Journal of KIIT. Vol. 16, No. 2, pp. 17-27, Feb. 28, 2018. pISSN 1598-8619, eISSN 2093-7571 17
http://dx.doi.org/10.14801/jkiit.2018.16.2.17

스마트 플러그인 스위칭 모드 전원장치의 구현

송호범*, 정준**

The Implementation of Smart Plug-in Switching Mode Power Supply

Ho-Bum Song*, Jun Jeong**

본 논문은 2017년도 동양미래대학교 학술연구과제 지원에 의하여 연구되었음.

요 약

소규모의 연구실이나 실험실에서 사용하는 대부분의 전원장치는 동시에 사용할 수 있는 3개 이하의 출력전압을 일괄적으로 제공하는 방식의 고정된 구조를 지니고 있으며, 과전류로 발생되는 내부회로의 고장을 방지하는 기본적인 기능을 내장하고 있다. 이에 본 연구에서는 AC-DC 컨버터, 모듈형 DC-DC 컨버터, 전압 센싱 모듈, 제어장치, 태블릿 등을 사용하여 스마트 플러그인 스위칭 모드 전원장치를 제작하였다. 이 전원장치는 플러그인 방식으로 설계하여 필요한 출력전원 모듈만 장착하여 사용할 수 있는 구조를 지니고 있으며, 태블릿에서 실행되는 딥-러닝 엔진을 내장한 앱 프로그램에 의하여 전원장치를 진단하는 기능을 실행시킬 수 있다. 실험을 통해 적절한 데이터로 학습된 딥-러닝 엔진을 사용하면 전원장치의 고장을 판별하는 데이터가 감지되는 것을 확인할 수 있었다.

Abstract

Most power supplies used in small laboratories have a fixed structure, which provides up to three output voltages that can be used simultaneously, and these devices have only basic function to protect the internal circuit by overcurrent. In this study, we implemented a smart plug-in switching mode power supply using AC-DC converter, modular DC-DC converter, voltage sensing module, controller and tablet. This power supply is designed as a plug-in type and can be used with only the necessary output power module, and the function of diagnosing the power supply can be executed by the app program which incorporates the deep learning engine running on the tablet. Through experiments, using a deep learning engine that has been learned with appropriate data, it was confirmed that data for determining the failure of the power supply was detected.

Keywords
reinforcement learning, SMPS, AC-DC converter, deep-learning, arduino controller, plug-in, tablet

* 동양미래대학교 로봇자동화공학부
- ORCID: https://orcid.org/0000-0002-3451-5729
** 동양미래대학교 로봇자동화공학부(교신저자)
- ORCID: https://orcid.org/0000-0002-9755-3060

· Received: Nov. 30, 2017, Revised: Jan. 15, 2018, Accepted: Jan. 18, 2018
· Corresponding Author: Ho-Bum Song
Dept. of Robot & Automation System, Dongyang Mirae University, 445 Gyeongin-ro, Guro-gu, Seoul, 152-714, Korea,
Tel.: +82-2-2610-1775, Email: hbsong@dongyang.ac.kr

Ⅰ. 서 론

정보통신 분야에 대한 연구를 진행하는 대부분의 스타트업 개발실 또는 대학 연구실에서 측정용 장비로 사용하고 있는 전원장치는 일반적으로 동시에 사용할 수 있는 3개 이하의 출력전압을 일괄적으로 제공하는 방식의 모델이다. 부가적으로 장치의 내부에는 출력 단에는 전류의 이상을 감지하는 과전류 보호회로, 입력 단에는 1차 전류 제한하기 위한 휴즈 등이 내장되어 있다. 24시간 가동하는 이동통신 회사의 통신장치에서 사용되는 고가의 전원공급용 전원장치는 별도의 제어장치를 내장하여 출력상태를 모니터링한 데이터를 활용하여 고장을 진단할 수 있는 기능을 제공하고 있으며, 이러한 기능에 대한 연구는 지속적으로 진행되고 있다. 또한 빅데이터를 효과적으로 처리할 수 있는 다양한 인공지능 알고리즘이 개발되고 있으며, 동시에 이 데이터를 실시간으로 처리할 수 있는 성능을 지닌 임베디드 제어장치가 상용화되고 있어, 지속적으로 가동되는 전원장치와 같은 빅 데이터를 발생시킬 수 있는 하드웨어 시스템에서는 인공지능 알고리즘을 채택하여 시스템의 고장에 대한 사전 예측 및 효율적인 운영에 대한 연구도 진행 되고 있다[1]-[3]. 이에 본 연구에서는 스타트업 개발실 또는 대학 연구실에서 경제적으로 사용할 수 있는 출력전압에 대한 진단 기능이 있으며, 절연 능력이 우수하고 용량이 작은 부하에 적합한 플라이백 방식의 스위칭 모드 전원장치(Flyback A/C-D/C Converter)를 제작하고자 한다. 이 전원장치는 플러그인(Plug-in) 방식으로 설계하여 필요한 출력전원 모듈만 장착하여 사용할 수 있는 구조를 지니도록 한다. 또한 오픈소스 기반의 인공지능 알고리즘인 딥 신경망(Deep Neural Network) 및 강화학습(Reinforcement Learning)을 이용하여 자체 개발한 독자적인 딥-러닝 엔진으로 데이터에 대한 학습을 진행토록 한다[4]-[8]. 부하가 접속된 전원장치에서 발생한 출력전압은 테블릿에서 실행되는 앱 프로그램의 입력 데이터로 사용되며, 학습된 딥-러닝 엔진은 일정한 입력 데이터 패턴을 감지하여 진단 기능을 수행토록 한다.

