**Recycling Robot Visualization**

Simplified Agile SRS

Juan Jimenez

Magnus Eriksson

Vitor Medeiros

Yuvrajsinh Adajania

6/17/2025

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Rev | Date | Description of Change | By |
| 1.0 | 6/17/2025 | Document creation | Juan Jimenez  Magnus Eriksson  Vitor Medeiros  Yuvrajsinh Adajania |

Table of Contents

[Revision History ii](#_Toc156920272)

[Section 1: Introduction 1](#_Toc156920273)

[Section 1.1: Purpose 1](#_Toc156920274)

[Section 1.2: Scope 1](#_Toc156920275)

[Section 1.3: Definitions, Acronyms, and Abbreviations 1](#_Toc156920276)

[Section 1.4: References 1](#_Toc156920277)

[Section 1.5: Overview 1](#_Toc156920278)

[Section 2: Overall Description 1](#_Toc156920279)

[Section 2.1: Product Perspective 1](#_Toc156920280)

[Section 2.1.1: User Interfaces 1](#_Toc156920281)

[Section 2.1.2: Software Interfaces 2](#_Toc156920282)

[Section 2.1.3: Operations 2](#_Toc156920283)

[Section 2.2: Product Functions 2](#_Toc156920284)

[Section 2.2.1: Function #1 2](#_Toc156920285)

[Section 2.3: User Characteristics 3](#_Toc156920286)

[Section 2.4: Constraints 3](#_Toc156920287)

[Section 2.4.1: Regulatory policies 3](#_Toc156920288)

[Section 2.4.2: Hardware limitations 3](#_Toc156920289)

[Section 2.4.3: Interfaces to other applications 3](#_Toc156920290)

[Section 2.4.4: Parallel Operation 3](#_Toc156920291)

[Section 2.4.5: Audit Function 3](#_Toc156920292)

[Section 2.4.6: Control Functions 3](#_Toc156920293)

[Section 2.4.7: Reliability Requirements 3](#_Toc156920294)

[Section 2.4.8: Criticality of the Application 4](#_Toc156920295)

[Section 2.4.9: Safety and Security Considerations 4](#_Toc156920296)

[Section 2.5: Assumptions and Dependencies 4](#_Toc156920297)

[Section 2.6: Apportioning of Requirements 4](#_Toc156920298)

[Section 2.7: Software System Attributes 4](#_Toc156920299)

Section 1: Introduction

## Section 1.1: Purpose

The purpose of this project is to develop a system that collects and stores operational data from Robot DR#1 into a database. This stored data will then be made accessible through a well-defined API, enabling Team 3 to retrieve and utilize the information for the development of a graphical user interface. The goal is to provide a clear, real-time graphical representation of the H-bot’s status and key metrics, thereby offering insights and enhancing situational awareness for all stakeholders interested in monitoring and analyzing the robot’s performance.

## Section 1.2: Scope

[describe the scope of the project]

## Section 1.3: Definitions, Acronyms, and Abbreviations

**Table 1.**

*Definitions, Acronyms, and Abbreviations for the Project*

|  |  |  |
| --- | --- | --- |
|  | Name | Description |
|  | DR | Delta Robot |
|  | API | Application Programming Interface |
|  | MS | Micro Service |
|  | DB | Database |
|  | MQTT | Message Queuing Telemetry Transport |
|  | HMI | Human Machine Interface |
|  | H-bot | Mechanical motion system used in robotics and CNC machines due the arrangement of its drive belts or rods forms an “H” shape |

## Section 1.4: References

1. Provide IEEE formatted references (cite them using [1] [2], etc. IEEE citation method)

## Section 1.5: Overview

This Software Requirements Specification (SRS) document outlines the functional and non-functional requirements for the system designed to collect, store, and serve operational data from Robot DR#1. The document provides a comprehensive description of the system’s intended functionality, interfaces, and performance criteria.

