JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

응용 및 융합

J. Adv. Navig. Technol. 24(6): 620-626, Dec. 2020

라즈베리파이를 이용한 Modbus TCP 기반 태양광 발전소 모니터링 시스템

Modbus TCP based Solar Power Plant Monitoring System using Raspberry Pi

박 진 환 · 김 창 복*
가천대학교 에너지IT학과

Jin-Hwan Park · Chang-Bok Kim*
Department of Energy IT, Gachon University, Gyeonggi-do, 13120, Korea

[요 약]

본 연구는 IOT 장비인 라즈베리파이를 마스터(master)로 이용하고 인버터를 슬레이브(slave)로 하여 모드버스 TCP 통신을 기반한 태양광 발전 모니터링 시스템을 제안하였다. 본 모델은 라즈베리파이에 다양한 센서를 추가하여 태양광 발전소의 모니터링에 필요한 정보를 추가하였으며, 실시간 발전량 예측을 통해 발전량 예측과 모니터링 정보를 스마트 폰으로 송신하였다. 또한, 서버에 태양광 발전소에서 지속해서 생성되는 정보를 빅데이터로 구축하였으며, 발전량 예측을 위한 딥러닝 모델을 학습하여 갱신하였다. 연구 결과로서 인버터에서 라즈베리파이로 모드버스 TCP 기반으로 안정적인 통신이 가능하였고, 라즈베리파이에서 학습된 딥러닝 모델로 실시간 예측이 가능하였다. 서버는 빅데이터로 다양한 딥러닝 모델 학습이 가능하였으며, LSTM이 학습 오차 0.0069, 테스트 오차 0.0075, RMSE 0.0866 등으로 가장 좋은 오차를 보임을 확인하였다. 본 모델은 다양한 제조사의 인버터에 대해서 보다 간단하고 편리하며 발전량을 예측할 수 있는 실시간 모니터링 시스템 구현이 가능함을 제시하였다.

[Abstract]

This research propose and simulate a solar power generation system monitoring system based on Modbus TCP communication using RaspberryPi, an IOT equipment, as a master and an inverter as a slave. In this model, various sensors are added to the RaspberryPi to add necessary information for monitoring solar power plants, and power generation prediction and monitoring information are transmitted to the smart phone through real-time power generation prediction. In addition, information that is continuously generated by the solar power plant is built on the server as big data, and a deep learning model for predicting power generation is trained and updated. As a result of the study, stable communication was possible based on Modbus TCP with the Raspberry Pi in the inverter, and real-time prediction was possible with the deep learning model learned in the Raspberry Pi. The server was able to train various deep learning models with big data, and it was confirmed that LSTM showed the best error with a learning error of 0.0069, a test error of 0.0075, and an RMSE of 0.0866. This model suggested that it is possible to implement a real-time monitoring system that is simpler, more convenient, and can predict the amount of power generation for inverters of various manufacturers.

Key word : Modbus TCP protocol, Inverter, Raspberrypi, Monitoring system, Deep learning.

https://doi.org/10.12673/jant.2020.24.6.620

Received 12 November 2020; **Revised** 26 November 2020
Accepted (Publication) 21 December 2020 (30 December 2020)

***Corresponding Author : Chang-Bok Kim**

Tel : +82-10-8908-3946
E-mail : cbkim@gachon.ac.kr

Ⅰ. 서 론

태양광 발전소는 태양광으로 전력을 생산하는 시스템으로 에너지 변환과정에서 기계적, 화학적 작용이 없으며, 구조가 단순하고 유지 보수가 간단하며, 수명이 20 ~ 30년 정도로 길며 환경친화적이다. 태양광 시스템은 태양광으로 전력을 생산하기 때문에, 무한한 자원을 지니고 있고, 지역에 상관없이 균등하게 생산할 수 있다.

태양광 발전소는 태양광 패널 모듈의 직렬 및 병렬로 구성된 태양광 패널, 태양광 인버터, 수배전 설비 등으로 구성된다. 태양광 패널은 전기를 생산하는 반도체 소자로서, 기상요소에 따라 민감하게 전압과 전류가 바뀌는 불안정한 전력 공급 장치이다. 인버터는 태양광 패널에서 발전한 직류전력을 전력회사의 전력과 동일한 전압과 주파수의 교류전력으로 변환하고 사고 발생 시에 계통을 보호하는 장치 등으로 구성된다. 또한, 인버터는 태양광 시스템의 모든 정보를 담고 있어 원활한 태양광 발전소의 관리를 위해서는 실시간 모니터링이 필수적이다. 최근, 태양광 시스템이 증가하고 태양광 시스템 모니터링을 위한 산업 인프라가 증가하고 있다. 그러나 태양광 발전소 모니터링 시스템은 시공 업체별로 서로 다른 솔루션을 통해 구축되어 관리와 모니터링의 통합이 어렵다[1].