Ⅱ. 스마트 전원장치의 설계 및 구현

그림 1은 본 연구에서 제안한 스마트 전원장치의 구성도를 나타낸다.

이 스마트 전원장치는 AC 입력전압을 메인 모듈인 AC-DC 컨버터에 의하여 DC 전압으로 변환한 후에 슬롯에 장착된 DC-DC 컨버터(Buck Converter)가 부하에 필요한 DC 출력전압을 공급한다. 반면에 다른 슬롯에 장착된 전압 센싱 모듈은 부하에 공급되는 DC 출력전압을 아두이노 제어장치가 A/D 변환을 위한 입력 데이터로 사용한다. 이 제어장치에 의해 디지털로 변환된 데이터는 블루투스 통신으로 테블릿에 전송되어 앱 프로그램의 입력 데이터로 사용된다. 앱 프로그램은 이 데이터를 딥-러닝 엔진 학습에 필요한 입력 데이터로 사용하거나, 출력전압의 일정한 패턴을 검증하는 입력 데이터로 사용한다.

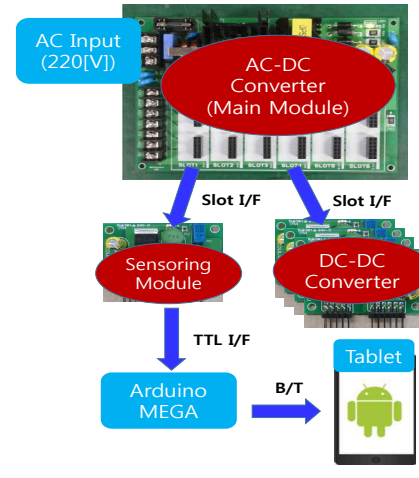

그림 1. 스마트 전원장치의 구성도
Fig. 1. Block diagram of smart SMPS

2.1 하드웨어

그림 2는 SolidWorks로 제작한 스마트 전원장치의 3D 도면으로써 테블릿 거치대는 별도로 구성한다. 내부에는 스마트 전원장치에서 발생하는 열을 방출시키기 위한 쿨링팬을 포함한 모든 부품들을 내장하고 있다.

