CHAPTER 1

LOW LEVEL DESIGN

1.1 Introduction

By utilising data analysis and machine learning techniques to forecast when maintenance is required, predictive maintenance is a proactive maintenance strategy that tries to identify probable equipment faults before they happen. Lora (Long Range), a wireless communication technology that permits long-range, low-power communication between devices, is one tool that may be utilised for preventative maintenance.

Real-time data from sensors on equipment, such as those measuring temperature, vibration, and pressure, may be collected using Lora. This data can then be analysed to look for trends and anomalies that could point to potential breakdowns. In order to save downtime, lower maintenance costs, and enhance equipment dependability, maintenance teams may use predictive maintenance with Lora to proactively solve problems before they become serious.

Companies must set up a network of Lora sensors and gateways that can communicate data to a centralized data management system in order to execute predictive maintenance using Lora. The data may then be analysed by this system using machine learning techniques to provide information on the operation and upkeep requirements of the equipment. For companies that depend on reliable equipment, predictive maintenance utilising Lora has the potential to revolutionize maintenance procedures and result in considerable cost savings.

1.2 Design Considerations

1.2.1 Design Goals

The design approach followed is data-driven approach.It involves collecting data from equipment(Ned Niryo 2 Robot), processing and analyzing it using machine learning algorithms to predict when maintenance or repairs will be required.

1.2.2 Architecture Choices

1. Pub/Sub Model
2. Client/Server Model

1.4 Use Case Diagram

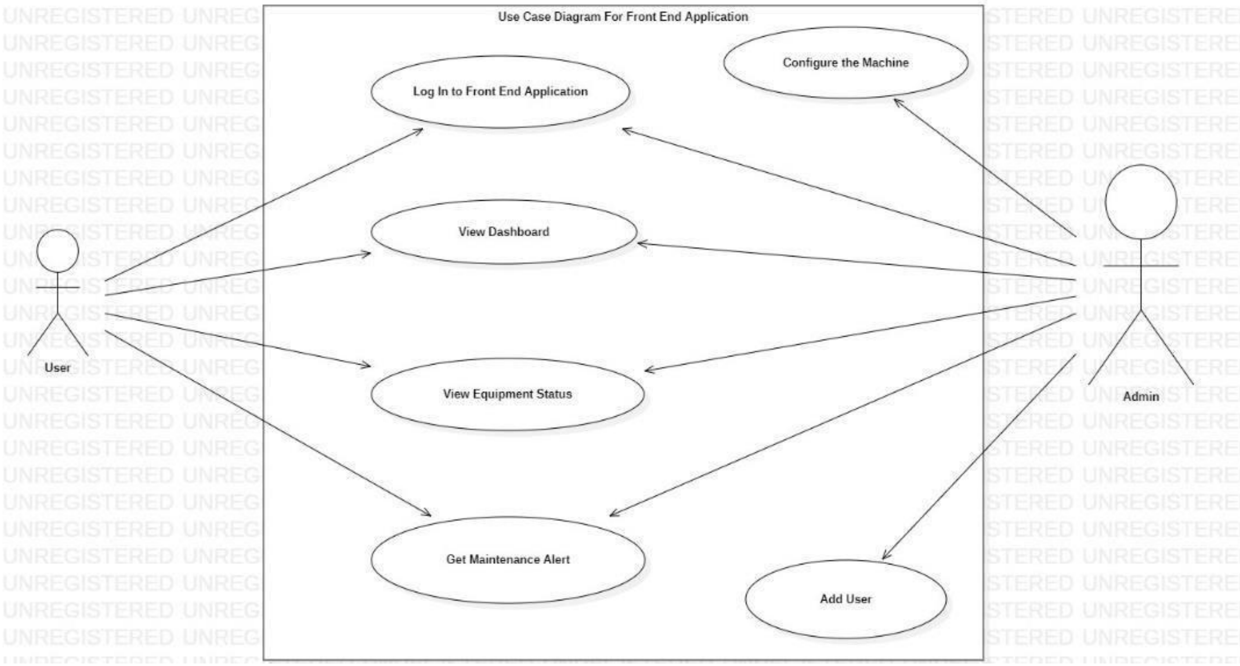

Figure 1.2: - Use Case Diagram

1.5 External Interfaces

- **LoRaWAN Interface:**

Purpose: Promotes communication between the LoRaWAN network and edge devices (ESPs, Arduinos, etc.).

