Architectural Definition Document

Members

Student Number	Name
218894857	Matthew Tsz Hin Yu
219683903	Jiyoun Shim
218210278	Zamzam Mohamed

ITEC3030 – Fall 2024

Executive Summary

This document is concerned with the HVAC systems architectural design for a laboratory facility of Water Bros Inc. The design objectives include some quality criteria: including high availability, rapid fault detection, and precise temperature accuracy. The approach proposed here is based on the use of building block components, specifically furnaces, air conditioners and temperature sensors, the whole of the components integrated with Raspberry Pi controllers. Rigorous architectural patterns and tactics employed in the system ensure that it is maintainable, inexpensive to produce, and independent of a specified vendor. The document addresses the problem statement,design concepts, the architectural approaches and related models.

Problem Definition

Introduction

Water Bros Inc., a water quality testing company has its own set of requirements with an HVAC system creating an environment to achieve specific temperature ranges within the lab. The design includes different functional areas with multiple utilization, such as sensitive analysis labs, office spaces and storage areas. The system need to achieve some quality requirements: 99.999% reliability in temperature sensing, fault detection within 1 minute, and temperature change detection within 20 seconds, while ensuring temperature readings are accurate to within +/-0.5°C. Besides, the scheme should be easily replaceable and inexpensive to ensure sensors and actuators from various suppliers can be fitted at a lower cost. This document provides an explanation of the intended answer which employs modular parts and better architectural styles to achieve these goals without much effort.

Architectural Approach

Quality Scenarios

The requirements given to you state several quality requirements. Devise three **quality scenarios** that exemplify these requirements on the topics of availability, performance (accuracy) and maintainability.

Scenario - Availability	
Overview	A temperature sensor in one of the sensitive labs fails during operation.
System State	The HVAC system is functioning normally, continuously monitoring temperatures in all rooms.
System Environment	All rooms are equipped with sensors connected to Raspberry Pi controllers, transmitting data over Ethernet (for critical rooms; labs and sample storage room) and WIFI (for non-critical rooms). This ensures strong reliability and reduced latency in critical environments.
Environment Changes	One of the sensors in a lab stops sending temperature readings due to a failure.
Required System Behavior	- The system detects the faulty sensor within 1 minute using heartbeat signals. - The controller switches to a backup sensor in the same room to ensure continued monitoring. - A log entry is created to record the failure for maintenance tracking. - The system continues monitoring without disruption, ensuring 99.999% reliability.

Scenario - Performance	
Overview	A sudden spike in temperature occurs in the sample storage room due to an open door.
System State	The HVAC system is maintaining stable temperatures, and all sensors are functioning correctly.
System Environment	The room is equipped with multiple temperature sensors transmitting readings every second to the central controller.
Environment Changes	The room temperature increases rapidly due to external interference (e.g., open door).
Required System Behavior	- The controller detects the temperature change within 20 seconds by processing sensor readings. - The HVAC system is triggered to restore the temperature to the desired range. - The system ensures readings remain within ±0.5°C accuracy throughout the adjustment process.

Scenario - Maintainability	
Overview	A furnace becomes unavailable and must be replaced with a unit from another vendor.
System State	The HVAC system operates normally, integrating components from the current vendor.
System Environment	The existing furnace uses Java RMI and is compatible with Raspberry Pi controllers.
Environment Changes	A new furnace from a different vendor is installed, which does not support Java RMI or Raspberry Pi natively.

Required System Behavior	- The system integrates the new furnace using an adapter to translate the furnace's communication protocol, ensuring compatibility with the existing controller. - The replacement process is documented in logs for tracking and compliance. - The system continues to function normally, demonstrating adaptability to vendor changes.

ARCHITECTURAL PATTERNS AND TACTICS

1. Detect faults and mitigate them to achieve the availability targets

❖ Tactic/Pattern : Fault Detection and Recovery/ Active Redundancy
- Deploy active redundancy with multiple sensors and use periodic heartbeat signals to detect faulty components within 1 minute and maintain 99.999% availability.

2. Ensure accuracy of measurements

❖ Tactic/Pattern : Reuse Resources and Results
- Cache validated sensor readings to reuse them for HVAC operations which ensures ±0.5°C accuracy while minimizing redundant processing and maintaining cost-effectiveness.

3. Achieve prompt temperature change detection

❖ Tactic/Pattern : Observer pattern in Event-Based Architecture
- Use the observer pattern for sensors to notify the controller of temperature changes in real-time, enabling HVAC adjustments within 20 seconds.

