Journal of Knowledge Information Technology and Systems(JKITS), Vol. 14, No. 3, pp. 257~266, June 2019
https://doi.org/10.34163/jkits.2019.14.3.005

Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

Implement of Big Data Collecting System Based on Distributed Messaging(BDCS-DM) Using the Rasberry Pi

Sun-Young Heo*

Online Lifelong Education Institute, Korea University of Technology and Education

ABSTRACT

Recently, as the Smart Factories become an issue, there is an active movement to construct CPS by collecting and analyzing PLC data at factories that already use pre-built systems such as MES, SCADA, and PLC. However, in order to read the PLC data from the pre-built MES middleware and transfer it to Hadoop, the pre-built MES middleware developer must input the labor, and it takes a lot of investment and time to collect various data from the PLC and store it in the big data storage. Due to these problems, SMEs have a lot of difficulties in making pre-built factories into smart factories. In this paper, we have designed and implemented Big Data Collecting System based on Distributed Messaging(BDCS-DM) that can read the PLC data transmitted through Modbus of pre-built MES system at low cost using Rasberry Pi board. Through simulation, we confirmed that data collection performance is improved by installing a Rasberry Pi board, and have come to the conclusion that three Rasberry Pi board can be added. Also, in order to guarantee the performance for PLC data collection per 1ms cycle, we could see that effective to transmit 10 points per 10ms cycle in the Ethernet environment. Implemented system is expected to be possible to apply on not only the smart factory but also any fields using Big Data and further researches will be made for possible application on other fields.

KEYWORDS: Smart factory, Big data collecting systems, IoT, CPS, Distributed messaging

ARTICLE INFO: Received 16 May 2019, Revised 28 May 2019, Accepted 7 June 2019.

*Corresponding author is with the Online Lifelong Education Institute, Korea University of Technology and Education, Tel : +82 41 560 1484, Fax : +82 41 560 1484.
E-mail address : hsysj119@koreatech.ac.kr

1. 서 론

CPS(Cyber-Physical System) 기반의 '스마트 팩토리'와 인더스트리 4.0(Industry 4.0)은 개인화된 고객의 요구사항을 반영한 대량 고객 맞춤화, 개인 맞춤형 제품 제조 시대로 그 흐름을 변화시켰다[1-3]. CPS는 서로 인터넷 상의 서비스와 연결되어 있는 다양한 센서를 통해 수집된 데이터를 기반으로 제조 현장의 설비 간 네트워크에서부터 설계, 운영에 관련된 최적화된 의사결정을 통합할 수 있어야 한다[4-6].

현재, CPS 기반의 스마트 팩토리 구축 관련 연구가 활발히 진행 중이다. [7]의 연구에서는 큰 크기의 파일처리에 적합하도록 설계되어 있는 Hadoop을 기반으로 작은 파일들을 효율적으로 처리하기 위한 병합 방법을, [8]의 연구에서는 자동화 시스템을 운반하는 다중 경로를 기계 학습 알고리즘을 이용하여 조작하는 알고리즘을, [9]의 연구에서는 스마트 팩토리 구축에 수반되어야 하는 것이 유선과 무선 네트워크 환경이 서로 공존임과 무선 네트워크상에서 발생하는 문제점 해결을 위한 연구가 빠르게 진행되어야 함을 피력하였다. 또한, [10]의 연구에서는 공정 데이터들을 고속의 빅 데이터 처리를 위한 인-메모리 데이터 그리드를 이용한 시스템을, [11]의 연구에서는 제조라인에서 발생하는 이상상황을 기계 학습 기반으로 진단한 후, 제조라인을 재구성하는 함으로써 제조라인의 신뢰성을 향상하고자 하였다.

본 연구에 앞서 수행한 선행 프로젝트로 MES, SCADA, PLC 등의 시스템을 사용하고 있는 기존 공장에서 CPS 구축을 위하여 MES 미들웨어에서 PLC 데이터를 읽어와 하둡(Hadoop)에 전송하는 프로젝트를 진행했었다[12-14]. 그 결과, MES 미들웨어로부터 PLC 데이터를 읽는 주기를 ms 단위로 할 경우 MES 시스템 운영에 영향을 주어서 주기를 1초 제한하여 처리할 수밖에 없었고, 기존 MES 미들웨어 개발업체의 인력이 투입되어야 하는 문제로 인하여, 기존의 공장들이 스마트 팩토리화하는데 어려움을 겪고 있다. 이에 본 논문에서는 MES 시스템과 무관하게 PLC 데이터를 저비용으로 수집할 수 있도록 라즈베리파이(Rasberry Pi) 보드를 이용한 Kafka 기반 빅 데이터 수집 시스템(BDCS-DM)을 설계 및 구현하였다.

