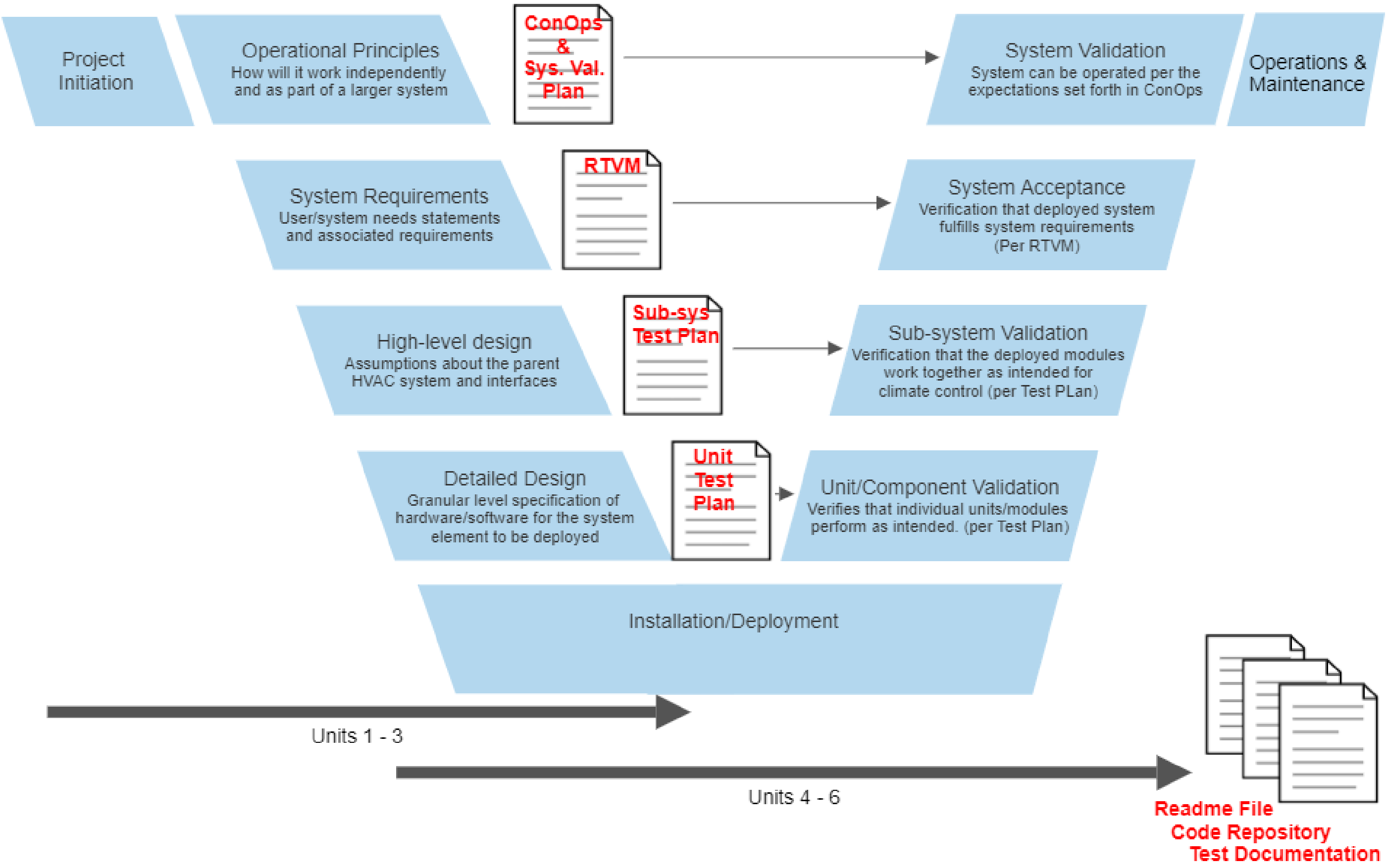
Design Specification: IOT Climate Control Solution

Introduction:

The team has been tasked to design an automated climate control system. This document outlines the design specification, security attack surface and countermeasures. The design process follows the systems engineering V-model as shown in figure 1.


Project Initiation
Operational Principles
How will it work independently and as part of a larger system
ConOps & Sys. Val. Plan
System Validation
System can be operated per the expectations set forth in ConOps
Operations & Maintenance
System Requirements
User/system needs statements and associated requirements
RTVM
System Acceptance
Verification that deployed system fulfills system requirements (Per RTVM)
High-level design
Assumptions about the parent HVAC system and interfaces
Sub-sys Test Plan
Sub-system Validation
Verification that the deployed modules work together as intended for climate control (per Test PLan)
Detailed Design
Granular level specification of hardware/software for the system element to be deployed
Unit Test Plan
Unit/Component Validation
Verifies that individual units/modules perform as intended. (per Test Plan)
Installation/Deployment
Units 1 - 3
Units 4 - 6
Readme File
Code Repository
Test Documentation

*The ConOps and System requirements are summarized to reduce word count. Figure 2 and 3 below presents the concept of operations and design logic.

