SMART BUILDING MANAGEMENT PLATFORM

1/1/2016	1.0	Muhammad Zahalqa
2/4/2016	1.1	Muhammad Zahalqa
26/6/2016	1.2	Muhammad Zahalqa
15/11/2016	1.3	Muhammad Zahalqa

1 Scope

SmartBuilding is a next generation platform for managing IoT enhanced smart building. The platform serves as the controller and integration hub for modules, devices, controllers and sensors that will enable efficient and comfortable operation of the building.

1.1 Scope of Project

The scope of the project is as following:

a. Provide a high level design
b. Provide a detailed level design
c. Provide an implementation using C++ for a POSIX compliant system.

2 Introduction

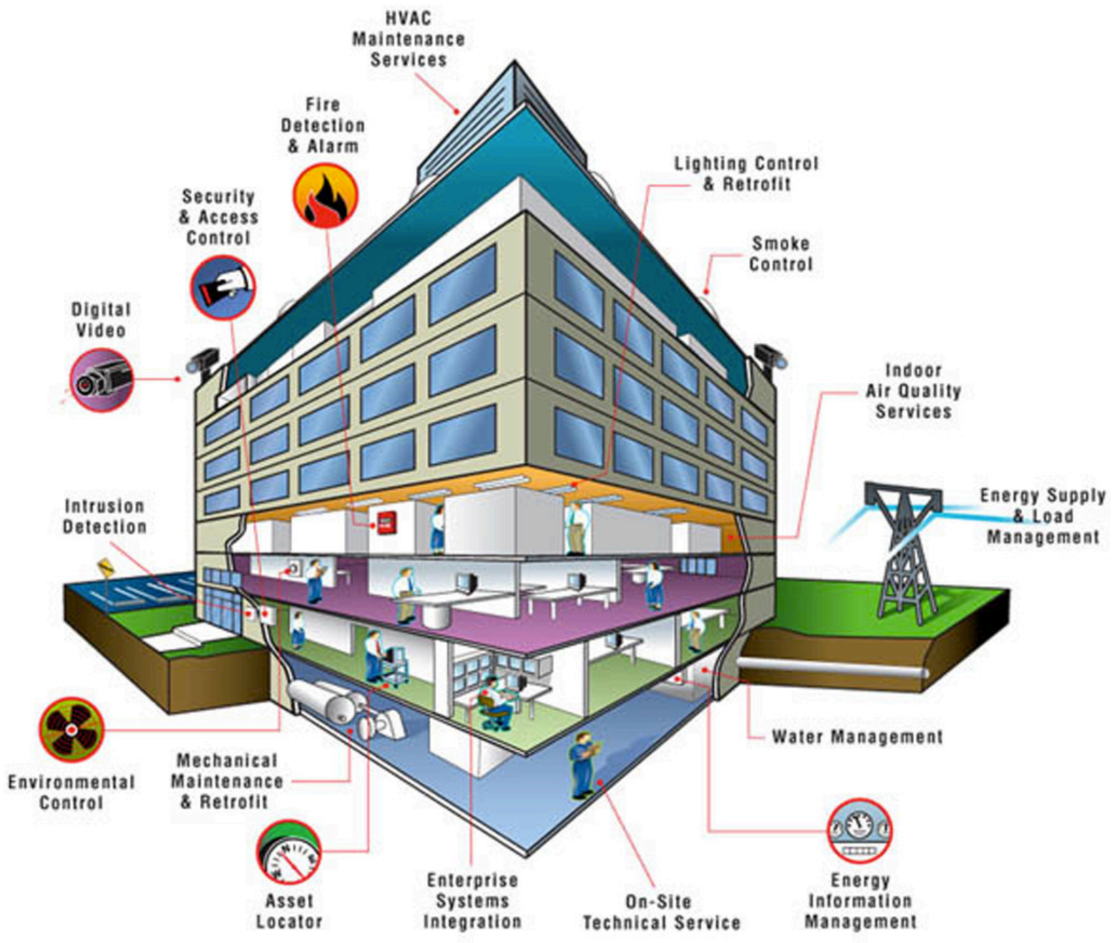
2.1 What makes a building smart?