사물인터넷(internet of things)은 각종 사물에 마이크로프로세서와 센서 그리고 유무선 통신 기능을 내장하여 인터넷으로 사물들을 연결하여 데이터를 주고받아 스스로 분석하고 학습한 정보를 사용자에게 제공하거나 사용자가 이를 원격 조정할 수 있는 인공지능과 함께 4차 혁명의 핵심 기술이다[2],[3].

모드버스(modbus)는 마스터(master)와 슬레이브(slave) 구조에 기반한 통신 프로토콜이다. 최근 발표된 모드버스 TCP는 초고속 이더넷 데이터 전송을 보장하고 간소화되고 있으며, 제조사에 구애받지 않는 데이터 구조로 다양한 제조사의 기기 사이의 통신을 가능하게 한다[4],[5].

본 연구는 IOT 장비인 라즈베리파이를 마스터로 이용하고 인버터를 슬레이브로 하여 모드버스 TCP 통신을 기반한 태양광 발전 시스템의 모니터링 시스템 모델을 제안하고 시뮬레이션하였다. 제안모델은 인버터, 라즈베리파이, 서버, 구글의 파이어베이스, 사업자의 스마트 폰으로 구성되며, 다양한 제조사의 인버터를 모드버스 TCP를 이용하여 모니터링 할 수 있는 모니터링 시스템을 제안하였다. 본 연구는 시뮬레이션을 위해 경기도 연성의 태양광 발전소 데이터를 사용하였으며, 개방형 모드버스 라이브러리인 pymodbusTCP 사용하였다.

본 논문은 2장에서 관련 연구로서 모드버스 TCP에 관해서 서술하였으며, 3장에서 모니터링 시스템 모델을 제안하였다. 또한, 4장에서 구현된 모델의 실험 결과 및 비교 분석을 하였으며, 마지막으로 결론에 관해서 서술하였다.

Ⅱ. 모드버스 TCP

모드버스는 산업용 필드 버스 분야에서 원거리에 있는 장비의 감시나 제어를 위해 SCADA(supervisory control and data acquisition)방식의 네트워크 통신에 사용되는 프로토콜이다. 모드버스는 마스터와 슬레이브 구조로 마스터는 슬레이브에 요청을 보내고 응답을 기다리는 구조이다. 그림 1에 모드버스 TCP 구조에 대해서 나타냈다[6],[7].

모드버스는 모드버스 시리얼, 모드버스 플러스 그리고 모드버스 TCP 등이 있다. 모드버스 TCP는 모드버스의 발전된 형태의 프로토콜로서 이더넷 상에서 운용되고 개방된 표준이며, 장치 사이의 정보 교환, 모니터링, 관리 등을 위해 광범위하게 사용된다. 모드버스 TCP는 최소한의 하드웨어와 다양한 운용체계 하에서도 개발할 수 있으며, 공개된 표준과 인터넷의 범용성으로 제조사에 종속될 필요도 없다.

마스터와 슬레이브 간의 통신을 위해서는 프레임구조에 맞는 데이터 패킷의 프로토콜을 이용하여 전송해야 한다. 그림 2에 모드버스 TCP의 프레임구조를 나타냈다. 모드버스 TCP는 MBAP (modbus application protocol), function code, data 프레임 형식으로 이루어져 있다. MBAP는 7 byte이며, transaction ID, protocol ID, length 등이 있다[8].

Transaction ID는 요청 및 응답에 관련한 작업의 순서번호를 나타내며 마스터에 의해 설정된다. 마스터는 동작할 때 0x0000부터 명령마다 1씩 증가하고, 슬레이브로 동작할 때는 마스터가 설정한 값에 맞게 응답한다.

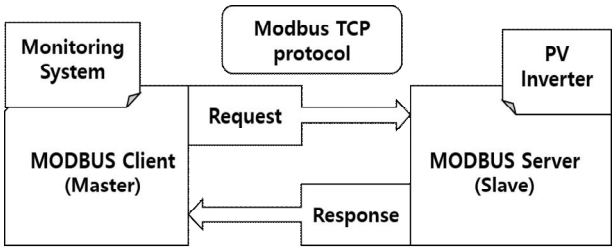

그림 1. 모드버스 통신 형식
Fig. 1. Modbus communication format.

