정형화된 패턴분석을 적용한 센서 데이터흐름 감지 및 MQTT 시뮬레이션 시스템 개발

최종원*, 박혜리*, 파이줄라에브 미루자롤**, 오염덕(교신저자)O
*한국교통대학교 소프트웨어학 전공,
**한국교통대학교 교통에너지융합학과,
O한국교통대학교 소프트웨어 전공
e-mail: bmsyg987@gamil.com*, hyeri9918as@naver.com*, Fayzullayevmirjalol@gmail.com**, rdoh@ut.ac.krO

Development of Sensor Data Flow Detection and MQTT Simulation System to apply formalized Pattern Analysis

JongWon Cho*, Hyeri Park*, Fayzullayev mirjalol**, Ryumduck Oh(Corresponding Author)O
*Department of Software, Korea National University of Transportation,
**Dept. of IT and Energy Convergence, Korea National University of Transportation,
ODepartment of Software, Korea National University of Transportation

● 요 약 ●

본 논문에서는 기존 철도 운영 및 관리시에 철도 주변환경으로 부터 발생하는 소음, 진동, 미세먼지 센서에서 다양한 실시간 스트림 데이터를 감지하고 정형화된된 데이터 패턴을 인식하고 분석할 수 있도록 데이터를 구성 및 저장하고 분석된 데이터를 표현할 수 있도록 시각화 지원을 위한 모니터링 시스템 플랫폼을 구현하였다. 데이터 전송을 위해 시리얼 통신 기법을 주로 적용하였으나, 센서와 디바이스의 증가로 인해 시리얼 통신의 한계가 나타났다. 따라서, 본 연구에서는 기존의 아두이노와 서버 간의 직접 통신 방식 대신 라즈베리파이를 도입하여 MQTT Broker(브로커)를 설치하고 통신을 진행하였다. 철도 데이터 모니터링 시스템 플랫폼은 NoSQL 데이터베이스인 MonGoDB와 데이터 시각화할 수 있는 Grafana를 이용하여 구축하였다.

키워드: 사물 인터넷 Iot(Internet of Things), 무선 통신(Wireless Communication), 시리얼 통신(Serial Communication), MQTT(MQ Telemetry Transport), 데이터 전송(Data Transmission), 효율적인 통신(Efficient Communication)

I. Introduction

철도 모형 환경 속 3개의 측정 구역에 설치한 여러 종류의 센서들 수가 점차 늘어나면서, 유선 통신인 시리얼 통신을 사용하게 되었을 경우, 관리와 유지보수가 어려운 단점들이 있었다.

본 연구는 이러한 단점을 극복하기 위해 MQTT(MQ Telemetry Transport) 프로토콜을 도입했다. 더불어, 이 연구에서는 기존의 아두이노와 서버 간의 직접 통신 방법을 대체하여 라즈베리파이를 도입하였고, MQTT Broker(브로커)를 설치하여 통신을 진행하였다. [1]

더불어, NoSQL 데이터베이스인 MongoDB와 데이터 시각화 도구인 Grafana를 활용하여 철도 모니터링 시스템을 구축한다.

새롭게 도입된 MQTT 시스템은 철도 모니터링 시스템의 유지보수 및 관리를 효율적으로 개선하였으며, 해당 프로토콜은 시스템의 확장성과 안정성을 높힐 수 있도록 지원하였다.[2]

본 논문은 시리얼 통신 대신 라즈베리파이를 활용해 철도 모니터링 시스템에서 MonGoDB와 데이터 시각화할 수 있는 Grafana를 이용하여 통합 플랫폼을 구축하였다.

II. Related works

본 논문은 "*Serial vs. Wireless Communication: A Comparative Study*"를 참고[1]하여 시리얼 통신과 무선 통신의 장단점을 비교하고, 특히 시리얼 통신이 무선 통신에 비해 연결 및 관리 유지 보수가 어렵다는 점을 확인하였다.

이를 바탕으로 기존 시스템에서 시리얼 통신을 사용한 센서 데이터 수집 방식을 MQTT 프로토콜을 활용한 무선 통신 시스템으로 변경하

는 과정을 제시한다.

또한 기존의 서버에서 직접 시리얼로 받아와 웹에 센서 데이터를 실시간으로 출력하는 방식을 대체하여 Mongodb를 사용해 센서 데이터를 적재하였고, Grafana를 활용해 철도 데이터 시각화를 구현하였다.