4. Ensure replaceability of sensor and actuator vendors

❖ Tactic/Pattern : Adapter Pattern

- Implement adapters to integrate sensors and actuators with different protocols, ensuring compatibility while preserving the controller's core system logic.

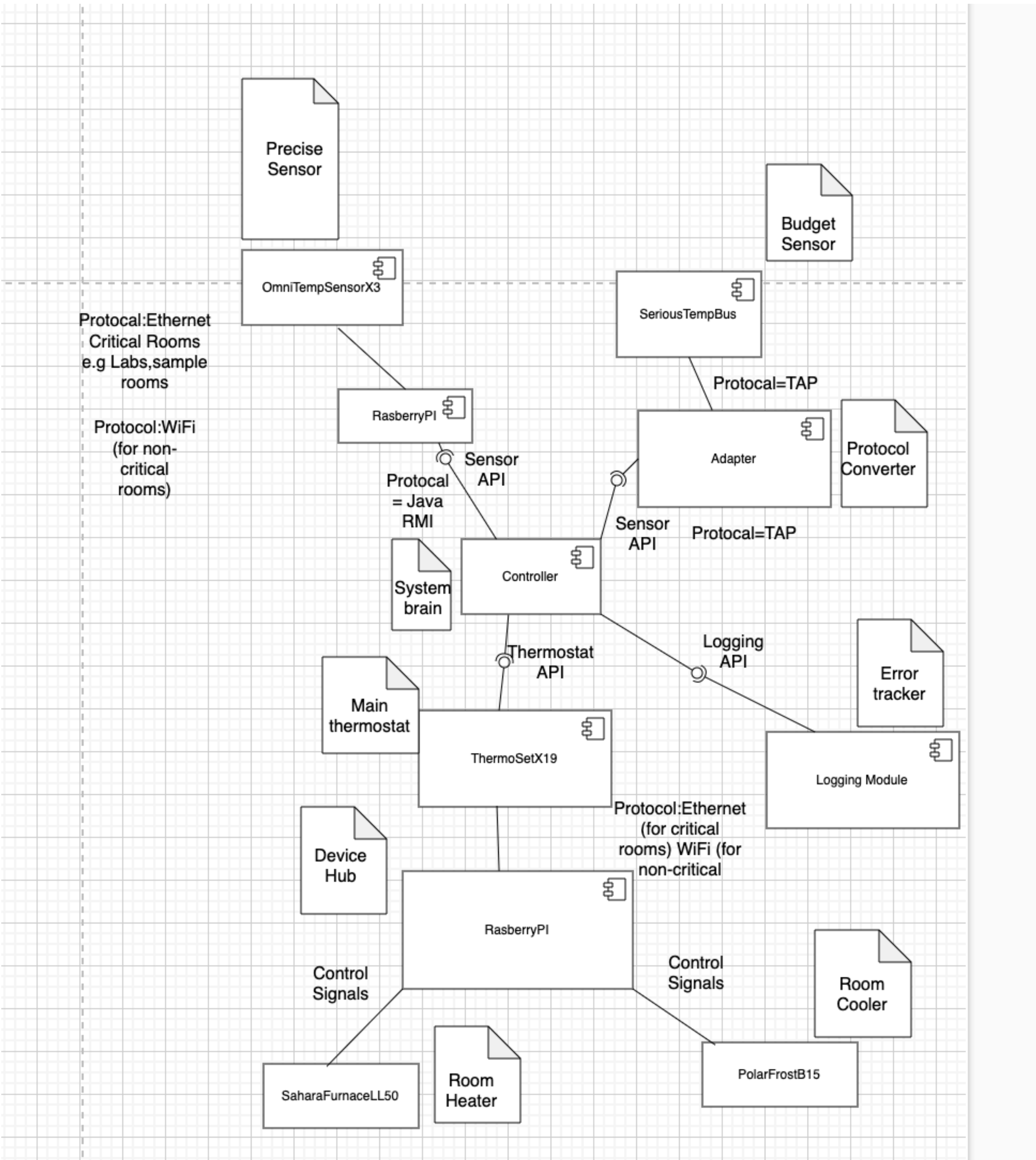
5. Attain a quality of your choosing (Maintainability)

❖ Tactic/Pattern : Information Hiding, supported by ISP, CCP, and OCP
  - Encapsulate hardware-specific logic in separate modules with grouping related changes (CCP) and allowing extensibility (OCP) while limiting exposed functionality (ISP) to minimize ripple effects and ensure maintainability.