The ability to collect data from environmental sensors such as HVAC, lighting, fire safety and other sources combined with the ability to extract, aggregate, analyze, and make decisions based on that data.

New technologies, mobile devices, the Internet of Things (IoT), and big data are rapidly creating a world with levels of connectivity and access to data never seen before. The intelligence derived from such data can bring unprecedented visibility and optimization to building operations

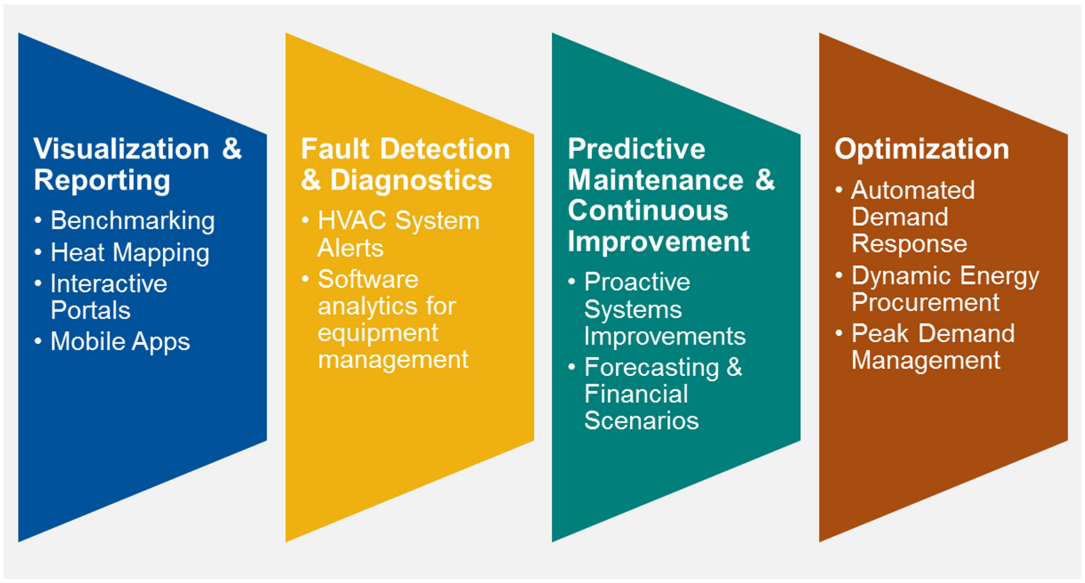
The main areas of improvement are:

a. Energy efficiency
b. Operational efficiency
c. Occupant comfort


HVAC Maintenance Services
Fire Detection & Alarm
Lighting Control & Retrofit
Security & Access Control
Smoke Control
Digital Video
Indoor Air Quality Services
Energy Supply & Load Management
Intrusion Detection
Water Management
Environmental Control
Mechanical Maintenance & Retrofit
Asset Locator
Enterprise Systems Integration
On-Site Technical Service
Energy Information Management

2.2 Benefits

The platform offers more than one benefit:


Visualization & Reporting
• Benchmarking
• Heat Mapping
• Interactive Portals
• Mobile Apps
Fault Detection & Diagnostics
• HVAC System Alerts
• Software analytics for equipment management
Predictive Maintenance & Continuous Improvement
• Proactive Systems Improvements
• Forecasting & Financial Scenarios
Optimization
• Automated Demand Response
• Dynamic Energy Procurement
• Peak Demand Management

2.2.1 Optimization

a. Access actionable analytics and reporting
b. Improve operations
c. Reduce OPEX costs