The system’s primary function is to capture real-time data from the robot, persist it in a database, and expose it through a robust API to support the development of a graphical interface by Team 3. This interface will facilitate monitoring and analysis of the H-bot's key operational metrics.

Section 2: Overall Description

## Section 2.1: Product Perspective

The product is designed as a middleware system that acts as an intermediary between Robot DR#1 and Team 3’s graphical interface. It is a critical component within a larger robotic monitoring and control system.

The system collects real-time data from Robot DR#1, stores it in a centralized database, and exposes this data through a RESTful API. Team 3 consumes this API to create graphical representations of the H-bot’s operational status for monitoring purposes.

This product depends on stable network communication with Robot DR#1 and reliable database services. It utilizes MQTT protocol for efficient telemetry data transfer and supports scalable data access through the API.

### Section 2.1.1: User Interfaces

The dashboard displays graphical representations of the H-bot’s status, key metrics, and relevant telemetry information.

Users cannot input or modify data through the dashboard; its sole purpose is to present updated information visually in an easy-to-understand format. Users can interpret and use the displayed data externally as needed.

### Section 2.1.2: Software Interfaces

Start

1. Data Extraction and Publish (DR#1)  
 This step involves extracting data and publishing it. Tagged as ML1 and Research.

2. MQTT Broker  
 Receives the published data and distributes it.

3. Subscribe, Transform, Save  
 This step subscribes to the MQTT messages, transforms the data, and saves it. Tagged as WS2 and TSCode.

4. Create Database  
 Stores the transformed data.

5. Database (DR#2, handled by Team 3)  
 Holds the processed data.

6. Create API  
 Builds an interface to access the database.

7. Dashboard (H-Bot)  
 Displays the data visually using graphs and indicators.

8. API Team 1  
 Supports the dashboard with API services.

End

### Section 2.1.3: Operations

**Data Acquisition Mode** Continuously extracts data from the source such as PLC or OPC UA server. Data is published via MQTT in real-time or at defined intervals.

**Message Handling Mode** MQTT Broker receives published messages. Data consumers such as the transformation service subscribe to relevant topics.

**Data Processing Mode** Subscribed services transform incoming data into a structured format. Cleaned data is stored in a centralized database.

**API Service Mode** Provides endpoints for internal and external applications to access stored data. Supports secure and scalable data queries.

**Visualization Mode** Dashboard interfaces display real-time and historical data. Includes charts, indicators, and system health metrics.

**Maintenance Mode** Allows administrators or development teams to update configuration, monitor logs, or perform health checks without disrupting core functions.

## Section 2.2: Product Functions

1. Data acquisition from industrial sources such as PLCs or OPC UA servers.
2. Real-time data publishing using MQTT protocol.
3. Data transformation and cleaning before storage.
4. Structured storage of processed data in a centralized database.
5. Provision of RESTful API for data access and integration.
6. Visualization of live and historical data through dashboards.
7. System monitoring and support for maintenance activities.

### Section 2.2.1: Function #1

**Section 2.2.1: Data Acquisition**  
 Pulls data from PLCs or OPC UA servers in real-time or on schedule. Supports industrial protocols, adds timestamps, and ensures reliable collection.

**Section 2.2.2: Real-time Data Publishing**  
 Publishes collected data via MQTT. Supports topic structure, QoS levels, and efficient message distribution to subscribers.

**Section 2.2.3: Data Transformation and Cleaning**  
 Subscribes to MQTT topics, applies rules to format and clean data, removes noise or invalid values, and prepares for storage.