Views

Functional Viewpoint

View



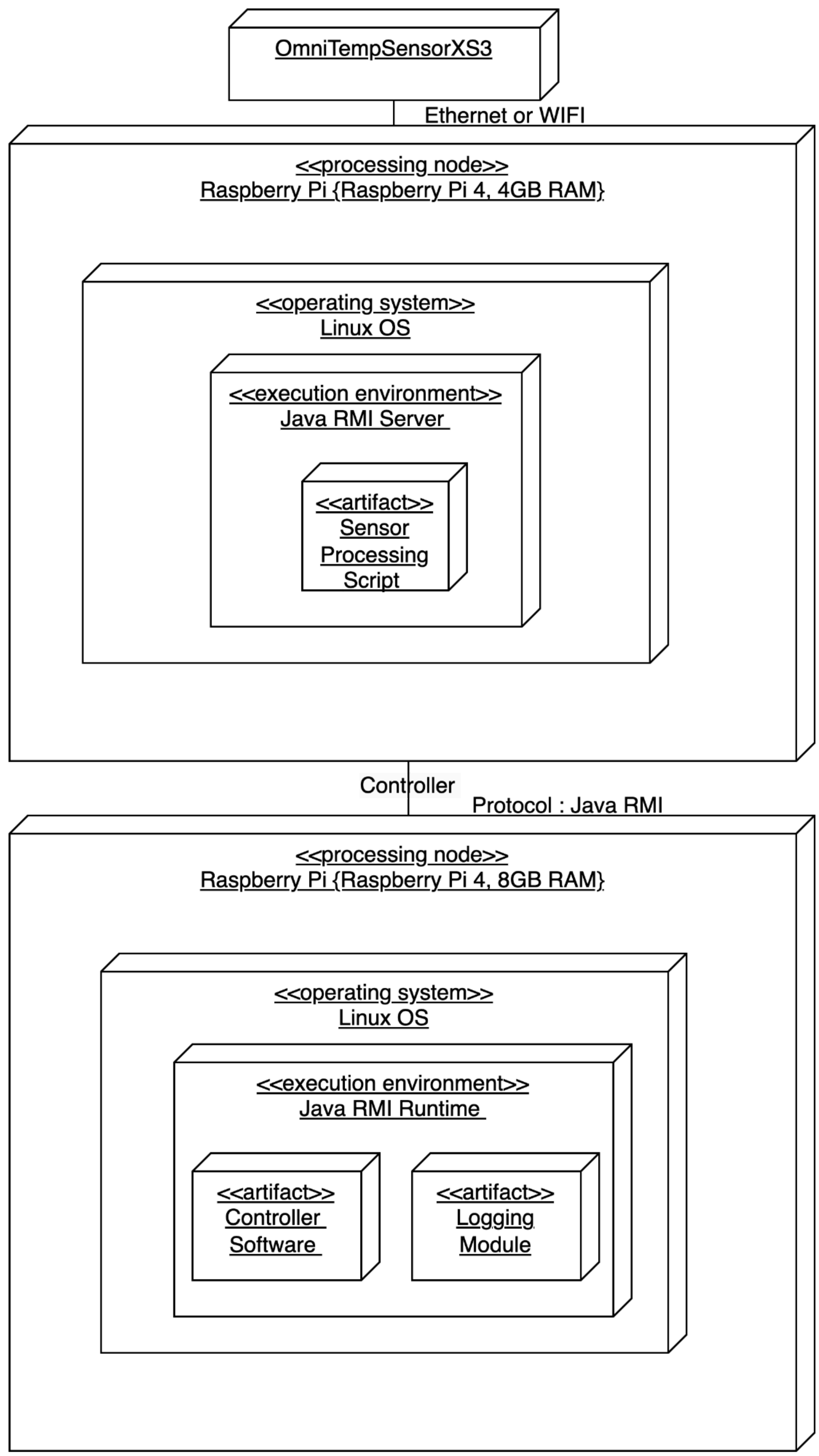
Principles Applied

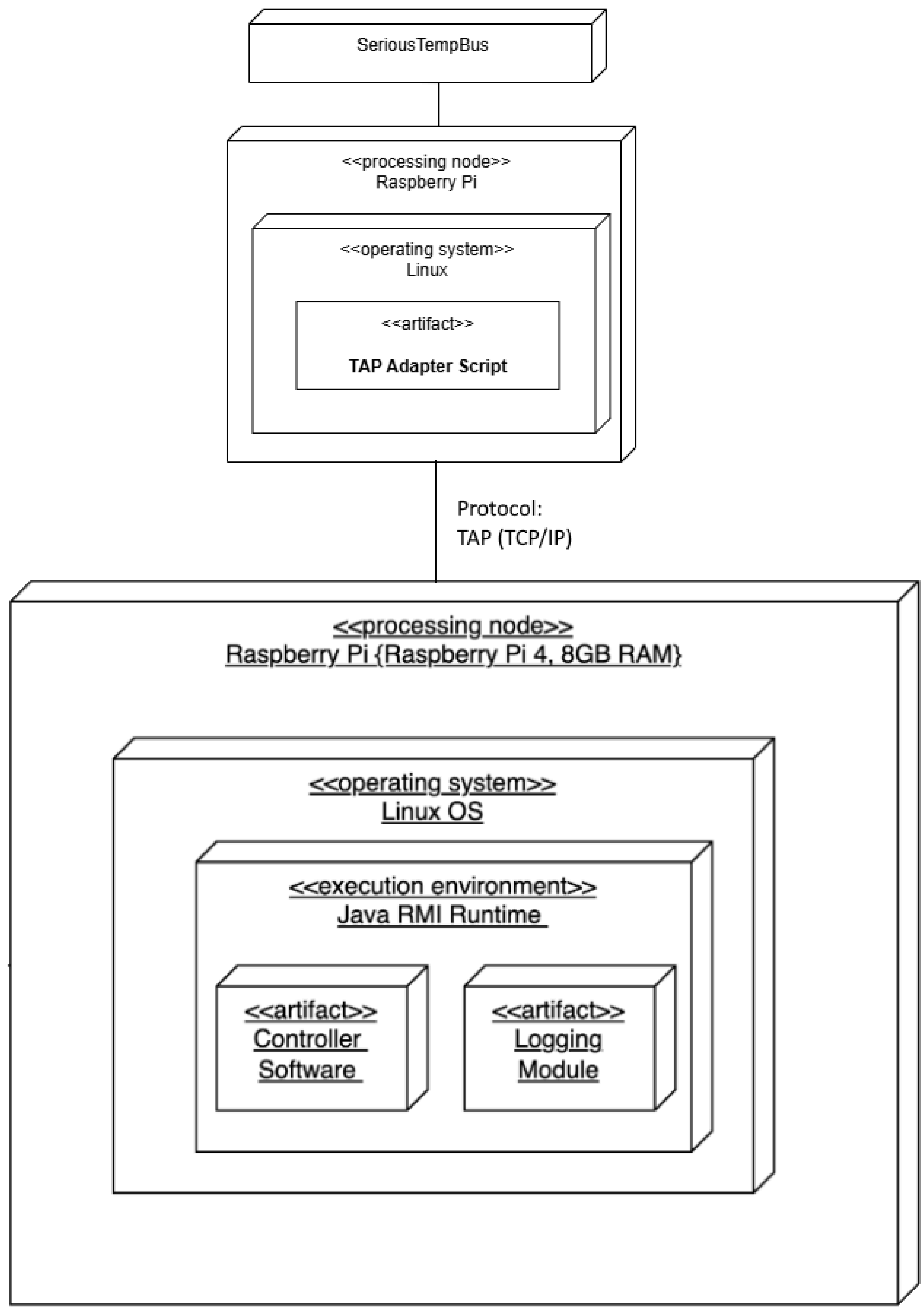
The functional diagram focuses on modularity and division of concerns by separating sensors, controllers, and actuators as distinct components. Redundancy guarantees reliability, meeting the requirement of 99.999% availability. Interfaces between components are standardized facilitating reusability and easy integration while the architecture allows to provide fault detection and rapid repair for critical subsystems.