2.2.2 Maintenance

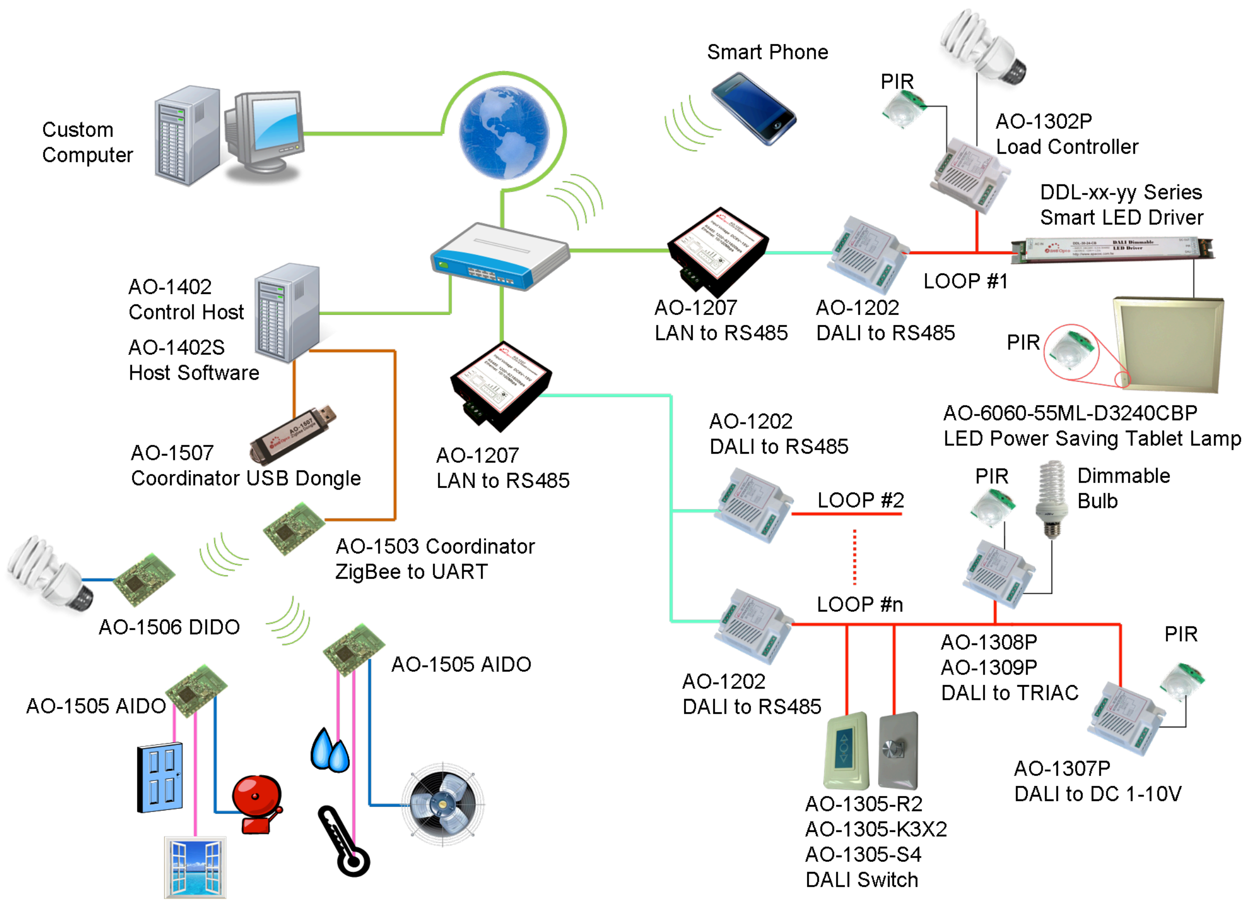
a. Preventative maintenance
b. Speed problem resolution
c. Improve equipment management
d. Increase occupant satisfaction