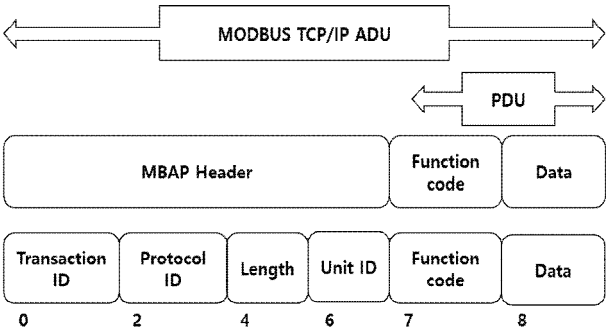

그림 2. 모드버스 프로토콜
Fig. 2. Modbus protocol.

표 1. 모드버스 TCP function code
Table 1. Modbus TCP function code.

Memory	Name	Funtion code	Device
Coil	Read Discrete Inputs	02(0x02)	Input / Output
	Read Coil	01(0x01)	
	Write Single Coil	05(0x05)	
	Write Multiple Coils	15(0x0F)	
Register	Read Input Register	04(0x04)	ADC / PWM
	Read Holding Registers	03(0x03)	
	Write Single Register	06(0x06)	
	Write Multiple Registers	16(0x10)	

Protocol ID는 프로토콜의 아이디를 나타내며 0x0000으로 고정값이다. length 해당 프레임의 마지막까지의 길이를 바이트 단위로 나타낸다. unitID는 TCP 구조가 아닌 다른 장치 아이디이며, TCP 포트는 0x01로 고정이다. function code는 슬레이브의 메모리에 값을 읽고 쓰기 위한 코드 번호이며, 모드버스 TCP는 1, 2, 4, 5, 6, 15, 16 값을 사용한다. 표 1에 모드버스 TCP의 function code에 대해서 나타냈다.

모드버스 TCP는 16 비트 워드 영역(resisters)과 비트 영역(coils)과 단지 읽기만 가능한 메모리와 읽고 쓰기가 가능한 메모리로 나누어져 있다. 이러한 메모리는 슬레이브의 메모리이며, 마스터는 function code를 이용하여 슬레이브의 메모리를 읽거나 원하는 값으로 변경할 수 있는 것이다. 표 2에 모드버스 데이터 타입에 대해서 나타냈다.

표 2. 모드버스 데이터 타입
Table 2. Modbus data type.

Data model	Data type	Read/Write	Explanation
Discrete Inputs	1 bit	Read-Only	Equipment port status
Coils	1 bit	Read-Write	changeable
Input Registers	16blt	Read-Only	Equipment port status
Holding Registers	16blt	Read-Write	changeable

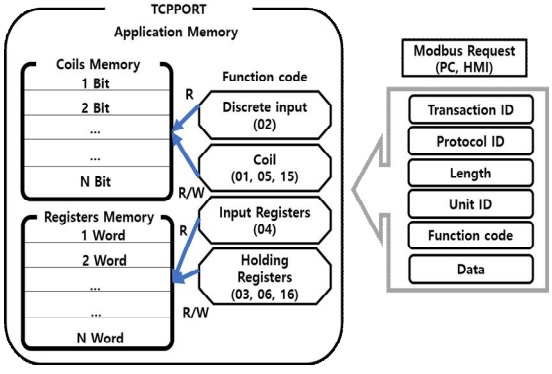

그림 3. 모드버스 TCP 메모리 구조
Fig. 3. Modbus TCP memory structure.

표 3. 인버터 레지스터 맵
Table 3. Inverter register map.

Function code	Read Input Register(measurement)		
	Reg.addr	address	parameter
04H	30027	26	Inverter State
	30028	27	S/W Fault State
	30029	28	H/W Fault State
	30030	29	Today Generation Time
	30033 - 30034	32-33	Total Generation Time
	30035 - 30036	34-35	PV voltage
	30037 - 30038	36-37	Inverter DC Voltage
	30039 - 30040	38-39	PV current
	30041 - 30042	40-41	PV Power
	30043 - 30044	42-43	Total Voltage
	30045 - 30046	44-45	Total current
	30047 - 30048	46-47	Total Active Power
	30049 - 30050	48-49	Total Reactive Power
	30051 - 30052	50-51	Total Power Factor

그림 3에 모드버스 TCP 메모리 구조에 대해서 나타냈다. 모드버스 TCP 메모리의 주소는 0000~270E이며, 해당 function code에 따라 어떤 메모리에 접근할 것인지, 어떤 작업수행(read / write)을 할 것인지가 나누어져 있다.