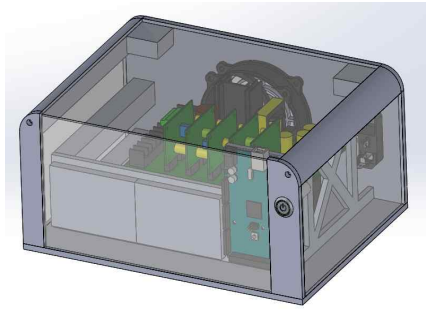

그림 2. 스마트 전원장치의 캐드 도면
Fig. 2. CAD drawing of smart SMPS

그림 3은 UC2844 PWM칩을 사용한 플라이백 방식으로 구동하는 스마트 전원장치의 AC-DC 컨버터 회로도를 나타낸다. AC 정격입력 전압은 220V, 입력전압의 허용 범위는 AC 180V ~ AC 240V, 출력전압은 24V ~ 25V, 최대 출력전류는 2.5A, 최대 출력전력은 60W이다. FET는 스위칭 주파수 78KHz에서 스위칭 동작을 하고, 회로의 보호를 위해 1차 전류 입력 단에 250V, 3A의 휴즈를 사용하고 별도로 정격전류의 150%이상에서 동작하는 과전류 보호회로를 내장한다. 플러그인 방식으로 슬롯에 탈착이 가능한 DC-DC 컨버터 모듈은 +5V, +15V, -15V, 가변전압 등 4가지 종류로 구성하고, 각 모듈의 출력전류는 최대 1A로 제한한다. +5V, 3A의 출력을 요구하는 부하에는 +5V DC-DC 컨버터 모듈 3장을 슬롯에 장착하여 사용한다. 또한 동일한 방식으로 슬롯에 탈착이 가능한 전압 센싱 모듈은 DC-DC 컨버터에서 발생하는 전압을 입력 데이터로 사용하여 0V ~ 3V 범위에서 선형적으로 변환한다. 이 데이터는 아두이노 제어장치의 아날로그 포트에 인가된 후에 디지털 신호로 변환되어 UART I/F로 접속된 블루투스 모듈을 통해 테블릿으로 전송된다.

2.2 소프트웨어

스마트 전원장치의 아날로그 출력전압은 제어장치의 A/D 컨버터를 통해 디지털 데이터로 변환된다. 태블릿에서 실행되는 비동기 데이터 수신용 쓰래드를 통해 버퍼에 이 데이터가 저장되어 딥-러닝 엔진의 학습에 필요한 입력 데이터로 사용되거나, 전원장치의 비정상적인 동작을 검증하는 입력 데이터로 사용한다.

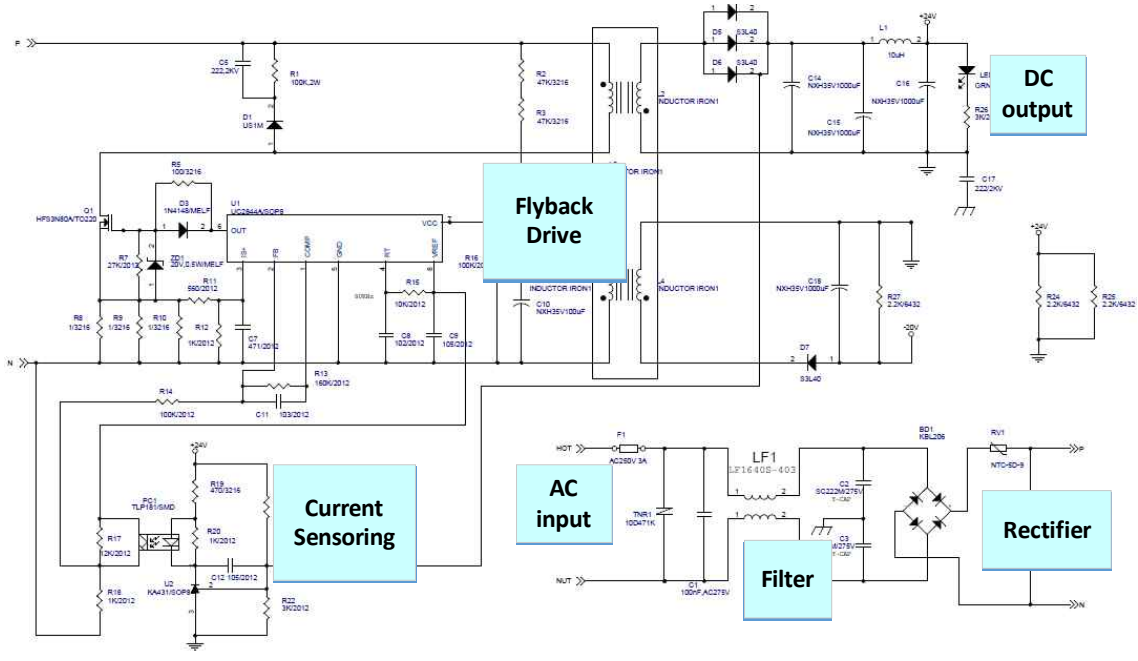

그림 3. 스마트 전원장치의 회로도
Fig. 3. Schematic diagram of smart SMPS

수집된 20개 연속적인 데이터를 1개 데이터 셋으로 정의한 후에, 20개 전체 평균값 1개, 20개 데이터를 5개 그룹으로 분할한 후에 각 그룹 평균값 5개 및 각 그룹 변위를 나타내는 미분값 4개, 20개 데이터를 2개 그룹으로 분할 한 후에 각 그룹 평균값 2개 및 각 그룹 변위를 나타내는 미분값 1개를 포함한 총 13개 값을 생성한다. 이 값들을 정규화한 후에 원핫 인코딩(One-hot Encoding)으로 9단계의 값으로 분류하여 딥-러닝 엔진의 입력뉴런 값으로 사용한다. 딥-러닝 엔진은 117개의 뉴런을 포함한 입력층 1개, 각 40개의 뉴런을 포함한 은익층 2개, 9개의 뉴런을 포함한 출력층 1개로 구성하고, 출력함수로는 은익층 및 출력층에 공통으로 시그모이드(Sigmoid) 함수를 사용한다.

