SDD - 001

Ali’s Boys

Open-Source Framework for Managing Low-Cost Sensors

Software Design Document

Names:
Dr. Ali Ozdagli
Andrew Holm
Jordan Kooyman
Andrew Bryan
Lucas Wilkerson

Date: 1/29/2024

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Purpose

This Software Design Document (SDD) outlines the architectural and system design of the open-source framework project for efficiently managing low-cost sensors developed by a team of senior students from FGCU. It is intended for use by project team members, stakeholders, and anyone involved in the development and implementation of the framework.

1.2 Scope

The scope of this software project is to design and implement an open-source framework for the efficient management of low-cost sensors. The project aims to address the following goals and objectives:

- Multi-Sensor Management: We will develop a system that allows users to manage multiple sensors simultaneously, enabling deployment and oversight of sensor networks across various infrastructure sites.
- Low-Cost Sensor Integration: We will focus on integrating low-cost sensors into the framework to make it accessible to a wider range of applications and organizations with budget constraints.
- Data Storage: We will provide robust data storage capabilities to capture and securely store sensor data over time, ensuring data integrity and accessibility for analysis.
- Data Visualization: We will implement intuitive data visualization tools using Grafana, allowing for real-time monitoring and historical data analysis.
- Sensor Management: We will include features for managing sensor configurations, calibration, and diagnostics, streamlining the process of deploying and maintaining sensors in the field.
- Open-Source Framework: We will contribute to the open-source community by developing a flexible and extensible framework, allowing for collaborative improvements and customization by other developers and organizations.

1.3 Overview

This document serves as a comprehensive guide to the software design and architecture of the open-source framework project. It is organized into several sections, each addressing specific aspects of the project's design and implementation. These sections include system overview, system architecture, data design, and human interface design.

2.0 SYSTEM OVERVIEW

The open-source framework project will develop a software system for efficiently managing low-cost sensors used for structural health monitoring in civil infrastructure. This system will enable users to deploy, monitor, and maintain sensor networks across various infrastructure sites. It focuses on integrating low-cost sensors, storing sensor data, visualizing data, and managing sensor configurations. The project's background lies in the need for cost-effective and sustainable solutions for monitoring and ensuring the safety of civil infrastructure.

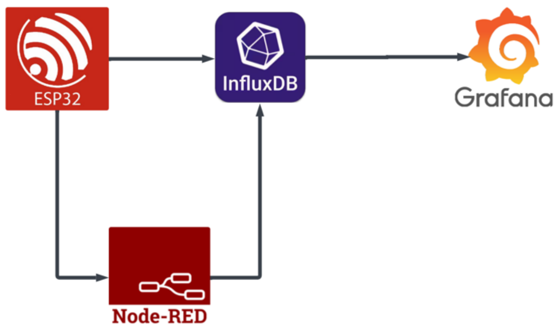
3.0 SYSTEM ARCHITECTURE

The system architecture is designed to be modular and flexible to achieve the complete functionality of the system. The major high-level subsystems and their assigned roles or responsibilities include:

- Sensor Management Subsystem: This subsystem is responsible for configuring and managing sensors, including calibration, diagnostics, and status monitoring.
- Data Management Subsystem: Handles the storage, retrieval, and secure storage of sensor data over time. It utilizes InfluxDB as the time-series database for efficient data storage.
- Data Visualization Subsystem: Employs Grafana as the primary platform for data visualization, providing customizable dashboards for real-time monitoring and historical data analysis.
- Hardware Interface Subsystem: Manages the interaction with the hardware components, specifically the ESP32-based sensors, and the Linux server.

These subsystems collaborate to ensure the efficient and reliable monitoring of the structural health of civil infrastructure.