본 논문에서는 성능 목표를 1ms 단위로 실시간 수집하는 것이었으나, 네트워크의 레이턴시 제약으로 1ms 단위의 이벤트 구현은 불가능하여 1ms당 10개의 데이터를 전송하는 것으로 목표를 수정하였다.

우선 PLC가 설치되어 있는 곳에 라즈베리파이 보드를 같은 네트워크에 연결하고 라즈베리파이 보드에서 직접 PLC의 데이터를 읽어와 Kafka를 통해 하둡 시스템에 데이터를 전송하는 PLC 데이터 수집 장치를 설계 및 구현한 후 최적의 데이터 수집 조건을 도출하기 위해 데이터 전송 네트워크 환경이 유선일 때와 무선일 때로 나누어 1차 테스트를 수행하였다. 1차 테스트 결과와 PLC Emulator를 라즈베리파이 보드에 구현한 것과 Kafka 서버에 구현해 놓은 PLC Emulator의 성능을 비교 및 분석한 후 1ms 주기의 PLC 데이터 수집 성능을 보장하기 위해한 동일한 환경에서 최적의 Data Agent와 Data Point 수를 구하기 위한 테스트를 병행하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 BDCS-DM의 설계 및 구현 후 테스트 결과에 대해서 살펴보고, 제 3장에서는 결론을 기술한다.

2. BDCS-DM의 설계 및 구현

2.1 BDCS-DM의 개발 환경

본 논문에서는 기 구축된 MES, SCADA, PLC 등의 시스템에 영향을 주지 않고 저비용으로 신속하게 스마트한 공장으로 재구축할 수 있도록 하기 위한 BDCS-DM를 설계 및 구축하였다.

표 1. BDCS-DM 개발 환경
Table 1. Environment for developing BDCS-DM

구분	개발 환경
PLC 모듈 개발	. H/W : 라즈베리파이 3 . OS : 라즈비안(Raspbian) 리눅스 . PLC Modbus 서버 : 간이 PLC 서버 구현 . 서버 간 데이터 이중화 : Zookeeper
빅 데이터 서버 구축	. Kafka Server : 분산 메시징 큐 기반의 데이터 수집 서버 . OpenTSDB : 시계열 DBMS 서버 . Grafana : 데이터 분석/모니터링 시각화 툴 . Hadoop HDFS : 빅 데이터 파일 시스템

BDCS-DM은 <표 1>에서 알 수 있듯이, 기존 시스템으로부터 PLC 데이터를 읽어오기 위해 저가의 라즈베리파이 보드에 데이터 수집 장치를 구현하고, 라즈베리파이 보드에 의해 수집된 데이터를 한 곳에 모아 분석하기 위해 분산 메시지 기반의 빅 데이터 수집 서버인 Kafka와 하둡, openTSDB를 이용하여 개발 환경을 구축하였다. 데이터 시각화를 위해서는 Grafana를 선택하였기에 Grafana와 가장 호환이 잘 되는 openTSDB를 데이터 저장 서버로 선택하였다. 개발 언어로는 Python을 사용하였으며, PLC 데이터의 안정적인 수집을 위해 Apache Zookeeper를 이용하여 데이터를 이중화하도록 구축하였다[15-17].

2.2 BDCS-DM의 구현

1) PLC Emulator(라즈베리파이 보드)

본 논문에서는 라즈베리파이 보드를 2가지 용도로 사용한다. 하나는 산업 현장의 PLC 시스템 역할을 대신하기 위한 PLC Emulator 개발 용도이고, 다른 하나는 PLC 시스템으로부터 발생한 데이터를 수집 시스템 개발 용도이다. 실제 산업 현장에서는 산업 장비 간 발생하는 데이터를 PCL Modbus TCP/IP를 이용하여 주고받고 있다. 그러나 본 논문에서는 실제 장비를 연결하여 테스트할 수 없으므로, PLC 시스템과 동일한 기능을 갖도록 PLC Emulator를 라즈베리파이 보드를 이용하여 구현하였다.

PLC Emulator의 PLC DataSet Set Client는 기존 산업 현장에서 발생하는 PLC 데이터 대신 sin 함수를 사용하여 임의의 데이터를 발생키고, timestamp를 이용하여 일정한 간격으로 발생 데이터를 PLC Modbus Server에 기록하며, 발생 데이터는 Modbus TCP/IP 프로토콜을 통해 송신되도록 구현하였다.

이때, PLC Modbus Server는 PLC Emulator에 연결된 장치들과 Modbus TCP/IP 프로토콜을 통해 송수신하게 된다. <그림 1>은 이를 도식화한 것이다.