Fig. 2 Operational Concept Diagram

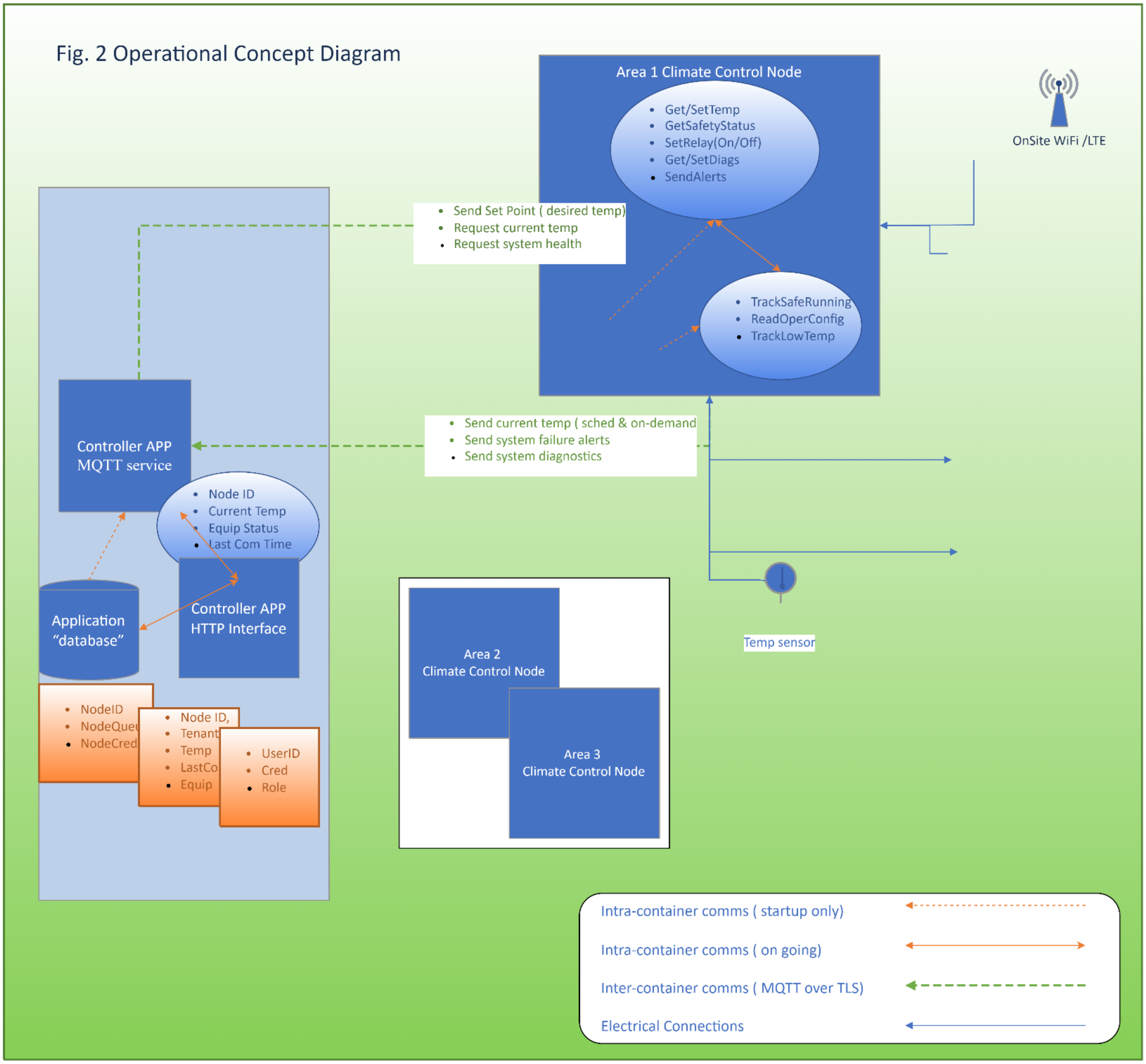
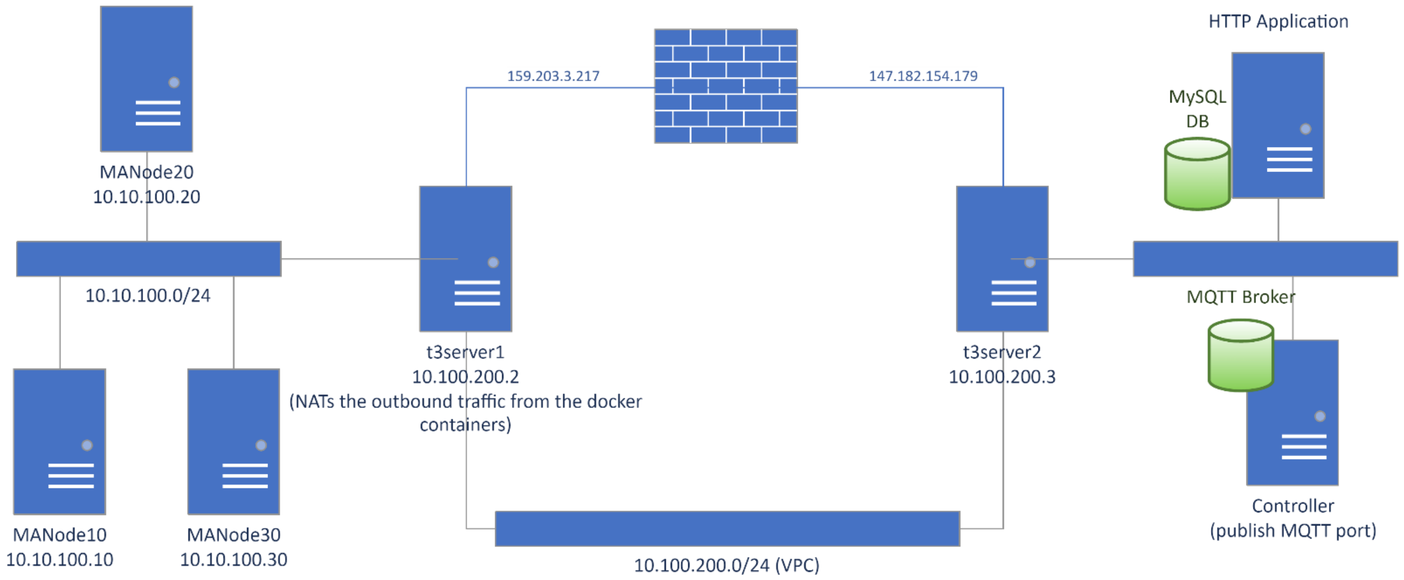