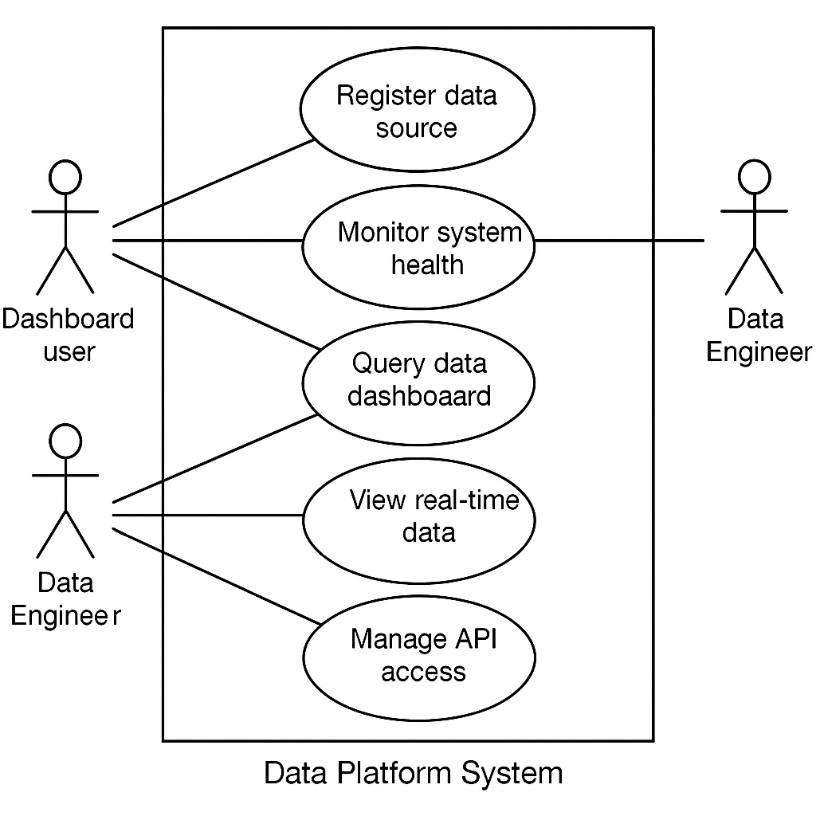
**Section 2.2.4: Structured Storage**  
 Stores processed data in a central database. Schema is optimized for querying, historical tracking, and integration with analytics tools.

**Section 2.2.5: API Provisioning**  
 Offers REST APIs to fetch data from the database. Allows integration with external systems, web apps, and internal dashboards.

**Section 2.2.6: Data Visualization**  
 Displays real-time and historical data using web dashboards. Includes graphs, charts, status indicators, and filtering options.

**Section 2.2.7: System Monitoring and Maintenance**  
 Provides tools for monitoring system health, managing logs, and performing updates. Ensures high availability and smooth operation.

**Figure: Use Case Diagram**



#### Overview:

This use case describes how data is acquired, processed, and presented within the Recycling Robot Visualization system. It covers tasks performed by data engineers and dashboard users, such as registering robot data sources, monitoring system status, visualizing performance, and managing API access.

#### Notes:

* The system collects data from industrial devices like PLCs or OPC UA servers used in the recycling robot infrastructure.
* Real-time communication is handled using MQTT.
* A RESTful API allows integration with other tools or services.
* Dashboards provide real-time of robot activity and system metrics.

#### Actors:

* **Data Engineer:** Sets up robot data feeds, defines transformation logic, monitors the system, and manages API access.
* **Dashboard User:** Interacts with visual dashboards to analyze robot operations, performance, and alerts

#### Preconditions:

* The **Recycling Robot Visualization** system is deployed and fully operational.
* Robot data sources are network-accessible and properly configured.
* Required backend services (MQTT, database, API) are online.