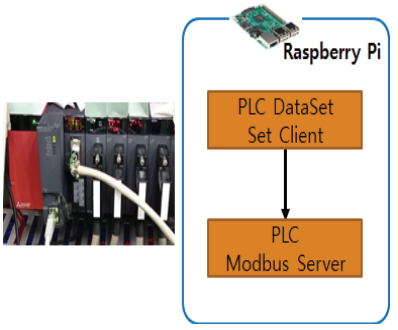

그림 1. PLC Emulator의 아키텍처
Figure 1. Architecture of PLC Emulator

2) PLC Transfer Agent

PLC Transfer Agent의 구성 모듈 중의 하나인 PLC DataSet Get Client는 PLC Emulator의 PLC Modbus Server로부터 전송된 PLC 데이터를 수신 받아 수집된 데이터를 한 저장소에 모아 분석하기 위한 빅 데이터 서버의 Kafka Broker에게 데이터를 전달해야 하는데, 수신 데이터 타입과 openTSDB에서 처리 가능한 데이터 타입이 일치하지 않는 문제가 있다. 그래서 Message Encoding 모듈을 통해 openTSDB에서 처리 가능한 JSON 타입으로 변환한 후 변환된 데이터를 Kafka Producer 모듈로 전달한 후 빅 데이

터 서버의 Kafka Broker에게 전송하도록 구현하였다.

또한, PLC DataSet Get Client 모듈은 timestamp를 활용하여, 동일한 에이전트가 동일한 시점에 이전 기록 PLC 데이터와 동일한 PLC 데이터를 빅 데이터 서버에 기록하는 것을 방지하도록 구현하였다. PLC DataSet Get Client 모듈도 Modbus TCP/IP 프로토콜을 통해 PLC Modbus Server에 설정한 간격과 동일한 Point의 수만큼 데이터를 읽도록 하는 기능을 구현하였다. <그림 2>는 이를 도식화한 것이다.

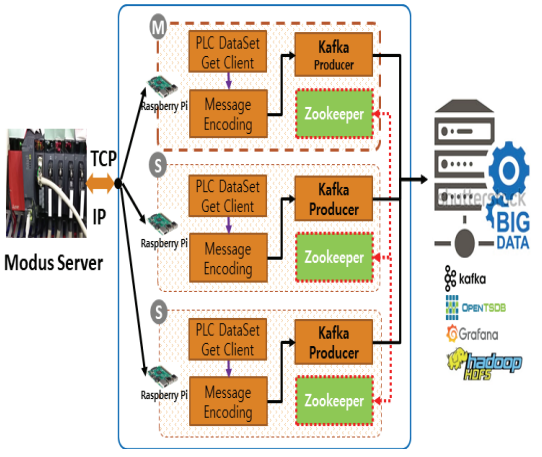

그림 2. PLC Transfer Agent의 아키텍처
Figure 2. Architecture of PLC Transfer Agent

안정적이고 빠른 데이터 수집을 위하여 3개의 라즈베리파이 보드에 각각 PLC Transfer Agent를 구현하고 Apache Zookeeper를 설치한 후, 1개의 마스터(Master: M)와 2개의 슬레이브(Slave: S)로 역할을 분담하였으며, Apache Zookeeper의 NameNode를 이중화함으로써 각 노드의 모니터링을 통한 안정적인 데이터 수집이 가능하도록 구축하였다.

3) 데이터의 이중화

데이터의 안정적인 수집을 위하여 여러 대의 서버가 동시에 데이터 처리에 있어 빠른 속도를 보장하는 하둡 분산 파일 시스템(HDFS)의 메타 정보를 메모리에 유지하고 모든 HDFS 클라이언트의 트랜잭션을 EditLog라는 파일에 수록하는 NameNode를 실제 기능을 수행하는 Active NameNode와 Active NameNode 장애 발생 시 Active NameNode의 역할을 수행하는 Standby NameNode로 이중화하여 구현하였다.

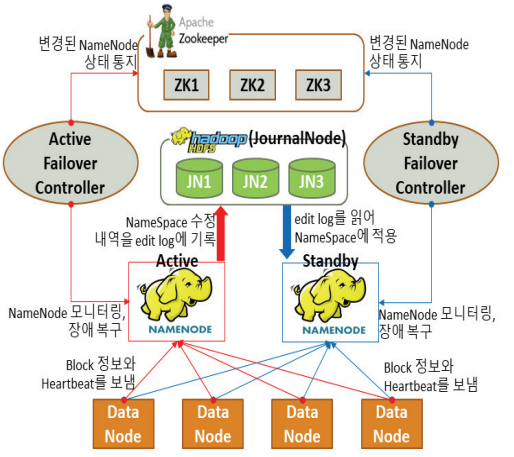

그림 3. 데이터 이중화를 위한 빅 데이터 서버 아키텍처
Figure 3. Big Data Server Architecture for Data Mirroring