Specifications:

- outlines the protocol for long-distance data transmission.
- describes the process by which devices join and authenticate with the LoRaWAN network.

- **MQTT Interface:**

Purpose: allows for communication between the gateway or central server and edge devices.

Specifications:

- Specifies MQTT topics for sensor data publication and subscription.
- Describes the MQTT message format that is used for communication between devices and the central server.

- **The Things Network (TTN) Interface:**

Purpose: acts as a bridge between edge devices and the backend.

Specifications:

- Describes how data transmission and storage can be integrated with TTN.
- Specifies the authentication procedures and TTN API endpoints.

- **Machine Learning Model (SVM) Interface:**

Purpose: combines the Support Vector Machine (SVM) with data analysis and forecasting.

Specifications:

- explains the format that the SVM model expects for input data.
- specifies the forecasts' or classifications' output format.

- **Dashboard Interface (Django, JavaScript, HTML):**

Purpose: Provides a user interface for data visualization and interaction.

Specifications:

- describes the Django API endpoints that can be used to get processed data back.
- explains how the JavaScript and HTML elements behave and are organised.

- **ROS (Robot Operating System) Topics Interface:**

Purpose: promotes communication amongst the various robotic system components (Niryo).

Specifications:

- Defines ROS topics for exchanging messages between robotic components.
- Specifies the data format and frequency of communication.

- **Documentation Interface:**

Purpose: Provides comprehensive documentation for developers and users.

Specifications:

- Includes API documentation for MQTT, TTN, Django, and ROS interfaces.
- Provides guides on setup, configuration, and troubleshooting.

- **Data Flow and Integration Interface:**

Purpose: Illustrates the flow of data and integration points between different components.

Specifications:

- diagrams showing how data is transferred from edge devices to the dashboard.

- explains the steps involved in processing, SVM model analysis, and dashboard presentation of data.

- **Versioning Interface:**

Purpose: Allows for updates and changes without disrupting existing integrations.

Specifications:

- explains version control methods for API endpoints, MQTT topics, and other interfaces.
- describes the protocol used to transmit version information between requests and answers

CHAPTER 2

SYSTEM DESIGN

2.1 IoT Architecture

An IoT system's layers and components are described in detail by the Internet of Things (IoT) architecture. It offers a methodical approach for creating interoperable, scalable, and flexible Internet of Things systems.

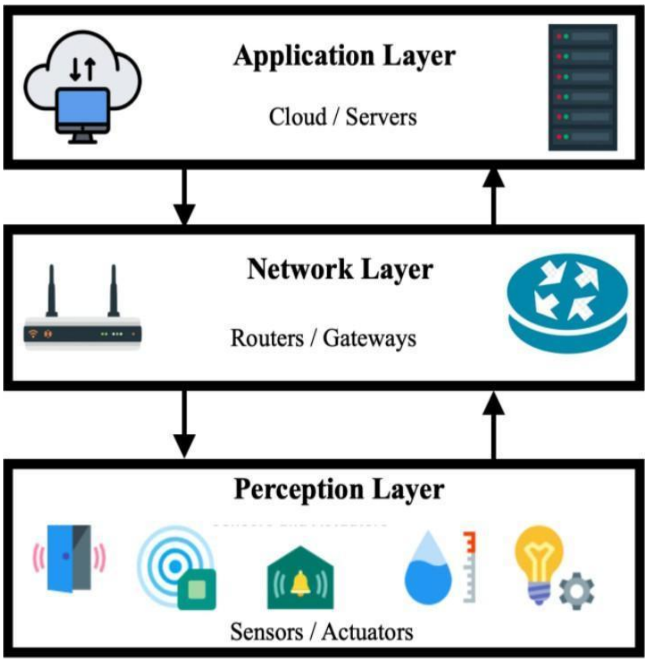

Figure 2.1: - A Generic Three Layered IoT Architecture

The above components are typically organized into three layers:

1. The perception layer, which includes the devices and sensors that collect data from the environment.
2. The network layer, which includes the connectivity layer and the cloud platform that facilitates the transfer and storage of data.
3. The application layer, which includes the analytics and applications that make use of the data generated by IoT devices.