본 논문에서는 초기 시리얼 포트(직접 통신)를 이용한 센서 데이터 수집 시스템을 MQTT(무선 통신) 프로토콜을 활용한 새로운 시스템으로 변경하는 작업을 진행하였다.

또한 기존의 서버에서 직접 시리얼로 받아와 웹에 센서 데이터를 실시간으로 출력하는 방식을 대체하여 Mongodb를 사용해 센서 데이터를 적재하였고, Grafana를 활용해 철도 데이터 시각화를 구현하였다.

1. The Architecture of Data Collection System

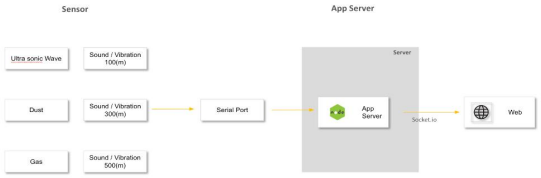

Fig. 1. Initial System Architecture

초기 철도 데이터 수집 시스템은 시리얼 통신을 활용하여 데이터를 수집하고 전송하는 방식이 도입되었다. Fig 1은 초기 시스템의 구조를 나타내는 아키텍처로, 해당 구조에서는 시리얼 통신을 통해 시스템 환경을 빠르게 구현할 수 있었다.

이러한 환경 구축은 단기간 내의 철도 시스템을 빠르게 가동할 수 있었으나, 현재 측정 구역과 측정 거리가 세분화되면서 센서의 종류와 수량이 증가함에 따라 시리얼 통신을 사용한 데이터 송수신량이 방대해지는 현상이 발생하게 되었다.

이에 따라 데이터 송수신 간의 오버헤드와 트래픽 문제가 발생하면서, 초기에 비해 통신 속도가 느려지는 성능 저하 현상이 발생하였다.

또한 초기 시스템에서는 서버에서 직접 센서 데이터를 시리얼로 수신하여 웹에 실시간으로 출력하는 방식을 채택하였으나, 이는 데이터의 기록과 통계를 추적하기 어려워 사용자가 데이터를 더욱 효과적으로 관리하고 분석할 수 없었다. 이러한 기존 시스템의 한계를 극복하고 시스템을 향상하기 위해 MQTT 프로토콜을 도입하여 새로운 시스템 구조를 적용하였다.

2. Introduction of MQTT and its Necessity

MQTT(Message Queuing Telemertry Transport)는 ISO 표준의 메시지 Publish-Subscribe 형식을 사용하는 경량화 메시징 프로토콜이다. [2]

IoT 모듈과 같은 소형 장치는 무선 네트워크를 활용하기 때문에 유선 네트워크보다는 불안정한 통신 대역폭을 가진다. MQTT 프로토콜은 이러한 환경을 고려하여 최적화된 Push 기반 프로토콜로, 낮은 전력 상황과 불안정한 트래픽으로 통신이 가능하며 IoT, M2M등에서 사용하기 용이하다.[3]

MQTT는 QoS(Quality of Service)를 통해 안정적인 메시지 전송이 가능하도록 설계되어 있다. QoS는 메시지의 전달 확실성에 대한 수준을 나타내며, 다양한 요구사항에 따라 0부터 2까지의 세 가지 레벨로 설정이 가능하다.

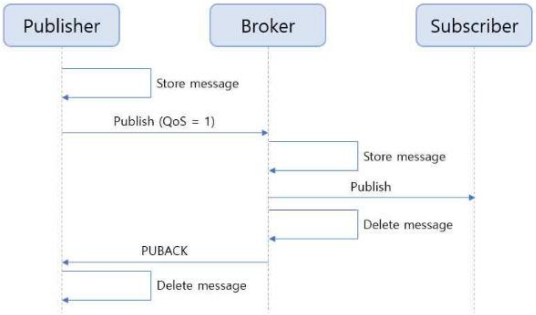

Fig. 2. Flowchart of MQTT Protocol QoS 1 Communication

Fig 2는 속도와 안정성 간의 균형을 이룰 수 있는 QoS 1에 대한 통신 흐름도이다. MQTT의 작동 방식은 Client-Broker-Client 구조를 따르고 있다. 해당 구조에서 Client는 Publish/Subscriber로 구분되고, 데이터를 발행하고 받는 역할을 하며, Broker는 중앙 서버 역할을 수행한다.[4]

모든 패킷은 Broker를 통해 전송되며, 패킷의 목적지는 Subsciber의 Topic 구독 여부에 따라 동적으로 결정된다. 이로써 1:1 통신과 1:N/N:1의 통신을 쉽게 구현할 수 있다.