3.2 Architectural Design

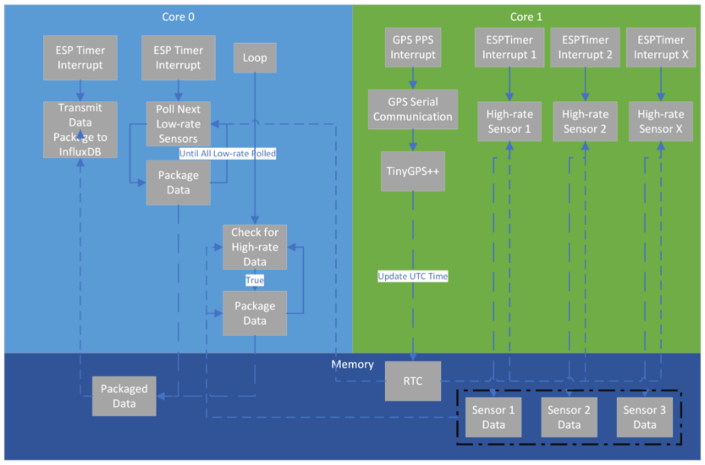


- **ESP32 with Sensors**: The ESP32 communicates with attached sensors to collect data related to structural health parameters and periodically send this data to InfluxDB, the ESP32 sends battery power, internet connection, and other board-specific data to Node-Red to manage sensors and errors.
- **Node-RED** (Error Handling and Sensor Management):
  - Node-RED acts as a central control and automation platform.
  - It continuously receives data from ESP32 sensors and checks for error cases or anomalies in the sensor data.
  - If an error or issue is detected, Node-RED can trigger specific actions, such as sending alerts, recording error information, or initiating corrective actions.
  - Node-RED also manages sensor configurations, calibration, and diagnostics as part of sensor management.

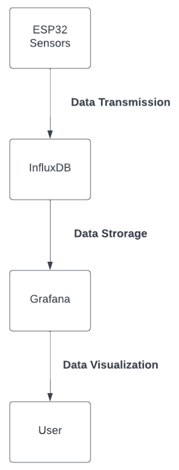
- **InfluxDB** (Data Storage):
  - Node-RED sends both the raw sensor data and any error data to InfluxDB for storage.
  - InfluxDB efficiently stores the data in a time-series format, making it available for historical analysis and data retrieval.
- **Grafana** (Data Visualization):
  - Grafana connects to InfluxDB to retrieve both the sensor data and any error-related data.
  - Users can monitor real-time sensor data and error status through Grafana dashboards.
  - Grafana provides historical data analysis, allowing users to identify patterns and trends related to sensor health and potential issues.

3.3 Decomposition Description

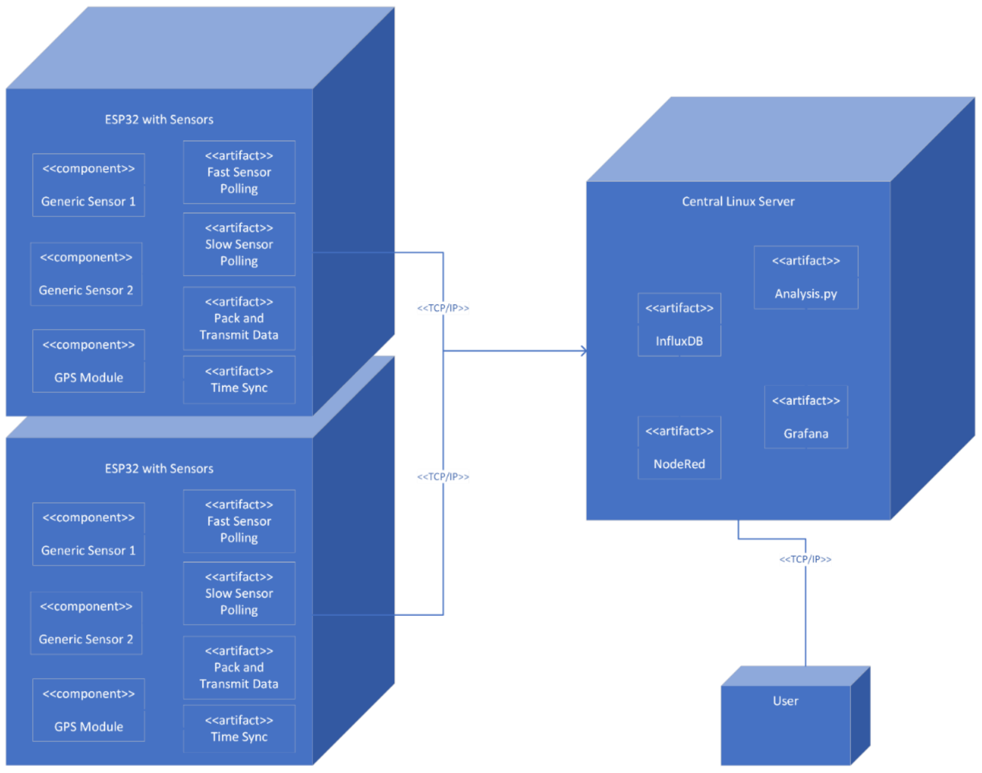
3.3.1 **Hardware Interface Subsystem**: This diagram showcases the sequence of execution of each separate process on both of the cores of the ESP32 microcontroller. This also shows how the different processes interact with each other to accomplish the task of data polling and transmission. On Core 1, there will be an ESPTimer Interrupt for each attached high-rate sensor, while Core 0 will only have an ESPTimer Interrupt for each polling period needed for the low-rate sensors attached. The shared memory of the system will be used to transfer data between cores asynchronously.