2.2 LoRa Architecture

A low-power wide-area network (LPWAN) technology called LoRa (Long Range) is utilised for long-distance communication between IoT devices. The LoRa architecture is made up of a number of parts that work together to support low-power, long-range wireless communication. The following are the main elements of LoRa architecture:

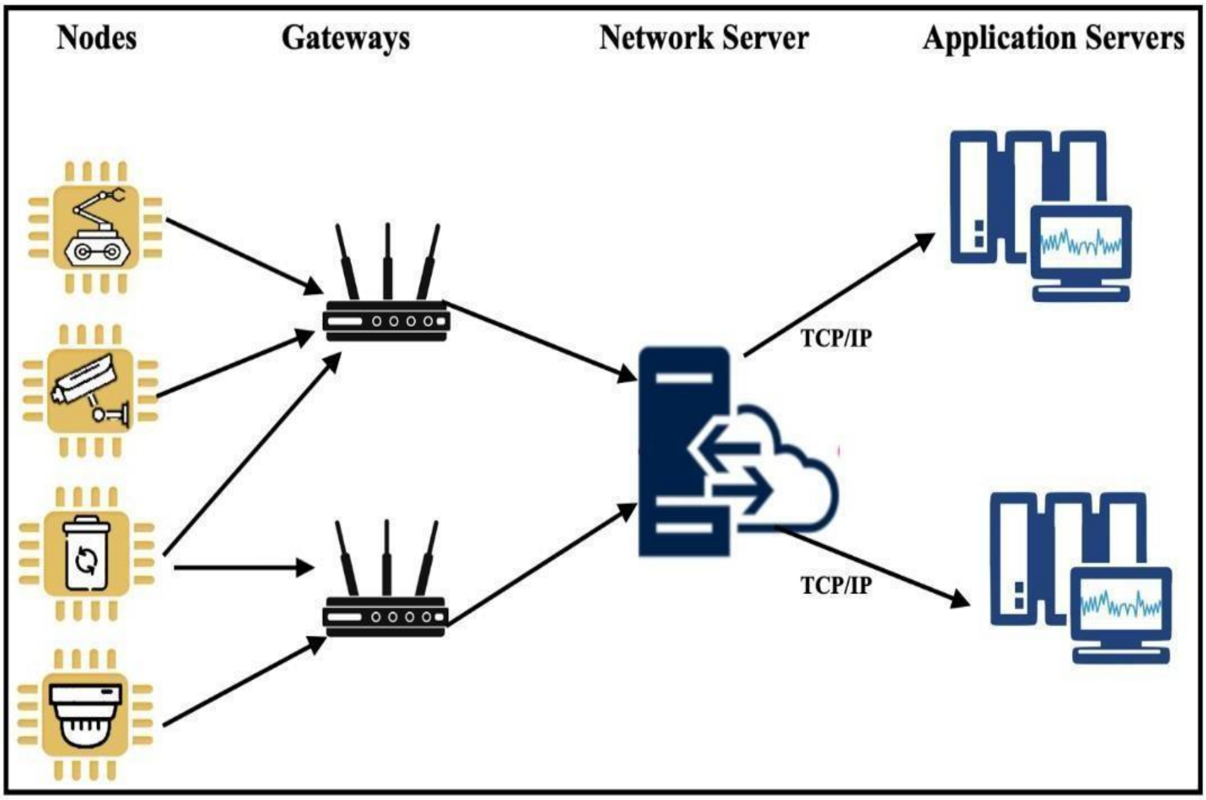

Figure 2.2: - A Generic LoRa Architecture

1. End Devices: These are the Internet of Things (IoT) gadgets that use LoRa technology to deliver and receive data. They are often battery-powered and made to use little electricity.
2. Gateways: Data from end devices is forwarded to the network server through LoRa gateways. They are in charge of translating the LoRa radio signal into internet-transmittable IP packets.
3. Network Server: The network server controls the flow of information between applications and endpoints. It receives data from gateways and routes it to the appropriate application server