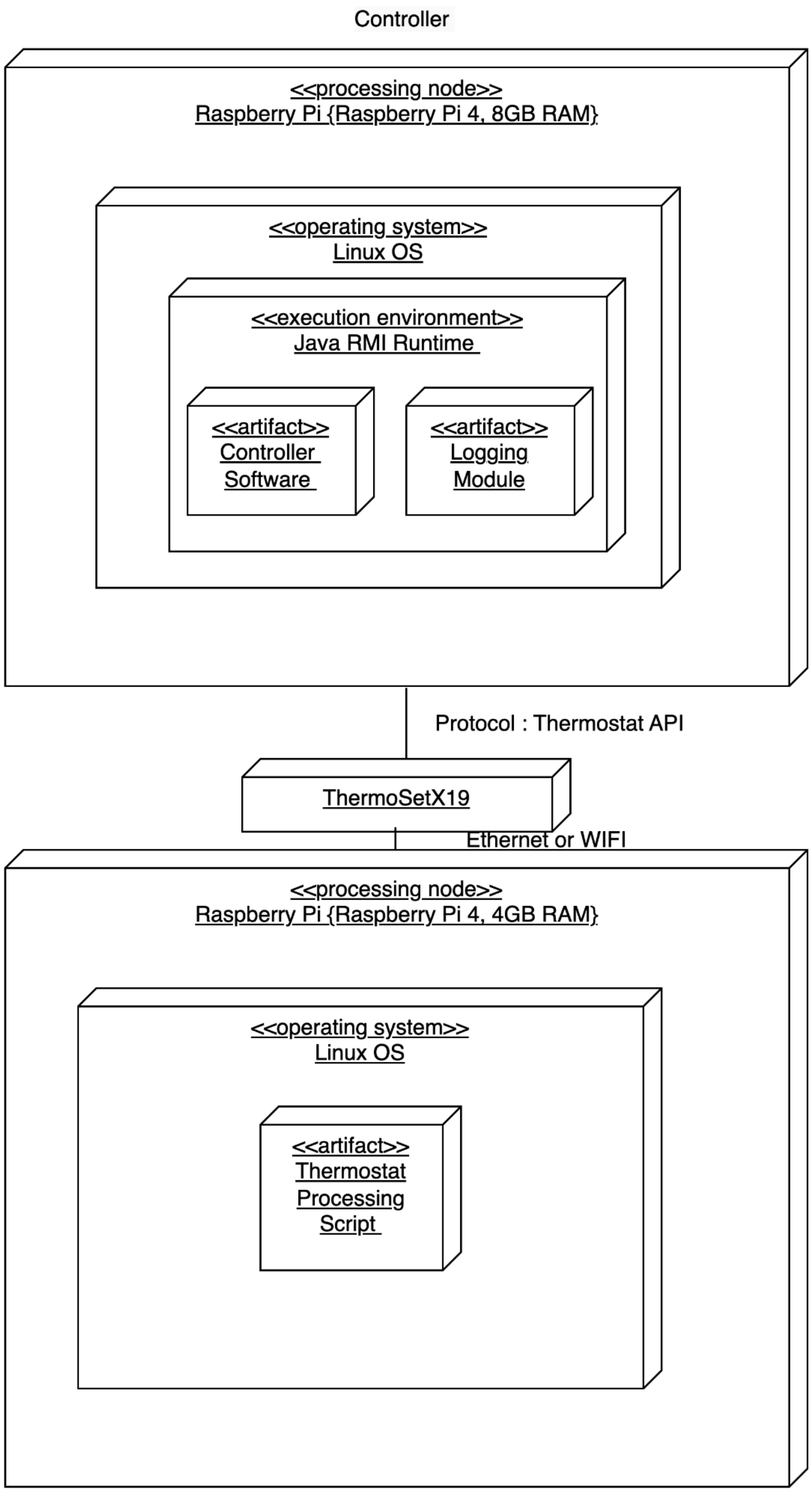
Deployment Viewpoint

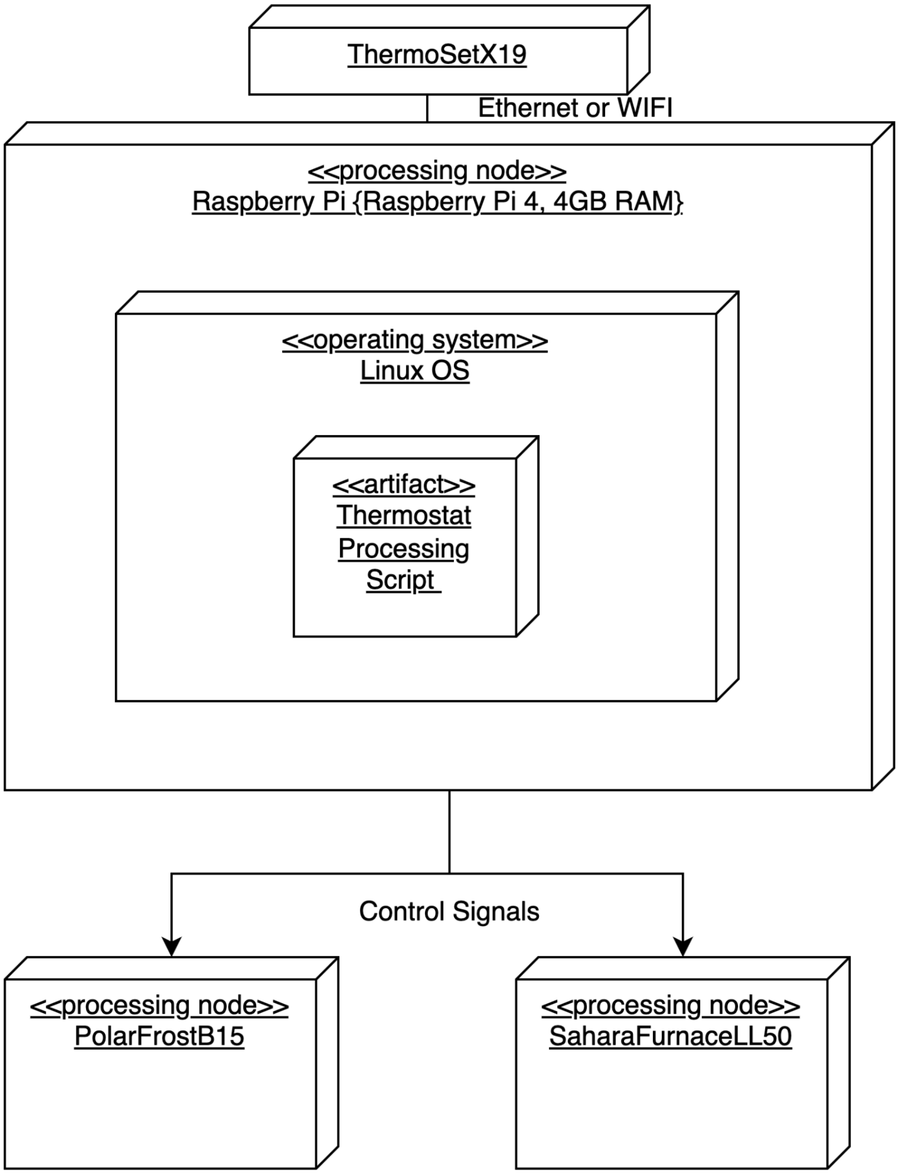
View

Since the diagram cannot fit in one page, duplicated elements can exist for better understanding.


OmniTempSensorXS3
Ethernet or WIFI
<<processing node>>
Raspberry Pi {Raspberry Pi 4, 4GB RAM}
<<operating system>>
Linux OS
<<execution environment>>
Java RMI Server
<<artifact>>
Sensor Processing Script
Controller
Protocol : Java RMI
<<processing node>>
Raspberry Pi {Raspberry Pi 4, 8GB RAM}
<<operating system>>
Linux OS
<<execution environment>>
Java RMI Runtime
<<artifact>>
Controller Software
<<artifact>>
Logging Module


SeriousTempBus
<<processing node>>
Raspberry Pi
<<operating system>>
Linux
<<artifact>>
TAP Adapter Script
Protocol:
TAP (TCP/IP)
<<processing node>>
Raspberry Pi {Raspberry Pi 4, 8GB RAM}
<<operating system>>
Linux OS
<<execution environment>>
Java RMI Runtime
<<artifact>>
Controller Software
<<artifact>>
Logging Module


Controller
<<processing node>>
Raspberry Pi {Raspberry Pi 4, 8GB RAM}
<<operating system>>
Linux OS
<<execution environment>>
Java RMI Runtime
<<artifact>>
Controller Software
<<artifact>>
Logging Module
Protocol : Thermostat API
ThermoSetX19
Ethernet or WIFI
<<processing node>>
Raspberry Pi {Raspberry Pi 4, 4GB RAM}
<<operating system>>
Linux OS
<<artifact>>
Thermostat Processing Script



Principles Applied

The deployment diagram illustrates that HVAC system components are spread across various Raspberry Pi nodes which communicate reliably through Wi-Fi/Ethernet. This distribution also allows achieving fault tolerance as well as scalability. The use of Java RMI and other inter-component standards allows for easier maintenance and independence from any vendor and compensates for the effective use of resources and costs.