III. The Railway System Platform

1. The MQTT Protocol

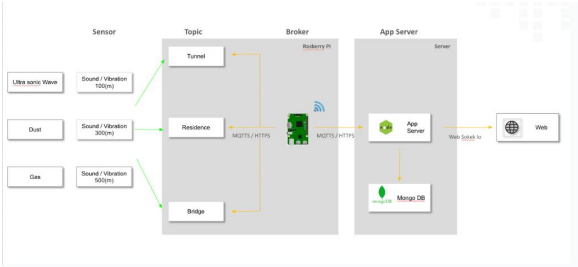

Fig. 3. Revised Structure Using MQTT Protocol

MQTT는 경량 프로토콜로서, 데이터를 효율적으로 전송하면서도 오버헤드를 최소화하는 특성을 갖추고 있다. 라즈베리파이는 해당 아키텍처에서 브로커 역할을 수행하며, 데이터의 안정적인 전달을 담당하고 있다.[2]

Fig. 4. MQTT Broker Data Transmission Log

실제로 라즈베리파이에서 MQTT를 활용한 데이터 수신을 확인하기 위해, ***mosquitto_sub -t [토픽] -h 브로커 주소*** 명령어를 통해 ***[토픽]***에 게시된 메시지를 구독하고, 해당 메시지가 라즈베리파이로 수신되면 Fig 4와 같은 화면이 출력된다.

이를 통해 아두이노에서 발생한 실시간 시리얼 데이터들이 라즈베리파이를 통해 JSON 형식의 데이터로 변환되어 안정적으로 전송되는 것을 확인할 수 있다.

MQTT 프로토콜을 적용한 환경을 통해, 데이터의 빠른 전송과 효율적인 통신을 실현할 수 있었고, 라즈베리파이와 같은 마이크로 pc를 사용하여 서버 과부하 저감 및 데이터 효율적 관리 및 기록, 분산화된 시스템 구축 등 시스템을 개선할 수 있었다.

또한 서버에서 직접 센서 데이터를 시리얼로 수신하여 웹에 실시간으로 출력하는 기존 방식은 데이터의 기록과 통계 추적 및 사용자의 데이터 관리 및 분석이 어려워, MongoDB를 도입하여 라즈베리파이를 통해 변환된 JSON 형식의 센서 데이터들을 저장하고 관리하는 시스템을 구축하였다.

MongoDB 환경을 도입함으로써, 실시간으로 다양한 형태의 데이터를 효과적으로 저장하고 검색할 수 있게 되었고, 데이터의 기록과 추적 또한 가능하게 되었다.

2. Introduction of MongoDB and Grafana for Data Visualization

아두이노 센서에서 수신된 데이터를 MongoDB에 적재하는 과정과 데이터 시각화를 위해 MongoDB와 Grafana를 통합하는 과정을 설명하고자 한다.

2.1 MongoDB Data Storage Process

```python
def save_to_sensor_mongodb(topic, data):
    client = pymongo.MongoClient("mongodb://localhost:27017")  # MongoDB 서버 주소와 포트에 따라 수정
    try:
        # 데이터베이스 및 컬렉션 선택 (your_database_name, your_collection_name을 사용하지 않고 topic을 이용)
        db = client['mydatabase']  # 데이터베이스 이름을 topic으로 설정
        collection = db[topic]  # 컬렉션 이름도 topic으로 설정

        # 데이터를 MongoDB에 삽입
        collection.insert_one(data)

    except Exception as e:
        print(f"Error while saving to MongoDB: {str(e)}")

# MQTT 메시지 수신 핸들러
1개의 사용 위치  ± choejong-won 1 *
def on_message(client, userdata, message):
    payload = message.payload.decode('utf-8')
    topic = message.topic
    print(f"Received MQTT message on topic '{topic}': {payload}")

    # MQTT 메시지 패키징 및 처리
    pre_processed_payload = pre_process_message(topic, payload)
    print(f"Pre Processed MQTT message: {pre_processed_payload}")
```

```python
def pre_process_message(topic, payload):
    # 메시지 처리 로직을 여기에 추가하세요
    message_dict = eval(payload)
    timestamp = int(time())  # 현재 시간을 초 단위로 가져옴
    message_dict['timestamp'] = timestamp  # 타임스탬프 추가

    return message_dict
```

Fig. 5. Code for Data Preprocessing and MongoDB Load

Fig 5 코드와 같이 서버는 센서 데이터 측정 시간을 ('time_stamp') 추가하여 실시간으로 토픽에 송신 중인 센서 데이터를 MongoDB에 저장한다.