<그림 3>에서 알 수 있듯이, Active NameNode에 장애가 발생했을 때 Active NameNode에 존재하는 정보를 어떻게 Standby NameNode가 가져오고, Standby NameNode가 Active NameNode의 장애를 어떻게 찾아내고 Standby NameNode가 Active NameNode로 변경되었을 때 이 두 서버가 Active NameNode인 생태가 발생하는 것을 어떻게 방지할 것인가가 NameNode의 이중화를 구성하는데 가장 골치 아픈 문제이다. 이와 같은 문제를 해결하기 위하여 edits 정보(파일 시스템의 journaling 정보)를 저장하고 공유하는 기능을 수행하는 JournamNode를 이용하여 저장하도록 하는 한편, Failover 처리는 Apache Zookeeper를 이용한 자동

장애 인지 처리 방법을 사용하였다.

Failover Controller는 Apache Zookeeper에게 변경된 NameNode의 상태 정보를 통지하고 NameNode의 상태를 모니터링 하여 장애 발생 시 이를 감지하고 Standby NameNode를 Active로 전환 및 복구하는 역할을 한다. 각 NodeName은 JournalNode의 Quorum을 통하여 NodeName의 상태를 공유하고 Standby NameNode는 1초를 주기로 JournalNode로부터 이전에 받은 EditLog의 정보를 받아 메모리의 파일 시스템 구조에 반영한다.

4) PLC 데이터 수집 및 시각화

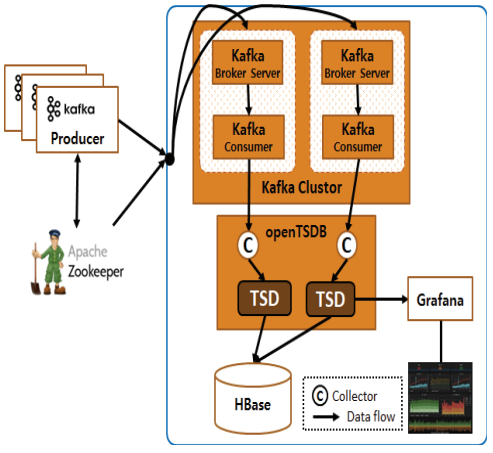

그림 4. 데이터 수집 및 시각화를 위한 아키텍처
Figure 4. Architecture for Collecting and Visualizing data

<그림 4>에서 보여주는 것과 같이, Kafka 서버로 전달된 PLC 데이터는 Kafka의 메시지 큐에 저장된다. Kafka의 메시지 큐에 저장된 상태로는 시각화하기 어려움이 있으므로, Kafka Consumer API를 이용하여 Kafka의 메시지 큐에 저장된 데이터를 읽어서 시계열 DB인 OpenTSDB에 저장한다. 시계열 데이터베이스의 스토리지 계층에서 데이터를 로드하고 액세스하기 위한 여러 가지 협력 구성 요소를 가지고 있는 OpenTSDB는 HBase를 사용하며 타임 시리즈 데이터를 가져와서 저장하며, HBase 스키마는 저장 공간을 줄이기 위해 비슷한 타임 시리즈를 빠르게 집약하는 것에 대해 최적화되어 있다.

OpenTSDB는 REST 인터페이스를 사용하여 JSON 형식의 집계 데이터를 읽고 자체 시각화를 생성하므로, 다양한 back-end 데이터 소스를 기반으로 시계열 데이터와 매트릭스 정보를 보여주는 Grafana 대시보드는 OpenTSDB로부터 데이터를 가져다 바로 시각화하여 보여줄 수 있다. 그래서 본 연구에서는 시각화를 위해서 OpenTSDB와 Grafana를 함께 연동하여 구현하였다. Grafana 대시보드로 Graphite, Elasticsearch, OpenTSDB, Prometheus, InfluxDB, Cloudwatch 등을 데이터 소스로 이용할 수 있다.

Kafka는 PLC Transfer Agent로부터 전달된 JSON 형태의 PLC 데이터를 고속 및 안정적 데이터 전달을 위하여 2개의 분산 서버로 구축하였으며, 클러스터의 개수가 너무 많으면 문제가 발생했을 때 데이터를 복구하는데 많은 시간이 걸릴 수 있으므로, 일단은 3개의 클러스터를 만들고, 하나의 클러스터는 1개의 Topic, 4개의 Partitions, 2개의 Replicas 로 구성된 “Single node-Multiple Broker Cluster” 유형으로 설정하였다. 적정한 클러스터 환경은 실제 생산 공정 라인에 연결하여 테스트하면서 성능에 문제가 생겼을 때마다 개수를 조정하면서 가장 적정한 수집 환경을 확정지을 예정이다.