2.2.3 Reporting and Alerting

a. Mobile and web Alerts
b. Analytics and Data Reports

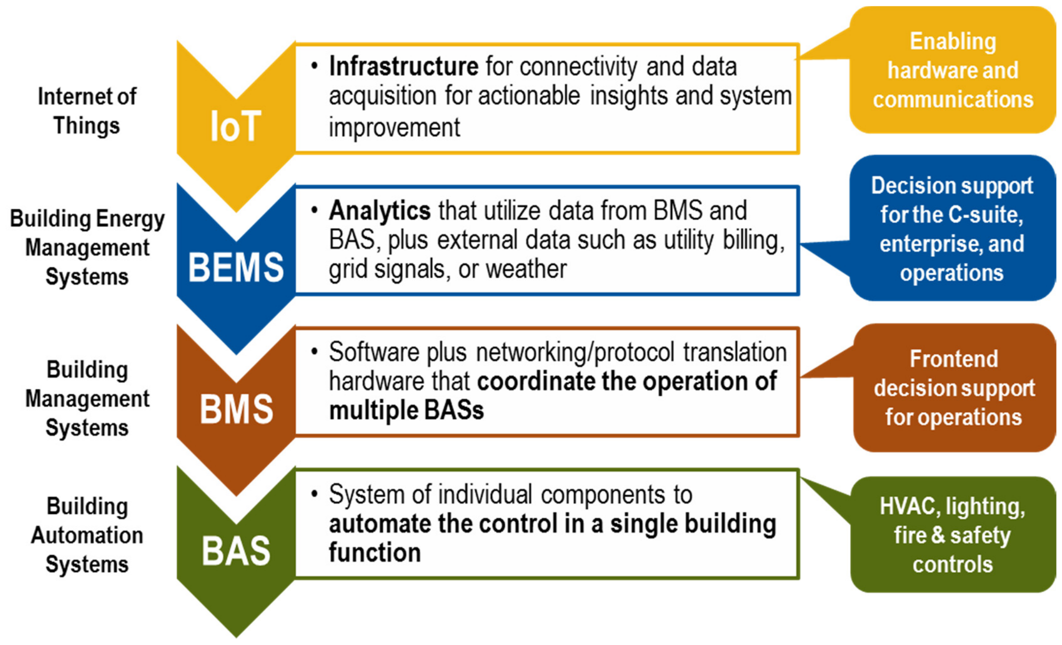
2.3 Sample installation

Sample installation using of the shelf components.



3 High Level Requirements

The project will implement the BAS layer (see figure bellow) of the smart building system. The focus of the BAS layer is to provide a server application that act as a central hub to mediate between software agents responsible for communication with the actual hardware device.



Internet of Things
IoT
• Infrastructure for connectivity and data acquisition for actionable insights and system improvement
Enabling hardware and communications
Building Energy Management Systems
BEMS
• Analytics that utilize data from BMS and BAS, plus external data such as utility billing, grid signals, or weather
Decision support for the C-suite, enterprise, and operations
Building Management Systems
BMS
• Software plus networking/protocol translation hardware that coordinate the operation of multiple BASs
Frontend decision support for operations
Building Automation Systems
BAS
• System of individual components to automate the control in a single building function
HVAC, lighting, fire & safety controls

Specifically, the platforms has the following components:

3.1 Management and Central Hub

The central hub is a server application that serves mainly as a host for software agents representing the devices. It will provide an efficient mechanism to receive events from sensors and dispatch these events to relevant controllers.

3.2 Events

The system will enable devices to generate and send custom events. Events will be generated by sensor agents and will be consumed by controller agents.
An event will consist of a type/topic, location info and arbitrary agent event specific payload.

3.3 Devices

The system will enable an operator to configure and install software agents that mediate communication with external hardware devices.

These devices can act as:

- Sensor
  A device that monitors and measures a physical phenomena.
  A sensor will publish the result of measurement/sensing as an event to the system.