Data는 function code에 따라 그 구조가 조금씩 다르지만, 기본적으로 start address, length, byte count, data 등이 있다. start address는 2 바이트이며 접근하는 메모리의 시작 번지를 나타낸다. length는 2 바이트이며 시작 번지부터 값을 읽거나 쓸 길이를 나타낸다. byte count는 1 바이트이며 메모리 데이터의 바이트 수를 나타낸다. data는 읽고 쓰는 실제 데이터이다.

모드버스 TCP는 인버터 레지스터의 정보를 읽는 데 사용된다. 본 연구에서 슬레이브인 인버터의 레지스터 정보를 읽는 데 사용되며 태양광 발전소 인버터의 레지스터 맵 정보는 인버터 제조업체에 따라 크게 다르지만, 수집된 정보에는 큰 차이가 없다. 표 3에 인버터 레지스터 맵에 대해서 나타냈다[1].

Ⅲ. 제안 모니터링 시스템

태양광 발전소는 태양광 패널, 인버터, 라즈베리파이로 구성되어 있다. 인버터의 레지스터는 현재 발전소 상태, 패널의 전압 및 전류, 발전 시간, 발전량 등 태양광 발전소의 중요한 정보를 담고 있다. 라즈베리파이는 와이파이와 블루투스 통신이 가능한 초소형 컴퓨터로써 각 센서 부착이 가능하다. 본 모델은 인버터, 라즈베리파이, 서버, 구글의 실시간 데이터베이스, 사업자의 스마트 폰으로 구성되어 있다. 인버터는 요청된 태양광 시스템의 정보를 응답하는 슬레이브이다. 라즈베리파이는 마스터로서 모드버스 TCP로 인버터의 정보를 수집하고 센서로

부터 태양광 예측을 위한 정보를 수집하고 저장한다. 또한 현재 입력된 모든 정보를 통해 발전량을 예측하고 구글의 실시간 데이터베이스에 전송한다. 서버는 일정 시간마다 라즈베리파이의 정보를 수집하여, 태양광 시스템의 모든 정보에 대해 빅데이터를 구성하고, 빅데이터를 이용하여 딥러닝으로 발전량을 예측한다. 스마트 폰은 구글의 실시간 데이터베이스의 정보를 모니터링한다. 그림 4에 제안 모델에 대해서 나타냈다. 본 모델은 다음과 같이 동작한다.

1. 슬레이브인 인버터는 모드버스 TCP 프로토콜로 발전량 데이터를 1시간마다 마스터인 라즈베리파이에 전송한다.

2. 라즈베리파이는 부착된 센서를 이용하여 현재 온도, 태양광 패널 온도, 경도 일사량, 수평 일사량을 측정하고, 인버터의 정보와 함께 임시 데이터베이스에 저장한다.

3. 라즈베리파이는 실시간 데이터를 서버에서 학습된 딥러닝 모델을 이용하여 1시간 후, 2시간 후, 3시간 후, 24시간 후 발전량을 예측한다.

4. 라즈베리파이는 실시간 데이터와 예측된 발전량을 파이어 베이스의 실시간 데이터베이스에 전송하여 저장한다.

5. 서버는 태양광 시스템의 빅데이터를 구축하기 위해 일정 시간마다 서버의 빅데이터 베이스에 저장하고 라즈베리파이의 데이터베이스를 삭제한다. 이것은 임베디드 시스템의 메모리 용량이 적기 때문이다.

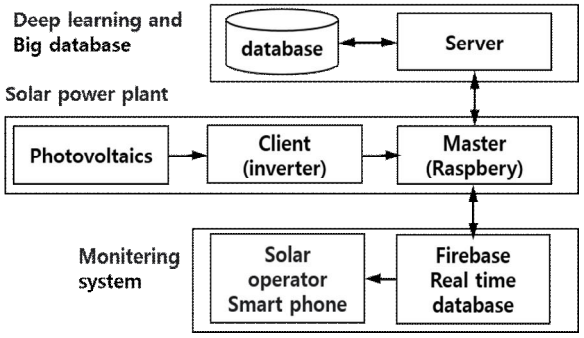

그림 4. 제안 모델
Fig. 4. Proposed model.

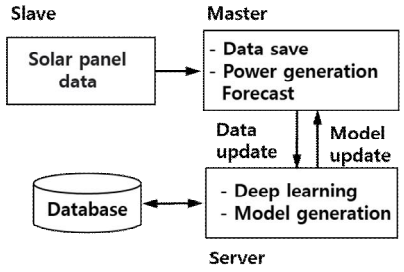

그림 5. 마스터와 서버
Fig. 5. Master and server.