2.3 성능 테스트 결과

본 시스템의 성능 분석을 위하여 네트워크 환경이 유선일 때와 무선일 때를 구분하여 테스트하였고, 라즈베리파이 보드에 구현한 PLC Emulator와 Kafka 서버에 구현해 놓은 고속의 PLC Emulator의 성능 비교를 위하여 동일한 환경에서 PLC DataSet Get/Get 주기 및 PLC Data Agent와 PLC Data Point 수를 조정하면서 테스트를 진행하였다.

1) 단일 노드 환경에서의 성능 테스트 결과

1 ms당 10개의 데이터를 전송 성능을 단일 노드이면서 Ethernet 환경에서 먼저 테스트한 결과, 어떤 조건에서도 데이터 손실이 발생하지 않았다. 그래서 WiFi 환경에서의 최적의 전송환경을 알아보기 위하여 다음과 같은 테스트를 진행하였다. 먼저, WiFi 환경에서 PLC DataSet Set 주기가 0.1초인 조건에 가장 적정한 PLC DataSet Get 주기를 알아보기 위한 테스트와 WiFi 환경에서는 Kafka 서버에 설치한 고속의 PLC Emulator와 라즈베리파이 보드에 설치한 저속의 PLC Emulator 간의 데이터 전송 성능 비교를 위하여 각 테스트 환경에 따른 데이터 손실률은 동일한 구간에 대한 데이터 손실 정도를 비교하였다.

표 2. WiFi, PLC DataSet Set(0.1초) 주기에서 테스트
Table 2. Tests with WiFi, a DataSet Set:0.1 sec

비교 항목	테스트 환경			
PLC Emulator	Rasberry Pi		Kafka 서버	
DataSet Get	0.005초	0.05초	0.005초	0.05초
PLC Data Agent	2대	2대	2대	2대
PLC Data Point	5point	5point	5point	5point
최소 손실구간	0.3초	0.1초	0.3초	0.1초
최대 손실구간	0.9초	0.9초	0.7초	0.7초
전송 데이터	모두손실	부분손실	부분손실	부분손실
손실 빈도수	19개	7개	11개	6개

<표 2>과 <그림 5>에서 알 수 있듯이, WiFi 환경이라는 특수한 환경 때문인지 라즈베리파이 보드에 설치된 PLC Emulator의 경우 Kafka 서버에 설치된 고속의 PLC Emulator에 비해 데이터 손실 빈도수가 높게 나타났고, DataSet Get 주기를 빠르게 할수록 두 PLC Emulator 간의 데이터 손실 빈도수는 확연한 차이를 보였다.

PLC DataSet Set 주기가 01초이면서 PLC DataSet Get 주기를 0.005초, 0.05초로 했을 때로 구분하여 테스트한 결과, PLC DataSet Get 주기를 0.005초로 했을 때보다 0.05초로 했을 때가 라즈베리파이 보드와 Kafka 서버 모두 데이터 손실 빈도수가 감소하였으며, 라즈베리파이 보드의 경우는 최소 손실 구간이 좁혀졌고 Kafka 서버의 경우는 최대 손실 구간이 좁혀진 것을 확인할 수 있었다. 따라서 PLC DataSet Get 주기는 0.005초보다는 0.05초가 적정하다는 결론을 얻을 수 있었다.

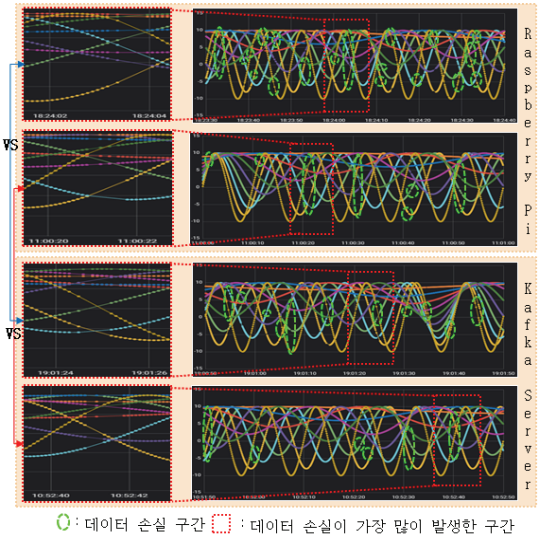

그림 5. 데이터의 손실 빈도수 비교
Figure 5. Comparison of data loss frequencies

1차 테스트에서 PLC DataSet Get 주기가 0.1초이면서 PLC DataSet Get 주기만 0.05초인 환경에서의 데이터 손실 빈도수 및 데이터 최대, 최소 손실 구간이 더 좁은 것으로 나타났으므로 2차 테스트는 동일한 환경에서 적정한 PLC Data Agent 수와 PLC Data point 수를 알아보기 위한 테스트를 진행

