow*.
2. Data & Analytics Layer
   1. Data ingestion
      * The platform makes use of “Reactive Programming” model to capture the abstraction of “message streams”.
      * *An application developer just needs to adhere to this programming model in code to receive/send streams of messages*.
   2. Data processing
      * The platform comes with a rich library of data processing operators to perform both quantitative and qualitative analysis.
      * *An application developer can use these operators in combination when writing the business rules to perform required operations on inbound data*.
   3. Data models and storage
      * The platform supports polyglot persistence – relational and document – since the data from IoT devices can come in many forms: structured, semi-structured and unstructured.
      * The key-value and graph data models can be easily supported in future, when necessary.
3. Edge & Connectivity Layer
   1. Communication protocols and messaging models
      * All boilerplate code sequences necessary for establishing communication, handling communication protocols and communication models are already supported by platform.
      * *An application developer can directly use them when writing the business rules to establish communication with necessary endpoints*.

# Improvement Opportunities

There are many improvement opportunities available on the proposed architecture. These can be evaluated and the right options can be chosen, once we know the system requirements and constraints more deeply.

* There are many latest developments in network connectivity solutions and corresponding communication protocols developed just to support the exclusive demands of IoT applications, e.g. ZigBee, Thread, Z-Wave.
* There are multiple disparate ways to discover the devices in home network and to authenticate the communication between them in a secured way.

# Scenarios used

Few sample user scenarios considered in proposing this architecture:

* The clock communicates with the watering system to water the garden daily at a regular time
* The personal calendar communicates with the local weather network and the combination of these two information can predict the chances of rain interfering an outdoor party scheduled two days away
* The air-conditioner stops cooling the room when a person wakes up in morning and this further triggers the geyser to start its thermostat in anticipation of the person taking bath 20 minutes later

# Next Steps

The immediate next step would be to discuss and improve the direction or gaps as identified during discussion and to evolve the architectural concept accordingly. We can further develop a proof-of-concept.

The design for only two majorly significant technical use-cases are shown in this document. Further important use-cases like calculation of trends and visualizing them on mobile device are omitted in the interest of time.