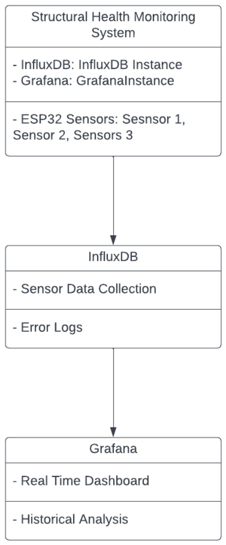
3.3.2 **Sequence Diagram**: This diagram showcases our system in chronological order and how these components collaborate to achieve the monitoring and visualization we've achieved. This diagram is ideal if you desire a step-by-step guide of how the data is collected, stored, and visualized in our system.



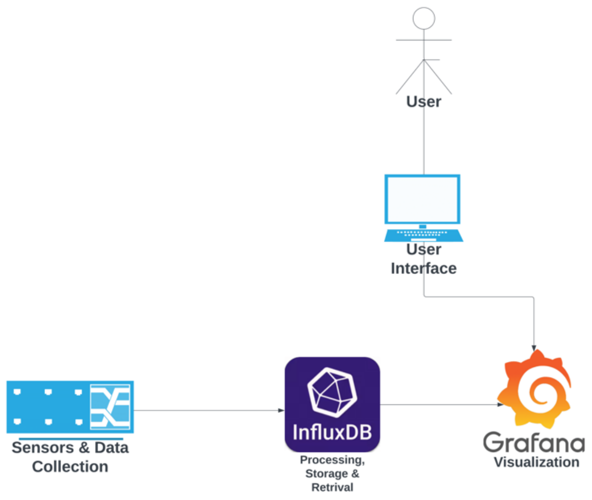
3.3.3 **Deployment Diagram**: This deployment diagram is a representation of the architecture of our system and how the hardware (ESP32 with sensors) interacts with software (Linux Server) to deploy and interconnected system. Our deployment diagram is composing of components (representing hardware devices), and artifacts (representing software components or files).


ESP32 with Sensors
<<component>> Generic Sensor 1
<<component>> Generic Sensor 2
<<component>> GPS Module
<<artifact>> Fast Sensor Polling
<<artifact>> Slow Sensor Polling
<<artifact>> Pack and Transmit Data
<<artifact>> Time Sync
ESP32 with Sensors
<<component>> Generic Sensor 1
<<component>> Generic Sensor 2
<<component>> GPS Module
<<artifact>> Fast Sensor Polling
<<artifact>> Slow Sensor Polling
<<artifact>> Pack and Transmit Data
<<artifact>> Time Sync
<<TCP/IP>>
<<TCP/IP>>
Central Linux Server
<<artifact>> Analysis.py
<<artifact>> InfluxDB
<<artifact>> Grafana
<<artifact>> NodeRed
<<TCP/IP>>
User

3.3.4 **Object Diagram**: This diagram represents instances of classes and their relationships at a particular point in time. Structural Health Monitoring System is an object representing the entire system, InfluxDB is an object instance of data storage, and the Grafana object represents the Grafana instance for data visualization.