하였다. 테스트는 PLC DataSet Get 주기가 0.1초이면서 PLC DataSet Get 주기만 0.05초인 환경에서 PLC Data Agent의 개수를 3대로 늘리고, 각 PLC Data Agent 당 할당 PLC Data Point의 개수를 3개로 줄여서 테스트해 본 결과 데이터 손실 빈도수가 준 것을 확인 할 수 있었다. 더 중요한 것은 Kafka 서버와 라즈베리파이 보드 간의 데이터 손실 빈도수의 차이가 없으며, 최대 손실 구간이 줄었다는 것이다. <표 3>은 위의 테스트 결과를 도표로 나타낸 것이다.

표 3. PLC DataSet Set(0.1초), Get(0.05초)주기에서 테스트
Table 3. Tests with a DataSet Set:0.1, Get:0.05 sec

비교 항목	테스트 환경			
PLC Data Agent	2대		3대	
PLC Emulator	Rasberry Pi	Kafka 서버	Rasberry Pi	Kafka 서버
PLC Data Point	5point	5point	3point	3point
최소 손실구간	0.1초	0.1초	0.1초	0.1초
최대 손실구간	0.9초	0.7초	0.7초	0.5초
전송데이터	부분손실	부분손실	부분손실	부분손실
손실 빈도수	7개	6개	5개	5개

위 테스트를 통하여, WiFi 환경에서는 Rasberry Pi에 비해 성능이 좋은 Kafka 서버에 설치된 PLC Emulator의 수집 성능이 더 좋으며, PLC DataSet Set 주기가 0.1초인 조건에서의 PLC DataSet Get 주기는 0.005초보다는 0.05초가 적정하며, 본 연구의 목적은 라즈베리파이 보드를 이용한 데이터 수집 시스템을 구현하는 것이므로 Kafka 서버와 라즈베리파이 보드의 데이터 손실 빈도수의 차이가 없고 최대 손실 수간이 줄어든 3대의 PLC Data Agent, 3개의 Data Point 환경이 적정하다는 결론을 얻을 수 있었다.

2) 데이터 이중화 환경에서의 테스트 결과

WiFi 환경에서 테스트한 결과, 적정한 PLC DataSet Set/Get 주기와 PLC Data Agent의 개수는 알 수 있었으나, 미비하나마 데이터 손실이 발생하는 단점을 보완하고자 Apache ZooKeeper를 이용한 데이터 이중화 수집 환경을 구축한 후 테스트를 시행하였다. 테스트 조건은 WiFi, PLC DataSet Set 주기는 0.1초, PLC DataSet Get 주기는 0.05초인 환경에서 PLC Emulator 설치 환경 및 PLC DataSet Set/Get 주기와 PLC Data Agent의 개수를 알아보기 위한 테스트를 시행하였다.

표 4. 데이터 이중화 환경에서의 테스트
Table 4. Testing in Data Mirroring Environments

비교항목	테스트 환경			
PLC Data Agent	2대		3대	
PLC Emulator	Rasberry Pi	Kafka 서버	Rasberry Pi	Kafka 서버
PLC Data Point	5point	5point	3point	3point
손실 구간	없음	없음	없음	없음
전송데이터	없음	없음	없음	없음
손실 빈도수	0개	0개	0개	0개

그 결과, <표 4>에서 알 수 있듯이 PLC Emulator가 설치된 환경 및 PLC Data Agent의 수와 PLC Data Point의 수의 변화에도 데이터 수집율의 영향을 받지 않는 것을 확인 할 수 있었다. 마지막으로 WiFi 환경에서 PLC Emulator 설치 환경과 PLC Data Agent당 할당된 Data point수에 따라 데이터 손실률의 차이를 보였던, PLC DataSet Set 주기가 0.1초, PLC DataSet Get 주기는 0.005초인 환경에서 PLC Data Point 개수도 10개로 늘려서 테스트해 보았으나 데이터 손실이 전혀 없는 것으로 나타났다.

위와 같이 다양한 조건을 주어 테스트한 결과, PLC 데이터 수집을 위한 라즈베리파이 보드를 추가함으로써 데이터 수집 성능 향상을 꾀하기 위해서는 Ethernet 환경이 적정하고, 각 PLC Transfer Agent 별로 데이터 수집 포인트를 분할 수집하는 것이 가능하며, PLC Transfer Agent 추가 구성 시 데이터 수집 성능이 향상된다는 것을 알 수 있었다.