6. 서버는 빅데이터 베이스를 LSTM 딥러닝 네트워크로 1시간 후, 2시간 후, 3시간 후, 24시간 후 발전량을 예측을 위한 학습 모델을 갱신하며, 이를 라즈베리파이에 업로드한다.

7. 실시간 데이터베이스의 정보를 태양광 사업자의 스마트 폰에 전송한다.

라즈베리파이와 서버는 데이터 갱신과 모델 갱신 등 상호 관계가 있다 그림 5에 라즈베리파이와 서버와의 상호 관계를 나타냈다. 다음은 모드버스 통신에 사용한 슬레이브인 인버터의 주요 데이터 코드이다.

```
builder = BinaryPayloadBuilder(byteorder=Endian.Big)
registers = builder.to_registers()
client.write_registers(address, registers, unit=UNIT)
```

인버터는 슬레이브 역할을 하며, 태양광 패널 데이터의 형식을 지정하기 위해 라이브러리에 포함된 BinaryPayloadBuilder 함수를 호출하고, 지정한 주소로 태양광 패널 정보를 포함한 registers 값을 전송한다. 다음은 마스터인 라즈베리파이의 주요 데이터 코드이다.

```
class CustomDataBlock(ModbusSparseDataBlock):
    def setValues(self, address, value):
        super(CustomDataBlock, self).setValues(address, value)
        for a in range(0, 5) :
            if value[a] > 32767 and value[a] < 65536 :
                value[a] = value[a] - 65536
def run_custom_db_server():
    StartTcpServer(context, identity=identity, address=("ip addr, 5020))
    block  = CustomDataBlock([0]*100)
    store  = ModbusSlaveContext(hr=block, ir=block)
    context = ModbusServerContext(slaves=store, single=True)
```

라즈베리파이는 마스터 역할을 하며, IP 주소와 포트 번호를 설정하여 StartTcpServer 함수에 의해 구동된다. 또한, 데이터 블록, 저장소, 컨텍스트 등을 설정하는 단계를 거친 후, 전송된 데이터 개수만큼 반복하는 재귀 함수를 실행하여 인버터 정보를 수신한다.

서버는 발전량 예측을 위해 DNN과 RNN 그리고 LSTM을 사용할 수 있다. 다음은 케라스로 작성된 LSTM 학습 모델과 학습 모델 저장 파일에 대해서 나타냈다.

```
model.add(LSTM(10, input_shape=(24, 5)))
model.add(Dense(1, activation='linear'))
model.compile(loss = 'mse', optimizer =optimizers.SGD(lr = 0.001))
modelCheckpoint = tf.keras.callbacks.ModelCheckpoint('model_1.h5')
```

타임 스탬프(stamp)는 태양광 시스템 데이터의 시계열 특성을 고려하여 1일 24시간을 반복하는 데이터이므로 24개로 하였다. 입력데이터는 현재 온도, 태양광 패널 온도, 경도 일사량, 수평 일사량, 인버터 발전량 등 5개이며, 실제 데이터인 라벨은 1시간, 2시간, 3시간, 24시간 후의 발전량으로 하였으며, 활성

화 함수는 linear를 사용하였다. 학습 결과 시간별 예측 모델의 가중치와 바이어스를 저장하였다. 본 학습 모델들은 라즈베리파이에 일정 시간마다 갱신하여 실시간으로 전송되는 데이터에 대한 발전량 예측을 수행하였다. 다음은 라즈베리파이에서 발전량 예측을 위한 모델들 중 24시간 후의 코드를 나타냈다.

```
model_24 = keras.models.load_model('LSTMmodel_24.h5')
predict_24 = model_24.predict(x_data)
```

LSTMmodel_24.h5은 서버에서 24시간 후의 발전량을 예측한 학습 모델이며, 모델을 읽어와서 실시간으로 전송되는 x_data 데이터를 이용해서 24시간 후의 발전량을 예측한다. 마스터인 라즈베리파이에서 예측된 발전량과 함께 슬레이브인 인버터에서 전송된 데이터들은 모니터링하기 위해 구글의 실시간 데이터베이스인 파이어베이스에 전송하였다. 다음은 파이어 베이스에 전송된 코드에 대해서 나타냈다.

```
ref = db.reference('modbus')
    name_ref = ref.child('modbus')
    name_ref.update({
    'Temperature':      temperate,      # 주위온도
    'Surface':          surface,        # 표면온도
    'Horizontal' :      horizontal,     # 표면 일사량
    'Longitude' :       longitude,      # 경도 일사량
    'Generation' :      generation,     # 현재 발전량
    'Prediction1'       prediction1,    # 발전량 예측 1
    'Prediction2'       prediction2 ,   # 발전량 예측 2
    'Prediction3'       prediction3 ,   # 발전량 예측 3
    'Prediction24'      prediction24 ,  # 발전량 예측 4
    } )
```

