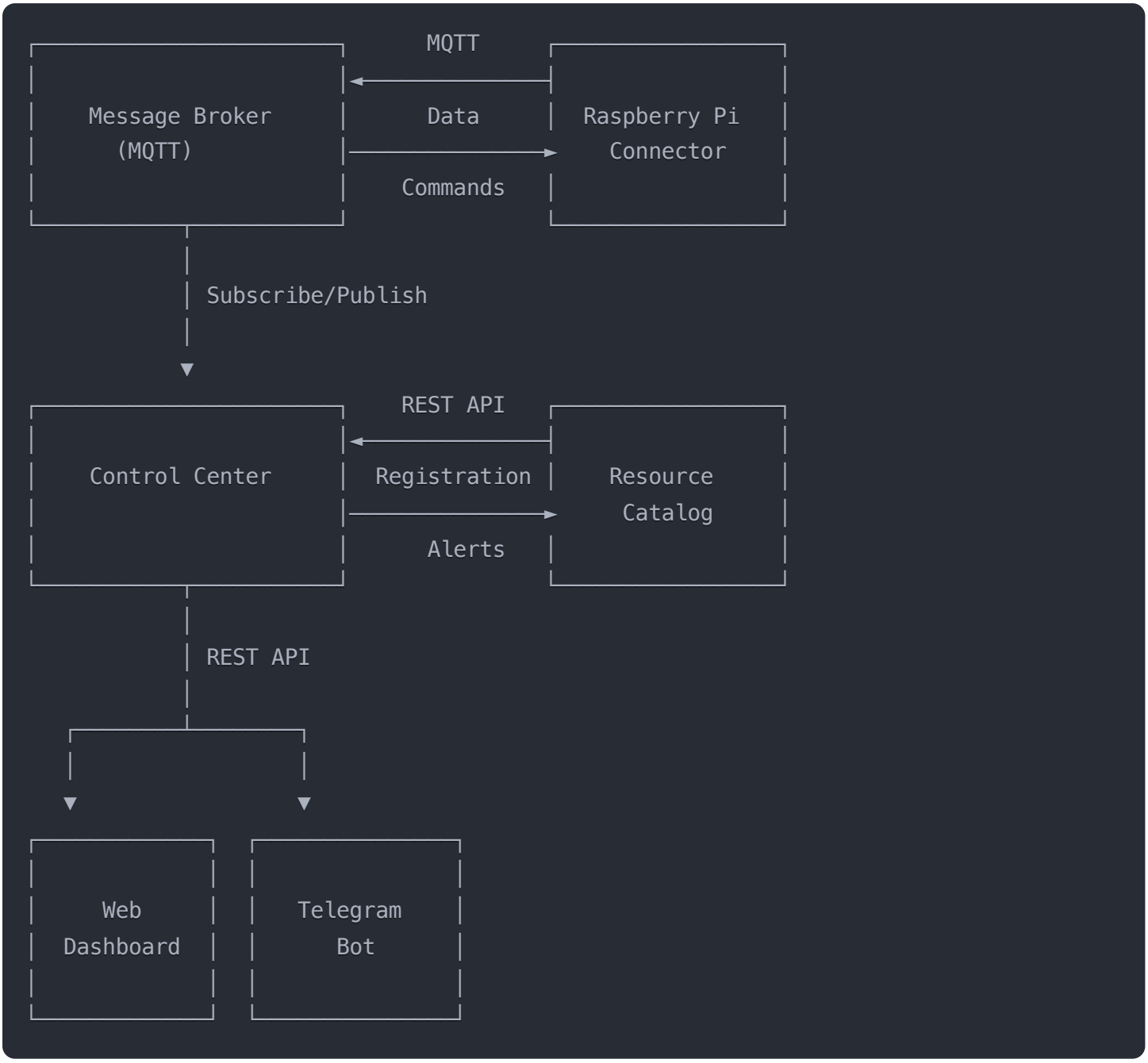
Smart IoT Bolt Control Center Documentation

1. Control Center Overview

Copy



The Control Center is a critical microservice within the Smart IoT Bolt for Pipelines system that serves as the central intelligence and decision-making component. This microservice is responsible for:

The Control Center is a critical microservice within the Smart IoT Bolt for Pipelines system that serves as the central intelligence and decision-making component. This microservice is responsible for:

- Processing real-time sensor data (temperature and pressure)
- Applying control logic based on predefined thresholds
- Automatically controlling valve operations

- Detecting critical conditions and notifying other components
- Providing manual control capabilities via REST API

The Control Center plays a pivotal role in ensuring pipeline safety and operational efficiency by continuously monitoring conditions and taking appropriate actions when necessary.