CHAPTER 3

METHODOLOGIES

3.1 Robot Design

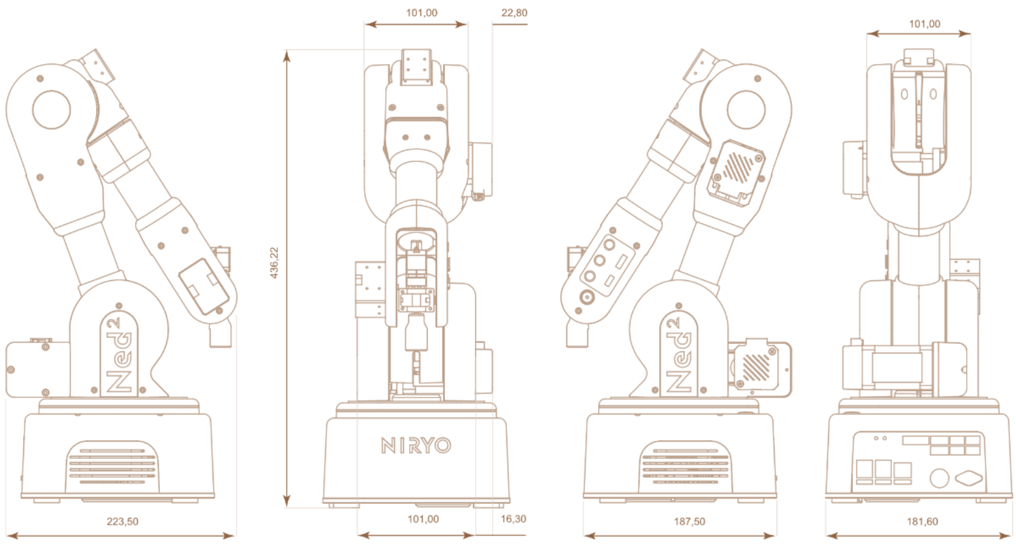

Figure 3.1: -Structure of Robot

Ned2, an open-source collaborative robot with six axes of motion, was created. It is meant for research, teaching, and Industry 4.0. Ned's replacement, Ned2, is more reliable and contains several enhancements that, when combined with an enhanced human-machine interface, enable users to learn collaborative robotics even more.

The rear panel of the Ned 2 is open and easily accessible, making it headache-free to attach your sensors and accessories.

Figure 3.2: -Back Panel

3.2 Pub/Sub Model

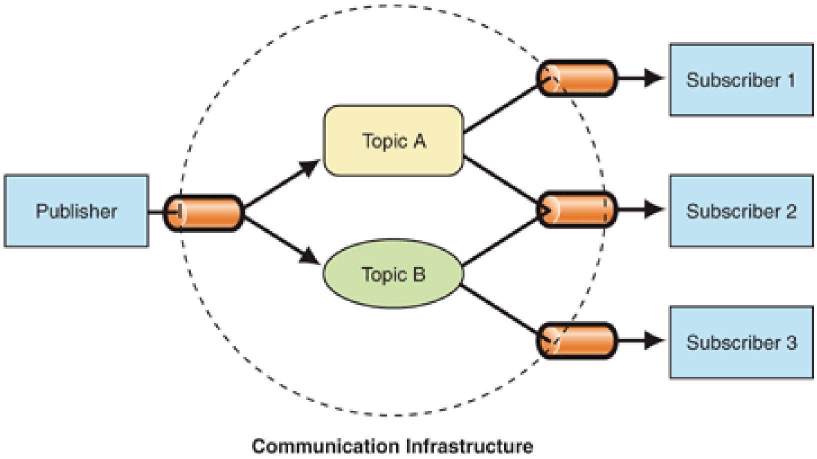

Figure 3.3: - Pub Sub Model

In order to enable publishers to send messages to a central hub (broker) and subscribers to receive particular messages of interest, the publish-subscribe (pub-sub) model is used to facilitate communication between components.

3.3 Client/Server Model

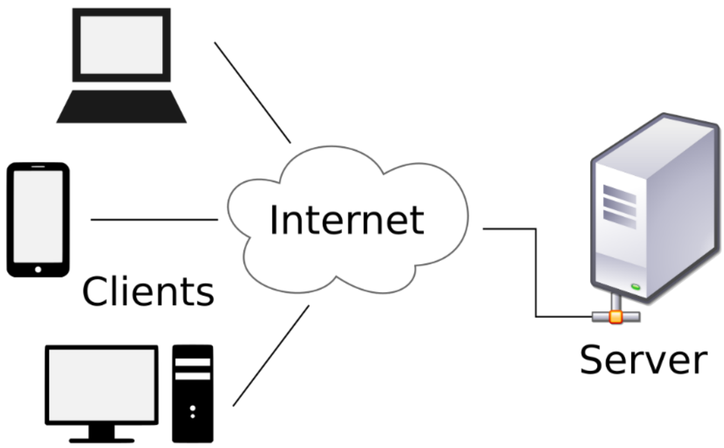

Figure 3.4: - Client/Server Model

In a client-server computing architecture, client devices send requests for resources or services, and servers respond by sending the required information or functionality.