파이어베이스는 시간당 인버터 정보와 발전량 예측 정보를 JSON 형태로 전송받아 저장한다. 모델의 파이어베이스 데이터베이스의 테이블 명은 modbus이며 각 필드는 모니터링할 정보로서 라즈베리파이에서 기본적으로 전송된 5개의 정보와 전송된 정보를 기반으로 예측된 발전량으로 1 ~ 3시간 후 그리고 24시간 후의 발전량 예측 결과이다. 최종적으로 태양광 발전 사업자에게 스마트 폰으로 인버터의 중요 정보 및 발전량 예측 정보를 전송하여 모니터링 한다.

Ⅳ. 결과 및 비교 분석

제안 모델은 리눅스 기반의 라즈베리파이 4를 사용하였으며, 윈도우 10 기반에 모델 구축을 위한 코드는 파이선 3.8로 구축하였다. 또한, 개방형 모드버스 라이브러리인 pymodbusTCP 사용하였으며, 딥러닝 학습을 위해 tensorfolw 2.3과 keras 2.2를 사용하였다. 본 모델을 시뮬레이션하기 위해 경기도 연성의 태양광 발전소 데이터를 사용하였다. 데이터는 총 17,520개이며, 학습을 위해 17,000개를 사용하였으며 테스트를 위해 2,520개를 사용하였다.

표 4. 데이터 구조
Table 4. Data structure.

Ambient temp.	Surface temp.	Gradient insolation	Horizontal insolation	Inverter output
-4	-5	0	0	0
-4	-5	40	0	14
-2	1	234	74	159
-1	6	354	179	289
0	9	419	258	821
0	10	453	304	1174
1	9	412	335	1116

```
응답 데이터 : [-2, -1, 19, 0, 0]
응답 데이터 : [-2, -2, 0, 0, 0]
응답 데이터 : [-3, -2, 0, 0, 0]
응답 데이터 : [-3, -2, 0, 0, 0]
응답 데이터 : [-3, -3, 0, 0, 0]
응답 데이터 : [-4, -3, 0, 0, 0]
응답 데이터 : [-4, -3, 0, 0, 0]
응답 데이터 : [-5, -4, 0, 0, 0]
응답 데이터 : [-5, -4, 40, 0, 14]
응답 데이터 : [1, -2, 234, 74, 159]
응답 데이터 : [6, -1, 354, 179, 289]
응답 데이터 : [9, 0, 419, 258, 821]
응답 데이터 : [10, 0, 453, 304, 1174]
응답 데이터 : [9, 1, 412, 335, 1116]
응답 데이터 : [8, 0, 397, 280, 924]
```

(a) slave

```
장치 1 요청 데이터 : [-2, -1, 19, 0, 0]
장치 1 요청 데이터 : [-2, -2, 0, 0, 0]
장치 1 요청 데이터 : [-3, -2, 0, 0, 0]
장치 1 요청 데이터 : [-3, -2, 0, 0, 0]
장치 1 요청 데이터 : [-3, -3, 0, 0, 0]
장치 1 요청 데이터 : [-4, -3, 0, 0, 0]
장치 1 요청 데이터 : [-4, -3, 0, 0, 0]
장치 1 요청 데이터 : [-5, -4, 0, 0, 0]
장치 1 요청 데이터 : [-5, -4, 40, 0, 14]
장치 1 요청 데이터 : [1, -2, 234, 74, 159]
장치 1 요청 데이터 : [6, -1, 354, 179, 289]
장치 1 요청 데이터 : [9, 0, 419, 258, 821]
장치 1 요청 데이터 : [10, 0, 453, 304, 1174]
장치 1 요청 데이터 : [9, 1, 412, 335, 1116]
장치 1 요청 데이터 : [8, 0, 397, 280, 924]
```

(b) master

그림 6. 모드버스 TCP 통신 결과
Fig. 6. Modbus communication result.