2.2 The structure of MongoDB

NoSQL 데이터베이스인 MongoDB는 동적 스키마를 통해 데이터의 구조를 사전에 정의하지 않고도 유연하게 다룰 수 있다. 이는 센서 데이터와 같이 동적으로 변하는 데이터 구조에 적합하다. MongoDB는 데이터의 형식이나 구조를 미리 정의하지 않아도 되기 때문에 신속한 개발 및 유연성이 요구되는 환경에서 특히 효과적이다.

SQL 데이터베이스와 비교할 때, NoSQL의 동적 스키마는 새로운 필드나 구조의 추가, 수정, 삭제가 용이하다는 장점이 있다. 이는 센서 데이터의 형식이나 측정 항목이 늘어날 때 빠른 대응이 가능하다는 것을 의미하며, 측정 기기의 업그레이드나 새로운 센서의 추가 등이 발생할 때 데이터베이스를 변경하기 위한 번거로움을 최소화한다.

2.3 Real-time Data Visualization with Grafana

MongoDB에 저장되는 데이터는 Grafana를 사용하여 시각화 작업을 진행하였다. MongoDB가 설치되어 있는 서버 공간에 Grafana를 설치한 후, Grafana와 MongoDB를 연동하였다.

Fig. 6. Data Validation in the MongoDB

기본적으로 Grafana의 플러그인에는 MongoDB가 없으므로 외부에서 MongoDB 플러그인을 설치한 후 연동을 진행한다. Grafana에서 데이터베이스 쿼리를 이용하여 원하는 데이터를 분류한 후, 시계열 특성을 갖는 필드 ***time_stamp***를 기준으로 데이터를 확인하면 데이터 변화량을 시각적으로 확인한다.

IV. Conclusions

본 논문에서는 초기 시리얼 포트(직접 통신)에서 MQTT(무선 통신) 프로토콜을 활용한 새로운 시스템으로 변경하는 작업과 MongoDB 와 Grafana를 활용한 모니터링을 할 수 있는 시스템을 구축하였다.

해당 시스템을 통해 센서의 동작 상태 및 이상 여부를 직관적으로 파악할 수 있게 되었으며, 센서 관리 및 시스템 모니터링에 대한 효율성을 향상시켰다. 또한, MQTT 프로토콜을 통한 데이터 송수신은 철도에서 발생하는 데이터를 신속하게 수집하고, MongoDB의 도입으로 JSON 형식의 철도 센서 데이터들을 저장하고 관리할 수 있었다.

Grafana의 시각화를 통해 철도에서 발생하는 소음 및 진동 발생 패턴을 명확히 확인할 수 있었다. 앞으로 해당 패턴을 활용하여 후속 연구를 텍스트 데이터 뿐만 아니라 시각적인 자료를 통해 더 직관적이고 효과적인 정보 전달을 목표로 한다.

Grafana의 시각적 자료를 현재 동일 연구에서 연구 진행 중인 웹페이지에 적용하여 웹페이지를 이용하는 사용자들이 데이터를 보다 쉽게 이해하고 분석할 수 있다.

이를 통해 철도 데이터 연구 결과를 보다 명확하게 시각화하여 저공함으로써, 철도에서 발생하는 소음 및 진동의 발생 패턴을 웹페이지로 제공하는 서비스를 제공할 수 있다.

ACKNOWLEDGMENT

2023년 한국교통대학교 지원을 받아 수행하였음.

REFERENCES

[1] Shlomi Arnon외 3인, “A comparative study of wireless communication network configurations for medical applications”, IEEE Wireless Communications, pp. 56-61, March 2003

[2] 민경현, “Node js MQTT 및 MongoDB를 이용한 효율적이고 유연한 센서 리액션 시스템 개발”, 한국정보처리학회 2016년도 추계학술발표대회, pp. 797-800, 2016

[3] 배동규외 2인. “열그리드 모니터링을 위한 MQTT 기반의 센서 네트워크 구축 방법 및 시각화”, 한국통신학회, 2021

[4] 김한기, “InfluxDB와 Grafana를 이용한 EDISON 플랫폼 모니터링 대시보드 설계 및 구현”, 정보과학회, 2020