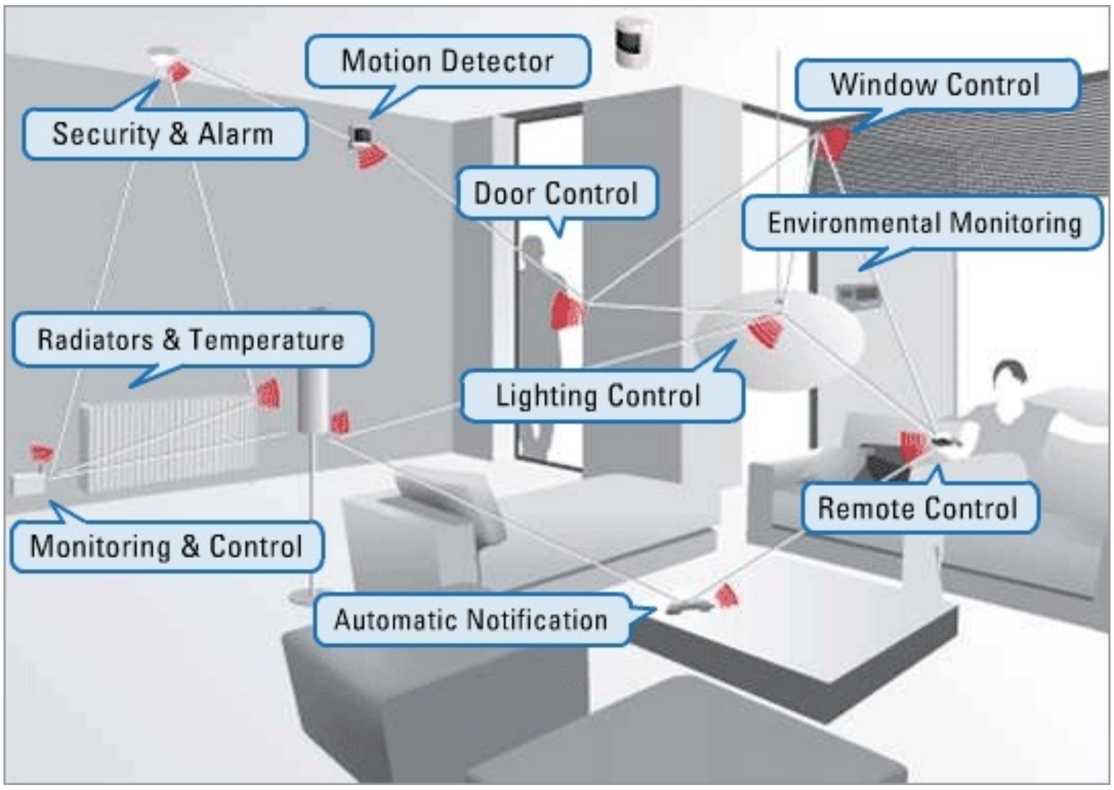
- Controller
  A device that will execute an operation. Usually affecting the environment. Examples include: Sprinkler, Alarm and automatic locks. Controllers will act upon the receipt of an alert/event from the system.

3.4 Connected Alerts

Setup connected alerts, to notify Web-based monitoring systems of unusual activity, such as unauthorized movement through the building or exceeded utilities usage.
NOT PART OF VERSION 1.0

4 System Description

Smart Building platform will consists of a server that manages a set of sensors and controllers as specified by a configuration file. It will route events generated by the sensors to the controllers interested to act upon these events.



4.1 Server

Smart home server will be the central piece of software that manages all the devices via software agents. It's the central point responsible for:

- Loading and Configuring the device agents
- Enable sensors to publish events
- Enable controllers to receive events of interest
- Enable a user to view a live log of event flow

Basically, on startup the server will read a configuration file describing what devices are installed in the building. This configuration will enable the server to create the agents objects representing the devices and initialize those agent objects.

It will also manage the events distribution within the system. It will enable a sensor to publish an event and a controller to consume a specific event.

4.2 Sensors & Controllers

Sensors and controllers are devices connected to the building network and communicate with the server via an agent object loaded by the server.

Example of possible agents:

a. Lights Controller – control hue and intensity of light bulb.
b. HVAC – control operation of HVAC system.
c. Smoke detectors
d. Fire safety – sprinklers
e. Emergency lightning
f. Elevators - minimize movement when unoccupied
g. Doors and Locks
h. ZigBee devices (Coordinators and End devices)

4.2.1 Agents

An agent is the software entity loaded by the server to represent an actual device and will allow the basic interaction between server and devices.
Agents will be loaded at system startup according to a configuration file.