실험은 모드버스 통신 결과를 나타내기 위해 마스터 슬레이브 간 현재 온도, 태양광 패널 온도, 경도 일사량, 수평 일사량, 인버터 발전량 등 5개의 정보를 사용하였다. 표 4에 본 모델에서 사용한 데이터 구조를 나타냈다. 그림 6에 슬레이브인 인버터에서 마스터인 라즈베리파이로 모드버스 TCP 통신 결과를 보였다. 서버는 전송된 데이터를 축적하여 딥러닝으로 학습하였다. 딥러닝은 작은 입출력 요소를 사용해야 학습이 용이하기 때문에 입력데이터는 정규화하였으며, 학습을 위한 실제 값인 발전량은 1,000으로 나누었다. 다음은 실제 학습한 값에 대해서 나타냈다.

$$x\_data = (x\_data - Min) / (MAX - MIN) \quad (1)$$

$$y\_data = y\_data/1000 \quad (2)$$

표 5 딥러닝 기반의 DNN과 RNN 그리고 LSTM의 예측 결과를 보였다. 표 5와 같이 DNN은 학습 오차 0.0078, 테스트 오차 0.0115, RMSE 0.2344를 보였으며, RNN은 학습 오차 0.0077, 테스트 오차 0.0077, RMSE 0.0876을 보였다. 또한, LSTM은 학습 오차 0.0069, 테스트 오차 0.0075, RMSE 0.0866을 보였다. 이처럼 DNN보다는 시계열 데이터에 장점을 보이는 RNN과 LSTM이 우수한 결과를 보였으며, RNN보다 긴 스탬프에 장점을 보이는 LSTM이 학습과 테스트에서 안정적이며, 우수한 결과를 보였다. 표 6에 시간별 예측 결과에 대해서 나타냈다. 그림 7에 LSTM의 학습 오차와 테스트 오차를 보였다. 또한, 그림 8에 20일간의 실제값과 예측값 결과를 보였다.

표 5. 모델 예측 결과
Table 5. Model forecast results.

model	Train loss	Test loss	RMSE
DNN	0.0078	0.0115	0.2344
RNN	0.0077	0.0077	0.0876
LSTM	0.0069	0.0075	0.0866

표 6. 시간 예측 결과
Table 6. Time forecast results.

Prediction Time	Train loss	Test loss	RMSE
1 time later	0.0071	0.0074	0.0857
2 time later	0.0107	0.0118	0.1086
3 time later	0.0117	0.0123	0.1107
24 time later	0.0180	0.0188	0.1370

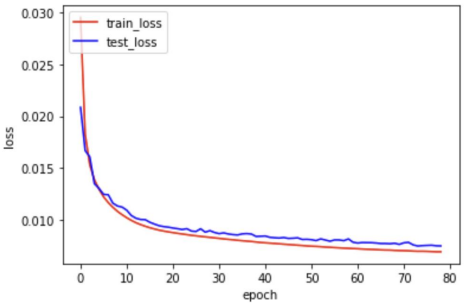

그림 7. 학습 오차와 테스트 오차
Fig. 7. Learning error and test error.

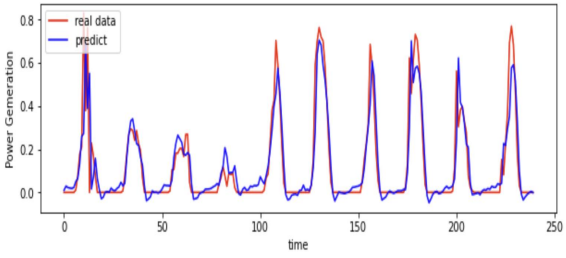

그림 8. 20 일간의 실제값과 예측값
Fig. 8. Actual and predicted values for 20 days.

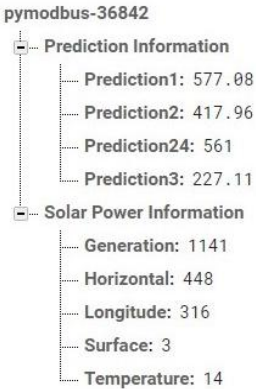

그림 9. 파이어베이스 실시간 데이터 베이스
Fig. 9. Firebase real-time database.

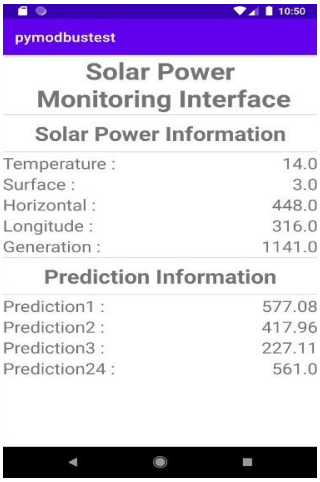

그림 10. 스마트폰 모니터링 화면
Fig. 10. Smartphone monitoring screen.

그림 9에 파이어베이스의 실시간 데이터베이스를 보였다. 또한, 그림 10에 태양광 발전 사업가가 모니터링 할 수 있는 스마트 폰 화면을 보였다. 모니터링 화면에 인버터에서 전송된 태양광 시스템의 정보뿐 아니라 라즈베리파이의 데이터와 예측된 발전량까지 출력됨을 볼 수 있다. 실험 결과 본 제안 모델은 다음과 같은 비교 분석 결과를 도출할 수 있다.

