University of Pisa

Artificial Intelligence and Data Engineering
Business and Project Management

Analyzing and Enhancing *Product Summaries* from Amazon Reviews Using ChatGPT API: A Parameter-based Approach

Ricky Marinsalda
Vittoria Acampora

August 25, 2023

Contents

1 Introduction **2**

2 Architecture **3**
2.1 Sensors . . . 3
2.2 Data encoding . . . 3

3 Analytics **4**
3.1 Database . . . 4
3.2 Grafana . . . 4

4 Implementation's Details **5**
4.1 Collector . . . 5
4.2 IoT Nodes . . . 5
4.2.1 The `Actuator` Interface . . . 5
4.2.2 The `ActuatorManager` Interface . . . 5
4.2.3 The `Fan` Class . . . 5

Chapter 1

Introduction

Chapter 2

Architecture

We have the following sensors:

- **float** sensors, used to monitor the water level
- **co2** sensors, used to monitor the co2 level
- **humidity**, used to monitor the environment's humidity

and the following actuators:

2.1 Sensors

2.2 Data encoding

Chapter 3

Analytics

3.1 Database

We're using a *MySQL*-compatibile database management system, cllaed *MariaDB*. We're storing historic sensors' data in tables, one for each sensor's class, in the form of tuples: timestamp, sensor's id and sensor's datum. The following *DDL* was used to build the zoo's database:

3.2 Grafana

Our project incorporates a real-time monitoring and visualization aspect through the implementation of a

Chapter 4

Implementation's Details

4.1 Collector

4.2 IoT Nodes

This code represents a system for managing actuators in an Internet of Things (IoT) environment. Let's break down the code and explain its functionality:

4.2.1 The `Actuator` Interface

The `Actuator` interface defines the contract for an actuator, which is responsible for sending

4.2.2 The `ActuatorManager` Interface

4.2.3 The `Fan` Class

The `Fan` class implements the `Actuator` interface and represents a specific type of actuator for a fan. It includes a constructor that takes an IP address as a parameter and initializes a CoAP client for communication.