Each agent is implemented as a dynamic object (.SO) and will implement and expose the required interfaces in order to communicate with the server.
These interfaces should be defined by an SDK allowing 3rd parties to implement the various add-on sensors and controllers.

Most of the agents will need to communicate with the respective sensor/driver using a network protocol. But this should be transparent to the server.
An agent can implement whatever IPC mechanism it wants to actually communicate with the hardware device it represents or control.

4.3 Events

Sensors are devices that will usually monitor an environmental variable and send events carrying information describing the event.

Sensors will generate events and publish them to the server, which will route these events and deliver them to devices interested in receiving those specific events.

Controllers will usually register their interest in such events. They will subscribe to specific event types originating in specific locations: specific room/rooms and/or specific floor/floors. The specifics of these subscriptions are either provided in the controller configuration (see later 4.5.1) or hardcoded into the controller.

4.3.1 Event details

Generated events will contain at least the following:

- Event timestamp
- Event type
    - The type is defined by the sensor it's self or configurable.
- Event data payload

    - Payload is opaque data generated by the sensor and understood by relevant controllers who will subscribe to this event type.
    - Examples:
        - Motion Detected event has no additional data
        - Temperature change event has additional data:
          double number representing current temperature
          Boolean flag indication if the temperature reading is rising or declining
- Event location
    - Where the event was detected. The location is determined by the location of the sensor publishing the event (building, floor, room)

Example 1:
Suppose we have a sensor for ambient noise and a controller that controls a noise cancelation device. The controller will subscribe to receive events of type NOISE_LEVEL in the specific location of interest. The sensor will send an event of type NOISE_LEVEL with additional data indicating noise level measured in dB(A). The system will route the event to the controller who will act according to the event's dB(A) value.

Example 2:

- A badge sensor will read and authenticate an employee identification badge. Sensor will then send an event of type ENTRANCE_REQUEST with payload data specifying employee ID, sensor ID, Lock ID to be unlocked and a secure signature identifying the badge.
- A security controller subscribing to this specific event will receive the event, validate the secure signature to make sure it's a legitimate badge for the given employee and then will decide if the a door should be unlocked to allow entrance.
- Security Controller will send an UNLOCK_ENTRANCE event with payload data containing lock id to be unlocked

4.4 Smart Building Configuration Map

The configuration file will list the details of devices present in the building. This map will serve to create the devices and manage the topology of these devices.

The configuration file is normal INI file with one section for every device. This section will have the following format (order is not strict):

> **[** *devic-id a unique device instance identification string* **]**
> ***type*** = *device-type*
> ***room*** = room-identification
> ***floor*** = floor-designation
> ***log*** = log name to be used by the device. (zlog logger)
> ***config*** = a device specific configuration string

The fields are:

- Device Id
  A string representing a unique id for the device in the system
- Type
  Mandatory filed, specify the type and used to create the actual instance
- Room & Floor
  Mandatory fields. Specify the location of the device. Passed to device object so it knows its location
- Log
  Optional field, passed to device agent to use as a log destination. If not configured pass a default log.
- Configuration
  Optional Configuration data passed to the device agent. The actual device type decides the format of this string. It can vary between agent types.

4.4.1 Example Configuration

```
[ light-1-a-1 ]
type = PhilipsHue
room = room_1_a
floor = 1
log = lights
config = iot:1003,sat:242, brigh:154, hue:8200

[ light-1-a-2 ]
type = PhilipsHue
room = room_1_a
floor = 1
log = lights
config =  iot:1008,sat:242, brigh:154, hue:8200

[badge-reader-4-g]
type = IDentCard
room = entrance_4_g
floor = 4
log = security
config = {Token: “78DF6A2BA25DA87”, Event: “badge id”}

[door-4-a]
type = ZigBeeLock
room = room_4_a
floor = 4
log = security
config = {Token: “78DF6A2BA25DA87”, Event: “badge id”, From: “entrance_4_g”}

[room-1-a-smoke]
type = smoke_detector
```

```
room = room_1_a
floor = 1
log = safety

[room-1-a-sprinkler]
type = sprinx
room = room_1_a
floor = 1
log = safety
```

4.5 Interaction between server and agents

Server will initialize the agents and provide each agent with: id, location, log and its configuration string as it appears in the configuration field.