2. System Architecture Integration

2.1 Communication Protocols

The Control Center implements two primary communication paradigms:

1. **Publish/Subscribe (MQTT)**
   - Subscribes to sensor topics: `/sensor/temperature` and `/sensor/pressure`
   - Publishes control commands to `/actuator/valve`
   - Receives real-time data from sensors via Message Broker
2. **Request/Response (REST)**
   - Registers itself with the Resource/Service Catalog
   - Provides API endpoints for manual control and status monitoring
   - Sends critical situation alerts to the Resource Catalog

2.2 Microservice Dependencies

The Control Center interacts with several components in the system:

- **Message Broker**: Receives sensor data and sends valve commands
- **Resource/Service Catalog**: For service registration and alert notifications
- **Raspberry Pi Connector**: Indirectly controls the valve actuator through MQTT
- **Analytics Microservice**: Complements the Control Center by providing predictive analysis
- **Web Dashboard and Telegram Bot**: Use Control Center's REST API for monitoring and control

3. Core Functionalities

3.1 Threshold-Based Control Logic

The Control Center implements a rule-based control system using predefined thresholds:

- **Pressure Thresholds**: Min (30) and Max (150) by default
- **Temperature Thresholds**: Min (10) and Max (80) by default

When sensor readings exceed these thresholds, the control logic determines the appropriate valve action:

- If pressure or temperature exceeds the maximum threshold → "OPEN" valve

- If both pressure and temperature fall below minimum thresholds → "CLOSE" valve

3.2 Real-Time Monitoring

The Control Center maintains the latest sensor readings and their timestamps in memory, allowing for:

- Immediate evaluation against thresholds
- Current system status reporting
- Historical context for decision-making

3.3 Alert Notification

When critical conditions are detected (thresholds exceeded), the Control Center:

1. Takes immediate action (valve control)
2. Sends an alert notification to the Resource Catalog
3. Includes detailed information about the condition and action taken

These alerts are ultimately propagated to the Web Dashboard and Telegram Bot for user notification.

3.4 Manual Override

The Control Center provides a REST API endpoint for manual valve control, allowing authorized users to:

- Override automatic control
- Open the valve manually
- Close the valve manually

This capability is essential for maintenance activities and emergency situations.

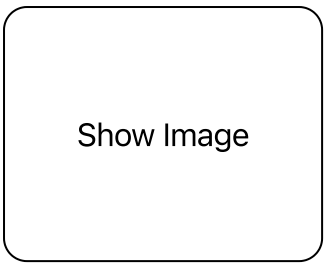
4. Technical Implementation

4.1 Component Structure

The Control Center follows a modular architecture with these main components:

- **MQTTHandler**: Manages MQTT communication
- **ControlService**: Implements the core control logic
- **ThresholdUtils**: Provides threshold evaluation functionality
- **ControlController**: Exposes the REST API endpoints

4.2 Data Flow



1. **Data Ingestion**:
   - MQTT messages containing sensor readings are received
   - Data is parsed and validated
   - Latest readings are stored in memory
2. **Processing**:
   - New readings trigger threshold evaluation
   - Decision is made about valve action
   - Command is generated if needed
3. **Action**:
   - Valve command is published to MQTT
   - Alert is sent if condition is critical
   - Status is updated for API queries

4.3 Error Handling & Resilience

The Control Center implements several resilience mechanisms:

- Graceful handling of MQTT connection failures
- Periodic retry for Resource Catalog registration
- Timeout settings for REST API calls
- Logging of all operations and errors

5. Configuration

The Control Center is highly configurable through environment variables:

- **Service Configuration**: Port, name, etc.
- **MQTT Connection**: Broker host, port, credentials
- **Thresholds**: Min/max values for pressure and temperature

This allows for easy deployment in different environments without code changes.

6. Interaction with Other Microservices

6.1 Raspberry Pi Connector

- The Control Center sends valve commands via MQTT
- The Raspberry Pi Connector executes these commands on the physical valve
- No direct communication exists; all interaction is through the Message Broker

6.2 Analytics Microservice

- While the Control Center handles immediate threshold-based decisions
- The Analytics Microservice complements with predictive analytics
- Together they form a comprehensive control system (reactive + proactive)

6.3 Web Dashboard & Telegram Bot

- These user interfaces retrieve status from the Control Center
- They can send manual control commands through the REST API
- They display alerts generated by the Control Center

7. API Endpoints

7.1 GET /api/control/status

Returns the current status including latest sensor readings and valve recommendation.

Response Example:

json Copy

```
{
  "latest_data": {
    "temperature": 75.3,
    "pressure": 142.7,
    "timestamp": "2025-03-30T17:45:23.456789"
  },
  "valve_recommendation": "OPEN"
}
```

7.2 POST /api/control/command

Allows manual control of the valve.

Request Body:

json Copy

```
{
  "command": "OPEN"
}
```

Response Example:

json Copy

```
{
  "status": "success",
  "message": "Command OPEN sent successfully"
}
```

8. Future Enhancements

Potential improvements for the Control Center include:

1. **Advanced Control Algorithms**: Implementing PID controllers or other advanced control strategies
2. **Machine Learning Integration**: Using ML models for more intelligent decision-making
3. **Multi-valve Support**: Extending to control multiple valves in complex pipeline systems
4. **Enhanced Authentication**: Implementing role-based access control for manual operations