또한, 1ms 주기로 단일 PLC Data Point를 전송하게 되면 실시간 데이터 수집 시 네트워크오버헤드 및 전송지연으로 인해 PLC 데이터 수집 성능을 보장하기가 어려우므로, 10ms 주기로 10개의 PLC Data Point를 묶어서 전송하는 것이 효과적이라는 결론을 얻었으며, 대부분의 EMS 기 구축 환경이 Ethernet 환경으로 되어 있다는 것을 감안했을 때 충분히 실효성이 있다고 할 것이다.

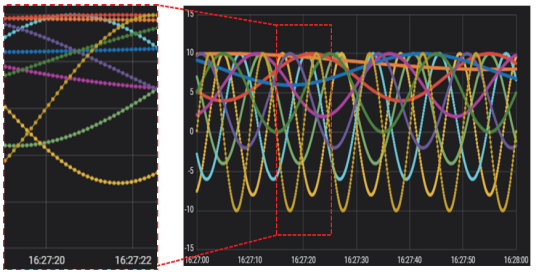

그림 6. WiFi 환경에서의 전송 데이터 시각화
Figure 6. Visualization of transmission data in WiFi

<그림 6>에서 알 수 있듯이, WiFi 및 Ethernet 환경에서의 데이터 전송의 신뢰성 향상을 위하여 3대의 라즈베리파이 보드에 데이터를 분산 전송하였으며, Apache Zookeeper를 도입하여 이중화하고 빠른 데이터 전송과 장애 발생 시 데이터 유실 없이 over/fail back이 가능하도록 구축한 결과, WiFi 환경에서도 데이터 손실 없이 전송되는 것을 확인할 수 있었다.

3. 결 론

인더스트리 4.0은 전통 제조업에 새로운 ICT를 결합, 적용해 모든 생산 과정에서 자율 최적화를 실현하겠다는 전략으로 다양한 상황 변경에서도 낭비나 시행착오가 없는 효율적인 '제조 최적화'의 달성하는 스마트 팩토리를 목표로 한다. 본 논문에서는 스마트 팩토리의 환경에서 발생하는 데이터 수집의 신뢰성 향상을 위해 라즈베리파이 보드(H/W) 와 분산 메시징 기반의 Kafka(S/W)를 적용한 Hadoop PLC 데이터 수집 장치 구현하고 에뮬레이터를 통하여 성능을 검증하였다.

그 결과, 라즈베리파이 보드 추가를 통하여 데이터 수집 성능 향상을 꾀할 수 있는 조건으로는 3대의 PLC Transfer Agent를 라즈베리파이 보드로 구현하는 것이 적정하고, 1ms 주기의 PLC 데이터 수집 성능을 보장하기 위해서는 단일 Point 의 1 ms 주기의 실시간 데이터 수집은 네트워크오버헤드 및 전송지연으로 인해 불가능하므로, Ethernet 연결 구성에서 10ms 주기 10개 Point 묶어서 전송하는 것이 효과적이라는 것을 알 수 있었다. 또한, 산업현장에서 장비의 안전성과 신뢰성은 매우 중요하므로 PLC 데이터 수집 장치를 복수 개로 배치하고 Apache Zookeeper를 도입하여 이중화를 구현함으로써 WiFi 및 외부 네트워크에서는 우려되는 성능 저하로 인한 데이터의 누수 문제를 해결할 수 있었다.

따라서 BDCS-DM은 소규모 공장의 PLC 설비 데이터 수집 시스템 구축에 있어 비용과 시간 절감 효과뿐만 아니라 설비 이상 또는 문제 감지 예측 분석을 위한 데이터 수집용으로 유용하게 활용 가능하리라 본다. 또한, 스마트 팩토리뿐만 아니라 빅 데이터를 사용하는 모든 분야에서 응용 가능할 것으로 판단된다. 향후에는 실제 생산 공정에 적용해 보고, 본 시스템이 생산성 향상과 품질 개선에 미치는 영향을 검증해 볼 계획이다.

References

[1] B. Behard, S. Yang, H. A. Kao, and J. Lee, *Cyber-physical systems architecture for self-aware machines in industry 4.0 environment*, International Federation of Automatic Control, Vol. 48, No. 3, pp. 1622-1627, 2015.

[2] *The based system of the fourth industrial revolution, CPS, Software Policy Research Institute*, MONTHLY SOFTWARE ORIENTED SOCIETY, pp. 18-23, Dec. 2016.

[3] J. Lee, C. Jin, and B. Bagheri, *Cyber physical systems for predictive production systems*, Production Engineering, Vol. 11, No. 2, pp. 155-165, Apr. 2017.