Structural Health Monitoring System
- InfluxDB: InfluxDB Instance
- Grafana: GrafanaInstance
- ESP32 Sensors: Sesnsor 1, Sensor 2, Sensors 3
InfluxDB
- Sensor Data Collection
- Error Logs
Grafana
- Real Time Dashboard
- Historical Analysis

3.3.5 **Data Flow Diagram**: This diagram represents how data flows throughout our system using symbols. This is used to help understand where the user and the system interact, and how the data is transported.



3.4 Design Rationale

The selected architecture was chosen for several reasons:

- Modularity: The modular design allows for flexibility and ease of future modifications, ensuring adaptability to changing requirements.
- Data Separation: By separating data management from data visualization, the system can scale efficiently, and data can be accessed independently.
- Industry Tools: Leveraging InfluxDB and Grafana allows for efficient data storage and visualization, benefiting from established and reliable solutions.
- Hardware Interface: The hardware interface subsystem is crucial for effective communication with the low-cost sensors and ensures real-time data collection.

Alternative architectures, such as a monolithic design, were not considered because they lacked the modularity and scalability required for this project. The selected architecture provides a balance between efficiency, flexibility, and reliability.

This architecture ensures that the system can effectively manage low-cost sensors for structural health monitoring while allowing for future improvements and adaptability to different sensor types and infrastructure needs.

4.0 DATA DESIGN

4.1 Data Description

InfluxDB is utilized as the time-series database for storing sensor data efficiently and providing easy retrieval and querying capabilities. It plays a critical role in managing and organizing data related to the structural health of civil infrastructure. The information domain of the system is transformed into structured data through InfluxDB, allowing for secure data storage and efficient processing.

- InfluxDB provides the following key features:
    - Time-Series Data Storage: It stores sensor data in a time-series format, allowing for the efficient management of data over time.
    - High Write and Query Throughput: InfluxDB is optimized for high write and query throughput, making it suitable for real-time data ingestion and retrieval.
    - Data Retention Policies: It supports data retention policies, enabling the system to automatically manage and expire old data based on defined policies.
    - Data Tagging and Fields: Data is organized using tags and fields, allowing for a flexible schema to represent different sensor types and measurements.

4.2 Data Dictionary

Here's a data dictionary for the major data entities used with InfluxDB in the context of this project:

- Measurement Data
    - Type: Time-Series Data
    - Description: Sensor measurements are written to InfluxDB as Influx line protocol. These measurements are then stored in time-structured merge tree and time series index files. Measurements may include data related to structural health parameters (e.g., vibration, strain, temperature).
- Tag Sets
    - Type: Metadata
    - Description: Tags are used to label and categorize data points using a key-value pair format. For example, tags may include the sensor's ID, location, and type.
- Field Sets
    - Type: Metadata
    - Description: Fields are used to represent the actual data values, such as vibration amplitude, temperature, or strain values.
- Retention Policies
    - Type: Metadata
    - Description: Retention policies define how long data should be retained in the database before it's automatically deleted.
- Queries
    - Type: Command
    - Description: Queries are used to retrieve specific data from InfluxDB based on specified criteria, including time range and tags.
- Database
    - Type: Data Storage
    - Description: A time-series database, specialized in handling time-stamped data efficiently.

The data dictionary describes the primary data entities and metadata used in the InfluxDB database for this project. These data entities are used to represent and store the sensor measurements and associated metadata efficiently.