#### Use Case Details

**Table:**

|  |  |
| --- | --- |
| Goal: | To collect, process, and visualize real-time data from recycling robots in order to monitor operations, detect anomalies, and support data-driven decision-making. |
| Actor(s): | * **Data Engineer**: Sets up data sources and configures data flow. * **Dashboard User**: Views real-time and historical performance data of the recycling robots |
| Trigger: | A recycling robot and data source comes online and begins sending data, or a user accesses the dashboard or API to retrieve insights. |
| Success Outcomes: | * Data is successfully acquired from the robot and stored. * The dashboard displays live metrics and trends. * Users can retrieve data via the API. * System operates with no errors or delays. |
| Exceptions: | * Data source is unreachable or sends corrupted data. * MQTT broker or database is offline. * Dashboard fails to load due to backend issues. * Unauthorized access attempt is blocked and logged. |

#### UI Mock-ups:

**Dashboard Home View:**

* The dashboard displays graphical representations of the H-bot’s status, key metrics, and relevant telemetry information.
* Status indicators (On/Off/Idle)

#### Scenario Notes:

* All UI elements are responsive and designed for both desktop and tablet use.
* Real-time updates are powered by MQTT
* Each visualization scenario is backed by data stored in the centralized system, ensuring consistency between real-time and historical views.

## Section 2.3: User Characteristics

### 1. Data Engineers

* **Technical Background**: Users have strong technical expertise in data pipelines, industrial systems, and software integration.
* **Responsibilities**:
  + Configure data acquisition from Robot DR#1 via MQTT or OPC UA.
  + Define transformation logic and ensure data is processed and stored correctly.
  + Monitor system performance and troubleshoot data flow issues.
  + Manage and secure API access and database connectivity.
* **System Interaction**: Use system logs, configuration interfaces, and GitHub for version control and maintenance.

### 2. Dashboard Users

* **Non-Technical to Semi-Technical Background**: May include operators, engineers, analysts, or managers focused on monitoring robot performance.
* **Responsibilities**:
  + Visually analyze H-bot metrics and operational data via the dashboard.
  + Use insights to support operational decisions or maintenance planning.
* **System Interaction**: Primarily read-only access through a web-based dashboard. No direct interaction with APIs or back-end systems.

## Section 2.4: Constraints

These constraints help ensure the dashboard is effective, secure, and user-friendly while meeting the specific needs of monitoring and controlling production data using positional values of the Robot DR1.

1. **Technical Constraints**:
   1. **Data Integration**: The dashboard must integrate with the production system to fetch real-time positional data.
   2. **Real-time Updates**: The dashboard must update positional data in real-time with minimal latency.
2. **Performance Constraints**:
   1. **Response Time**: Data updates on the dashboard should occur within 1 second of changes in the production system.
   2. **Scalability**: The dashboard should handle data from multiple production units simultaneously without performance degradation.
3. **Security Constraints**:
   1. **Access Control**: It must follow Magna’s Policy for access control to differentiate viewer to editor.
   2. **Data Encryption**: Sensitive positional data transmitted between the production system and the dashboard must be encrypted.
4. **Environmental Constraints**:
   1. **Network Connectivity**: The dashboard must function reliably in environments with varying network connectivity.
   2. **Device Compatibility**: It should be accessible from different devices, including desktops, tablets, and smartphones.
5. **Project Constraints**:
   1. **Timeline**: The dashboard must be developed and deployed within the specified project timeline.

### Section 2.4.1: Regulatory policies

The development and operation of the data collection and API system for Robot DR#1 must comply with relevant Canadian regulatory policies and industry standards, including but not limited to:

* **Personal Information Protection and Electronic Documents Act (PIPEDA)**: If any personal or sensitive information is collected, stored, or transmitted by the system, compliance with PIPEDA is required to ensure the privacy and security of personal data.
* **Canadian Standards Association (CSA) Standards**: Compliance with applicable CSA standards related to robotics, automation, and electrical safety may be necessary depending on the system’s operational context.
* **Cybersecurity Guidelines**: Adherence to cybersecurity best practices and standards, such as those recommended by the **Canadian Centre for Cyber Security**, to ensure data confidentiality, integrity, and availability, particularly for API access.
* **Industrial Safety Regulations**: Compliance with workplace safety regulations governed by **Workplace Safety and Insurance Board (WSIB)** or other provincial occupational health and safety laws if the robot operates in an industrial or commercial environment.
* **Internal IT and Security Policies**: The system must align with the client’s internal IT governance policies regarding software development, data handling, and API security.