1. IOT 장비인 라즈베리파이에 다양한 센서의 추가로 최적의 모니터링 및 제어를 위해 필요한 정보를 수집할 수 있다.
2. 서버는 태양광 발전소에서 끊임없이 생성되는 정보를 빅데이터로 구축할 수 있으며, 빅데이터를 이용하여 태양광 발전소의 다양한 분석을 통해 발전량 및 고장 예측이 가능하고 사업자의 요구에 최적화된 모니터링이 가능하다.
3. 태양광 발전량은 기상요건에 간헐적이기 때문에 빅데이터를 통해 더 정밀한 단기간 예측뿐 아니라 장기간 예측을 통해 사업자의 태양광 시스템의 운영 관리에 많은 정보를 제공할 할 수 있다.

4. 다양한 제조사의 인버터에 대해서 보다 더 간단하고 편리한 모니터링 모니터링 시스템 구현이 가능하다.

Ⅴ. 결 론

본 연구는 다양한 제조사의 인버터를 모드버스 TCP를 이용하여 모니터링 할 수 있는 모니터링 시스템을 제안하였다. 본 연구는 라즈베리파이를 이용하여 태양광 시스템 관리에 필요한 추가 정보를 획득할 수 있어 더욱 정밀한 모니터링과 관리가 가능하다. 또한, 서버의 빅데이터를 이용하여 태양광 사업자의 요구를 심층적으로 분석하여 요구에 적합한 모니터링 시스템 구축이 가능하다. 태양광 시스템의 최적의 관리와 제어를 위해서는 빅데이터가 필요하며, 이러한 빅데이터를 수집하고 관리하며 효율적으로 분석하는 기술에 관한 연구가 지속해서 필요하다,

Acknowledgments

본 연구는 산업통상자원부(MOTIE)와 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구 과제입니다. (No. 20194030202290)

References

[1] S. Y. Kang, and I. W. Lee, “Implementation of PV monitoring system using python,” in *2019 21st International Conference on Advanced Communication Technology,* PyeongChang: Korea, pp. 453-455, 2019.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (IoT): a vision, architectural elements, and future directions,” *Future Generation Computer Systems*, Vol. 29, No. 7, pp. 1645-1660, 2013.

[3] L. In, and K. C. Lee, “The internet of things : applications, investments, and challenges for enterprises,” *Business Horizons*, Vol. 58, No. 4, pp. 431-440, 2015.

[4] J. S. Jeong, and S. B. Lee, “Design and implementation of wireless lighting LED controller using modbus TCP for a ship,” *Korean Institute of Navigation and Port Research*, Vol. 41, No. 6, pp. 395 − 400, 2017.

[5] T. Y. Kim, and H. S. Kim, “A study on status monitoring and control of wind power based on modbus TCP protocol,” in *Proceeding of The Korean Institute of Information Scientists and Engineers*, Busan: Korea, pp. 1,122 - 1,123, 2014.

[6] Q. Liu, and Y. Li, “Modbus/tcp based network control system for water process in the firepower plant,” in *2006 6th World Congress on Intelligent Control and Automation,* Vol. 1, IEEE, Dalian: China, pp. 432-435, 2006.

[7] X. He, E. Robards, R. Gamble, and M. Papa, “Anomaly detection sensors for a modbus-based oil and gas well-monitoring system,” in *2019 2nd International Conference on Data Intelligence and Security*, IEEE, South Padres: Island, pp. 1-8, 2019.

[8] S. D. Anto, S. Kanoor, D. Fraunholz, and H. D. Schotten, “Evaluation of machine learning-based anomaly detection algorithms on an industrial modbus/tcp data set,” in *Proceedings of the 13th International Conference on Availability*, *Reliability and Security*, Hamburg: Germany, pp. 1-9, 2018

[9] [Internet]. Available: https://Firebase.google.com.

박 진 환 (Jin-Hwan Park)
2014 3월 ~ 현재: 가천대학교 에너지IT학과 재학
관심분야 : 딥러닝, IoT, 임베디드 시스템

김 창 복 (Chang-Bok Kim)
1986 2월 : 단국대학교 전자공학화 졸업(공학사)
1989년 2월 : 단국대학교 전자공학과 (공학석사)
2009년 2월 : 인천대학교 컴퓨터 공학과(공학박사)
1994년 ~ 현재 : 가천대학교 IT대학 에너지 IT학과 교수
※관심분야 : 데이터 마이닝, 딥러닝, 강화학습, 사물인터넷