5.0 HUMAN INTERFACE DESIGN

5.1 Overview of User Interface

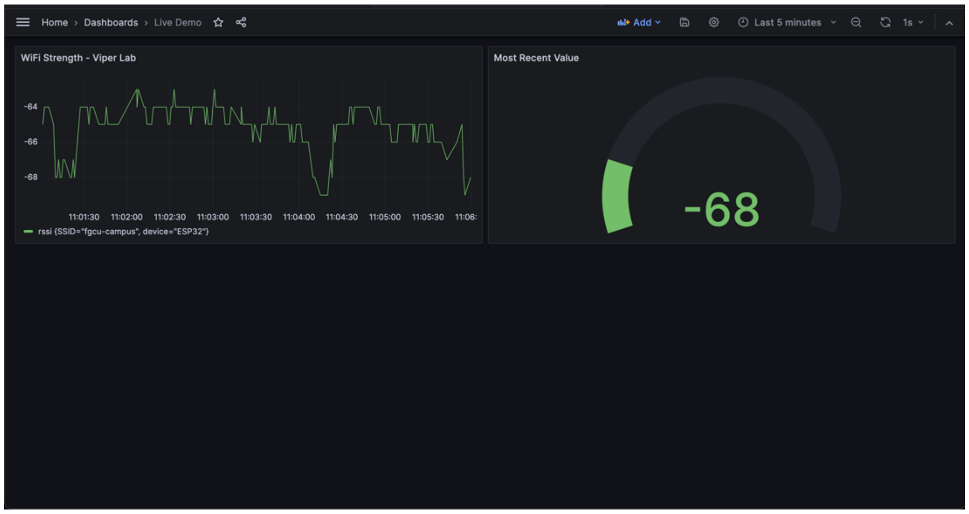
Grafana is utilized as the primary platform for data visualization and providing an intuitive user interface for monitoring the structural health of civil infrastructure. From the user's perspective, the system offers the following functionality:

- Real-Time Monitoring: Users can view real-time data from the sensors, such as vibration levels, strain, or temperature, through customizable dashboards.
- Historical Data Analysis: Historical data is accessible, allowing users to analyze trends and patterns over time to make informed decisions about structural health.

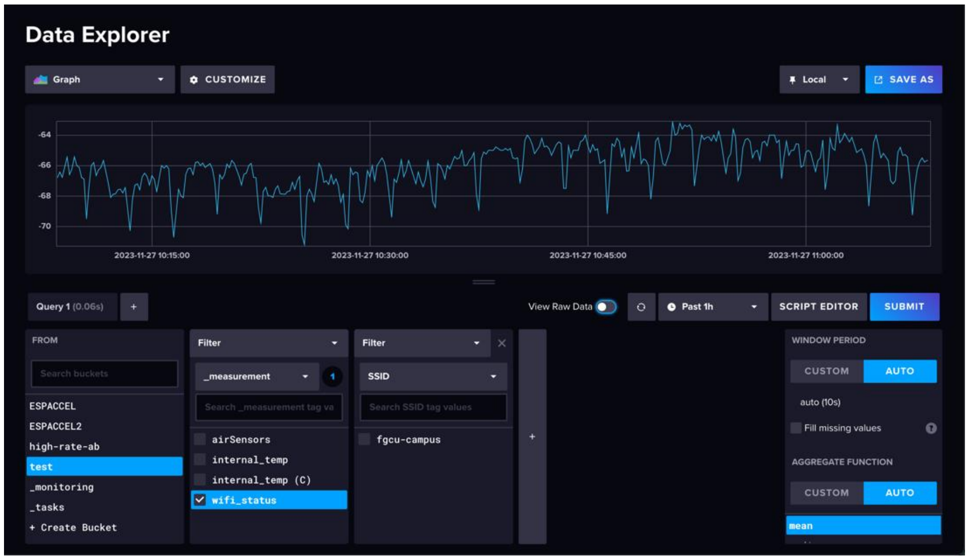
- Customizable Dashboards: Grafana allows users to create customized dashboards by selecting the sensors and parameters to display. Users can arrange panels and choose visualization options.
- Alerting and Notifications: The system can trigger alerts and notifications when sensor data exceeds predefined thresholds, ensuring immediate attention to critical issues.
- Data Exploration: Users can explore data using zoom and pan features to focus on specific time intervals for in-depth analysis.

5.2 Screen Images

Grafana Screenshot:



Influxdb Screenshot:



5.3 Screen Objects and Actions

Screen Objects in Grafana include:

- Dashboards: These are the primary screens where users can customize and view data panels.
- Data Panels: Panels display specific sensor data, and users can configure them with various visualization options.
- Alerting Rules: Users can set up alerting rules to trigger notifications or actions based on data thresholds.
- Time Picker: Allows users to select the time range for data visualization.
- Zoom and Pan Tools: Provide interactive data exploration capabilities.

User actions include creating and editing dashboards, adding panels to dashboards, defining alerting rules, setting time ranges, and exploring data using zoom and pan tools.