### Section 2.4.2: Hardware limitations

* **Robot DR#1 Dependency**
* All data collection relies on the operational availability and proper functioning of Robot DR#1’s onboard sensors and interfaces.
* Hardware failures (e.g., communication modules, I/O ports) can interrupt the data stream.
* **PLC / OPC UA Interface Limitations**
* Data acquisition is limited by the polling rate and processing capacity of the connected PLC or OPC UA server.
* High-frequency data collection may be constrained by the source device's hardware throughput.
* **Local Server Constraints**
* The database and MQTT broker are hosted on local infrastructure (e.g., File Server 192.168.0.212).
* System performance is dependent on available CPU, RAM, disk I/O speed, and network bandwidth.
* **Dashboard Display Devices**
* Users may access the dashboard from varying devices (e.g., tablets, smartphones, industrial PCs).
* Device limitations (screen size, performance, browser support) may affect dashboard rendering quality or response time.
* **Network Environment**
* The IoTAcadamy\_EXT4 network must support stable and low-latency communication between all hardware components.
* Unreliable or slow connections can delay data updates or cause MQTT message loss.

### Section 2.4.3: Interfaces to other applications

* Applications must publish/extract/transform data from the correct UNS bucket.
* Data type must not be corrupted or changed throughout the process.

### Section 2.4.4: Parallel Operation

The system supports multiple robots working in parallel. MQTT keeps robot data streams isolated. The backend handles concurrent data processing and dashboard access without performance issues. Real-time updates and API calls can happen simultaneously without conflict.

### Section 2.4.5: Audit Function

Version control and audit tracking for system files and configuration changes will be managed using **GitHub**. Each team member will commit and push their changes to a shared GitHub repository, leveraging GitHub’s built-in audit functions to track who made changes, when, and what was modified.

This approach provides a transparent, reliable audit trail of all revisions, ensuring accountability and enabling easy review of change history. Team members should follow established Git workflows and commit message guidelines to maintain clear records.

### Section 2.4.6: Control Functions

* **Microservice 1 (Data extraction and publish)**: Data Acquisition (OPC to MQTT): This service connects to the robot using opc protocol, collects real-time data, and publishes it an MQTT topic based on UNS standard. It also supports reconnection attempts and data buffering if the MQTT broker is temporarily unavailable or if the OPC connection is down
* **Microservice 2 (Ingestion App):** This service subscribes to the MQTT topic, processes incoming data (filtering, validation and calculations), and stored in database. Any data processing errors are logged, and alerts are raised if persistent issues are detected.
* **Microservice 3 (API Exposure):** This service exposes an HTTP API for external consumers (Such as dashboards other services). It controls access using authentication tokens and logs all incoming requests for traceability.

### Section 2.4.7: Reliability Requirements

These requirements ensure that the intended software functions properly without faults.

1. Uptime: The software must generate data considering key components are functional.
2. Error Rate: The software should have an error rate of 0.01% during normal operation.
3. Data integrity: The software must ensure that no data has been lost or corrupted.
4. Redundancy: The software must limit the number of redundancies, ensuring data congestion for IT/OT components.
5. Load Handling: The software must ensure the deletion of older data to maintain the capacity for recording new data.

### Section 2.4.8: Criticality of the Application

The application plays a critical role in monitoring recycling robot performance and system health. It supports operational decisions, early fault detection, and maintenance planning. Any downtime or failure in the system could delay recycling workflows or lead to undetected issues in robot behavior. High availability and reliability are essential.

### Section 2.4.9: Safety and Security Considerations

This system is designed with safety and security as key priorities to ensure reliable operation and protection of sensitive data.

**Security Requirements:**

* **Data Protection:** All operational data collected from Robot DR#1 will be securely transmitted and stored to prevent unauthorized access, alteration, or loss.
* **Access Control:** Access to the database and API will be restricted through authentication and authorization mechanisms to ensure that only authorized users and systems can retrieve or modify data.
* **API Security:** The API will incorporate security measures.
* **Version Control Security:** All system files and configurations will be stored in a GitHub repository with controlled access permissions. GitHub’s audit functions will be used to monitor changes, ensuring traceability and accountability.