4.5.1 Initializing Agents

Server will initialize the agents and provide each agent with: id, location, log and its configuration string as it appears in the configuration field.

4.5.2 Sending Events

Agent will be able to send events to the server for publishing to interested subscribers. The server will also enable communications between the sensors and controller components using a publish/subscribe model.

4.5.3 Receiving Events

Agent will be able to subscribe to certain events at the level of one or more rooms or one or more floors.

5 SDK

The system will provide a documented SDK that enables third parties to develop software agents for sensors and controllers.

The SDK will provide:

- Documentation
- API in the form of CPP header files and optionally libs to be used by 3rd party agents.
- A sample of 3-4 agent implementations for educational purposes.

5.1 SDK APIs

The server should expose a programmatic API and a specification to enable building 3rd party sensors and controllers.

5.1.1 Server API

The server exposes to the device agent an API. The device agent will use this API to consume services like publishing events to the system.

The API should enable the agent to:

- Publish events to the server
- Subscribe to receive events of interest

5.1.2 Device API

The device will implement this API in order to allow the server to seamlessly interact with the device agent. Including creating new instances of a device, initializing it and pushing events according to the device subscriptions.

The API should enable the server to:

- Create the agent object
- Initialize the agent and with configuration information read from configuration file
- Connect the agent to the server API it to the server
- Push events to the agent
- Destroy the agent

5.2 SDK Agents

The SDK will provide implementation for several devices for testing and educational purposes.

5.2.1 Temperature Sensor Simulator

A random based temperature sensor that is configured to periodically emit a random reading of ambient temperature from a configurable range.

Sample configuration section:

```
[Temprature-1-a]
type = ambient_temp
room = room_1_a
floor = 1
config = units: F; lower: -5; upper:55; period: 12
```

The sensor will be thus configure to emit a random reading in degrees kelvin from the range -5 to +55 every 12 seconds.

5.2.2 HVAC Controller Simulator

A controller that will adjust air conditioning operation according to the desired temperature as set in the configuration. For that it will register for temperature reading events and will activate or shutdown air conditioner if ambient temperature is different from desired temperature.
It will shutdown the conditioner on fire alert.

Since this is a mockup agent it will simply write actions to the log instead of controlling an actual device.

Sample config:

```
[hvac-a-1]
type = TestHVAC
room = room_1_a
floor = 1
log = hvac_log
config = iot:10.10.1.64; tmp:77;  shutdown: Fire_Detected|ROOM_EMPTY
```

5.2.3 Fire Safety Sensor Simulator

A sensor that will emit a fire alert when receiving a SIGUSER2 signal.
Sample config:

```
[fire-drill-test]
type = TestFire
room = room_1_a
floor = 1
log = safety
config = Fire_Detected; SIGUSER2
```

5.2.4 Sprinkler Controller Simulator

A sensor that will activate sprinklers when a fire alert is received.
Sample config:

```
[sprinkler-1-a]
type = TestSprinkler
room = room_1_a
floor = 1
log = safety
config = On:Fire_Detected|Smoke_Detected; IoT: 10.10.3.42
```

6 Guidelines

Work will proceed in phases:

I. Analysis
Understand Requirements and make sure they are clear. Ask questions.

II. HLD
Deliver a high-level module/subsystem design document.
What's the responsibility of each module? What are the dependencies?

III. DLD
Detailed level design document with class diagrams and important sequence diagrams.

IV. Development
Work according to following guidance:
- Organize source code in an appropriately structured directory tree.
- Source code must be checked into source control at least 4 times each day.
- Build using make file with appropriate targets including test and demo target.

7 Deliverables

Artifacts to deliver must include:

- HLD document
- DLD Document
- Unit tests
- A Configuration file for demo
- SDK folder with a document describing how to develop a new agent