Fig. 3 System Logic



System Requirements:

The system shall consist of a centralized controller and multiple monitoring & actuating nodes (MA nodes) placed in each location heating or cooling is required.

The controller shall:

- be able to independently monitor and control multiple nodes, three shall be used for testing.
- report current temperature from each MA node.
- report last connection from each MA node.
- Employ a message system between the centralized controller and each MA node.
- Employ a TCP/IP network (wireless or wired)

Each MA Node shall:

- Employ a dedicated microprocessor with input and output capability and, enough onboard storage and memory to support operational functions in the event communications with the centralized controller are disrupted. (Commonly referred to as Fog computing within the Internet of Things (IOT) context).

Message Prioritization & Deconfliction

MQTT (Oasis, 2019) version 5.0 is employed for all messaging control.

- MQTT supports confirmed delivery of messages.
- Identifiers within the message are used within the programming logic to ensure temperature control messages are prioritized over status queries.
- MQTT supports multiple queues on the same network service allowing messages to be sent to specific queues based on their priority.
- All messages are published to a central broker service to which all clients subscribe. Each message is assigned to one client only, therefore deconfliction is not required.

Message response time requirements have not been defined in the assignment, therefore assumptions based on the intended purpose of the IOT solution have been made. MQTT protocol characteristics and the typical response requirements for building climate control systems will allow stakeholders to make choices regarding response time.

- *Active connections to the MQTT broker are monitored through TCPIP keep alive events, and the python (2021) library defaults to 60 seconds. Subscriber response time to a newly published message is typically sub second. Wang (2018).*
- *In a scenario with limited bandwidth or power the node could be modified to connect, retrieve messages, act on instructions, update status then return to low power disconnected mode.*
    - *A facility can sustain a 60-minute or longer heating failure before damage is likely.*
    - *Failure of a refrigeration unit may take several hours to reach an unacceptable temperature.*

Functional Risk Management:

Subscribe and Publish latency as well as communication outages can be further addressed through the following two design principles:

- Each heating zone controller shall include an onboard safety system to prevent damage through excessive heating or cooling.
- Communication failure shall result in a stable, fail-safe mode until communication returns.

Safety Control Requirements:

- MA nodes must only accept instructions from the authorized centralized controller
- MA nodes must only communicate with the authorized centralized controller
- The MA node programing must only act on messages that are explicitly allowed:
    - The default response must disregard all unrecognized instructions
    - All instructions that would result in exceeding predefined safety limits must be disregarded.
- All system events are recorded to an activity log
- Remote administration over VPN is permitted.
- Physical access to the system is protected for appropriate level of trust.
- Administrative access to the operating system must be restricted using non-trivial authentication and explicitly allowed devices/IPs.
- Equipment runtime must not exceed a predefined duty cycle.
- A predefined inactive period must occur between each active cycle.
- Safety and security settings must be read on system startup and not be editable during operation.
- System startup configuration is encrypted with a non-trivial key

Attack/Defense Trees

ADTree models were created for Actuator-nodes and Central Control. Figures 4/5 present the attack surface with recommended countermeasures for each. The attitude domain selected is the probability domain and the calculations are included in the diagrams.

Insert figure 4

Insert figure 5

References:

Oasis (March 7, 2019) MQTT version 5.0 Oasis Open Committee https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf [Accessed November 11, 2021]

Wang, C, (November 26th, 2018) HTTP vs. MQTT: A tale of two IoT protocols. Blog, IOT & Devices – Google Cloud. Available from: https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols [Accessed November 20, 2021]

Python (2021), Python 3.10.0 Documentation. Available from: 3.10.0 Documentation (python.org) [Accessed November 20, 2021]