**Safety Considerations:**

* Since the system interfaces with Robot DR#1, care will be taken to avoid disruptions to the robot’s real-time operations. Data collection will be designed to have minimal impact on the robot’s performance and safety.
* Any failures or errors in data collection or API availability will be handled gracefully to prevent cascading effects on dependent systems.

## Section 2.5: Assumptions and Dependencies

**Assumptions**

* The Robot DR#1 is fully operational and capable of providing consistent real-time data over the MQTT protocol.
* All network infrastructure (e.g., IoIAcadamy\_EXT4) is stable and allows uninterrupted communication between system components (MQTT broker, database, API).
* The dashboard users will only require read-only access to data and will not attempt to modify or push data back into the system.
* All teams (Team 1, Team 2, Team 3) will follow defined API contract standards to ensure consistent data communication.

**Dependencies**

* **Team 1**  
   The system depends on Team 1 to deliver and maintain a robust, secure, and documented API layer that the dashboard (developed by Team 2) will use to access processed data from the database.
* **Team 3**   
   Team 2’s functionality relies on Team 3 to design, implement, and maintain the DR#2 database schema. The dashboard's data visualizations require properly structured and available data.
* **MQTT Broker Service**  
   A functioning MQTT broker is essential for message-based data transmission. This system depends on continuous uptime and valid topic publishing from DR#1.
* **External Systems (PLC / OPC UA Server)**  
   The data acquisition process assumes that upstream industrial systems like the PLC or OPC UA server remain accessible and operational.

## Section 2.6: Apportioning of Requirements

1. **Advanced Dashboard Features**
2. Customizable views (e.g., drag-and-drop widgets, user-defined filters).
3. Historical data comparison tools and trend forecasting.
4. Export functionality (e.g., PDF/CSV reports).
5. **Role-Based Access Control (RBAC)**
6. Current dashboard access is read-only and unrestricted.
7. Implementation of RBAC for differentiated permissions (e.g., viewer vs. admin) will be added in a later release.
8. **Edge Deployment Support**
9. While currently designed for cloud/on-prem environments, future iterations may include deployment to edge devices.
10. **Multi-Robot Scaling**
11. The current system supports a single robot (DR#1).
12. Scalability to multiple robots with isolated or aggregated views is planned but not in scope for the current release.
13. **Automated Anomaly Detection**
14. Future versions may include automated fault detection or alerts based on abnormal patterns in robot telemetry data.
15. **Enhanced API Logging and Analytics**
16. Basic audit trails are included via GitHub.
17. Full API analytics and usage monitoring are deferred.

## Section 2.7: Software System Attributes

**AVAILABILITY:**  
 **Must** be available 24/7 during robot operational hours to ensure live monitoring.  
 **Should** include automatic service restart on failure.

**RELIABILITY:**  
 **Must** reliably capture and display data without loss or corruption.  
 **Should** include fallback logging if real-time processing fails.

**SECURITY:**  
 **Must** enforce authentications for all users (at the moment open for all)  
 **Must** restrict API access to authorized clients. (at the moment not secured)  
 **Should** support role-based access control (RBAC).  
 **Could** include audit logs for critical actions.

**MAINTAINABILITY:**  
 **Must** be modular and well-documented to support future updates.  
 **Should** allow configuration changes without full redeployment.  
 **Could** provide a web-based admin interface for maintenance tasks.

**PORTABILITY:**  
 **Should** be deployable on both cloud and on-prem environments using Docker or similar tools.  
 **Could** support deployment on edge devices in future iterations.

**EXTENSIBILITY:**  
 **Should** support adding new data sources or visualization widgets with minimal effort.  
 **Could** include plugin support for custom robot types or metrics.