[4] H. J. Shin, and C. H. Oh, *Study on CPS implementation for smart manufacturing*, In Proceedings of the 2016 Asia Workshop on IT Convergence, Feb. 2017.

[5] M. S. D. Brito, S. Hoque, R. Steinke, A. Willner, and T. Magedanz, *Application of the fog computing paradigm to smart factories and cyber-physical systems*, Transactions on Emerging Telecommunications Technologies, pp. 1-14, May 2017.

[6] H. J. Shin, K. W. Cho, and C. H. Oh, *SVM-based dynamic reconfiguration CPS for manufacturing system in industry 4.0*, Wireless Communications and Mobile Computing, Vol. 2018, p. 13, Jan. 2018.

[7] S. Kim, Y. Kim, and W. Jim, *The design of method for efficient processing of small Files in the distributed system based on Hadoop framework*, J. of the Korea Institute of Electronic Communication Sciences, Vol. 10, No. 10, pp. 1115-1121, Oct. 2015.

[8] H. Li, *An approach to improve flexible manufacturing systems with machine learning algorithms*, IECON 2016 – 42nd Annual Conference of the IEEE Industrial Electronics Society, pp. 54-59, Oct. 2016.

[9] J. Jang, and E. J. Kim, *Survey on industrial wireless network technologies for smart factory*, Journal of Platform Technology, Vol. 4, No. 1, pp. 3-10, Mar. 2016.

[10] J. B. Park, A. Lee, and T. Kim, *Implementation of high speed big data processing system using in memory data grid in semiconductor process*, The Journal of The Korea Institute of Intelligent Transportation Systems, Vol. 15, No. 5, pp. 125-133, Oct. 2016.

[11] H. J. Shin, M. H. Jeon, and C. H. Oh, *Development of smart work-based production management system to improve work efficiency of workers in smart factory*, Far East Journal of Electronics and Communication, Vol. 18, No. 1, pp. 101-111, Jan. 2018.

[12] MTConnect Institute, http://www.mtconnect.org/standard -documents/, Jan. 2017.

[13] OPC Foundation, https://opcfoundation.org/developer-tools/ certification-specifications, Feb. 2017.

[14] The Modbus Organization, http://www.modbus.org/specs.php, Feb. 2017.

[15] Apache Kafka, https://kafka.apache.org/, Oct. 2017.

[16] Apache Hadoop, https://hadoop.apache.org/, Oct. 2017.

[17] Apache ZooKeeper, https://zookeeper.apache.org/, Oct. 2017.

Rasberry Pi를 이용한 분산 메시징 기반 빅 데이터 수집 시스템 구현

허선영

한국기술교육대학교 온라인평생교육원

요 약

최근 스마트팩토리가 이슈화되면서 기존 MES, SCADA, PLC 등의 시스템을 사용하고 있는 공장에서 PLC 데이터를 수집 및 분석하여 CPS를 구축하고자 하는 움직임이 활발하다. 그러나 현장에서는 기존 시스템을 분석하고 빅데이터 수집 및 저장을 위한 환경 구축 비용과 시간이 많이 투자된다는 이유로 기존 공장을 스마트팩토리화하는데 많은 어려움을 격고 있다. 이에 본 논문에서는 기존 시스템의 Modbus를 통해 전송되는 PLC 데이터를 읽어서 빅데이터 서버로 송신하기 위한 Rasberry Pi 보드를 추가설치함으로써 PLC 데이터 수집이 가능하도록 하기 위한 분산 메시징 기반 빅 데이터 수집 시스템(Big Data Collecting System based on Distributed Messaging:BDCS-DM)을 설계 및 구현하였다. 시뮬레이션을 통하여 Raspberry Pi를 설치함으로써 데이터 수집 성능 향상됨을 확인할 수 있었고, 추가 가능한 적정한 Raspberry Pi의 개수는 3대이며, 1ms 주기의 PLC 데이터 수집 성능을 보장하기 위해서는 Ethernet 환경에서 10ms당 10개 의 Point를 묶어서 전송하는 것이 효과적이라는 것을 알 수 있었다. 구현한 시스템은 스마트팩토리뿐만 아니라 빅데이터를 사용하는 모든 분야에서 응용 가능할 것으로 판단된다.

Sun-Young Heo received B.S.(1995) from Daejeon University, the M.S.(2005) and the Ph.D.(2012) in Department of Electrical and Electronic Engineering at Korea University of Technology and Education. She is a research professor at Online Lifelong Education Institute, Korea University of Technology and Education. Her current research interests include Intelligent LMS, Recommendation system, Big Data, and Motivation.

E-mail address: hsysj119@koreatech.ac.kr