그림 4에 나타낸 것과 같이 강화학습에서는 1번의 에피소드로 80개의 연속적인 입력신호를 사용하여 4개의 Q-값 배열을 생성하고, 이 값을 기준으로 리위드(Reward)를 결정하여 딥-러닝 엔진을 학습시킨다. 딥-러닝 엔진을 학습시키기 위한 앱 프로그램은 두 가지 학습모드를 제공한다.

첫 번째로 딥 신경망인 딥 모드는 딥-러닝 엔진에 랜덤 샘플링으로 생성한 입력 데이터 셋을 인가하여 역전파 알고리즘을 통해 학습을 진행한다. 이 모드가 실행되는 절차는 다음과 같다.

① 그림 5에 나타낸 것과 같이 9가지 종류의 20개 입력 데이터 셋을 랜덤 샘플링으로 생성한다.

② 이 값을 전처리 과정을 통해 딥-러닝 엔진에 사용될 입력뉴런 값 117개로 변환한다.

③ 딥-러닝 엔진을 실행하여 출력층의 9개 Q-값을 순차적으로 그림 5의 9가지 종류의 데이터 입력 셋과 일대일로 대응시킨다.

④ 입력 데이터 셋에 대응하는 Q-값이 포함된 출력 뉴런만을 선택하여 목표 값을 결정한 후에 역전파 알고리즘을 통해 딥-러닝 엔진을 학습시킨다.

⑤ 주기적으로 스마트 전원장치의 출력전압을 “Normal Mode”, “Low Mode”, “High Mode”가 되도록 설정하여, 여기서 발생한 실제 전압 값을 입력 데이터 셋으로 사용하여 딥-러닝 엔진을 추가 학습시킨다. 이 과정을 실행하여 딥-러닝 엔진이 랜덤 샘플링에 의해 발생되는 학습 오류를 보상한다.

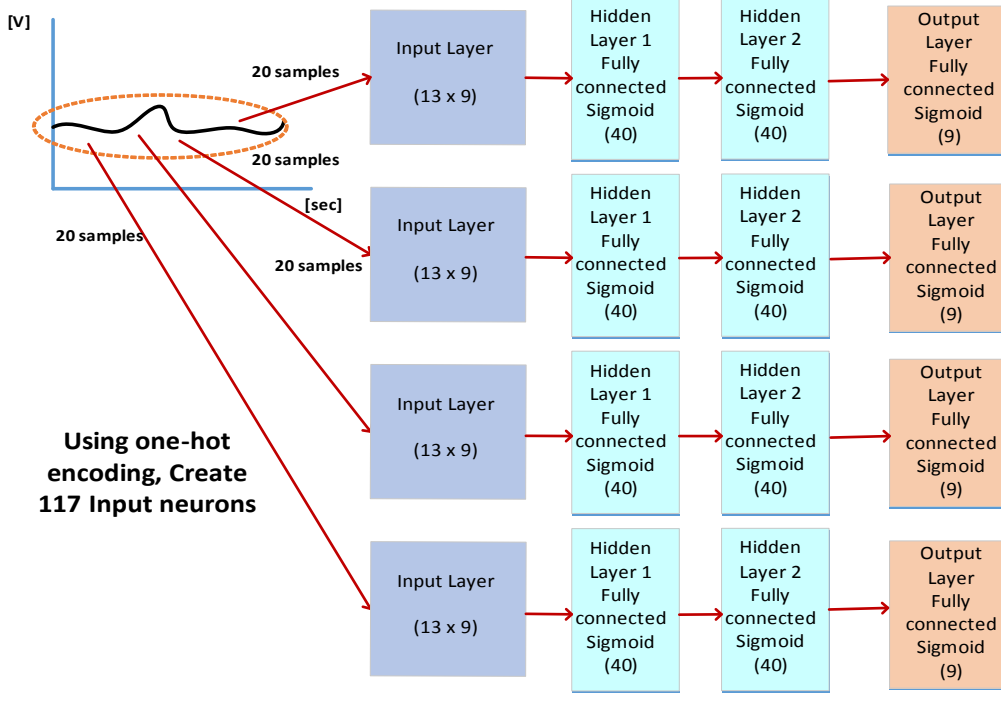

그림 4. 딥-러닝 학습 구조

Fig. 4. Architecture of deep learning policy

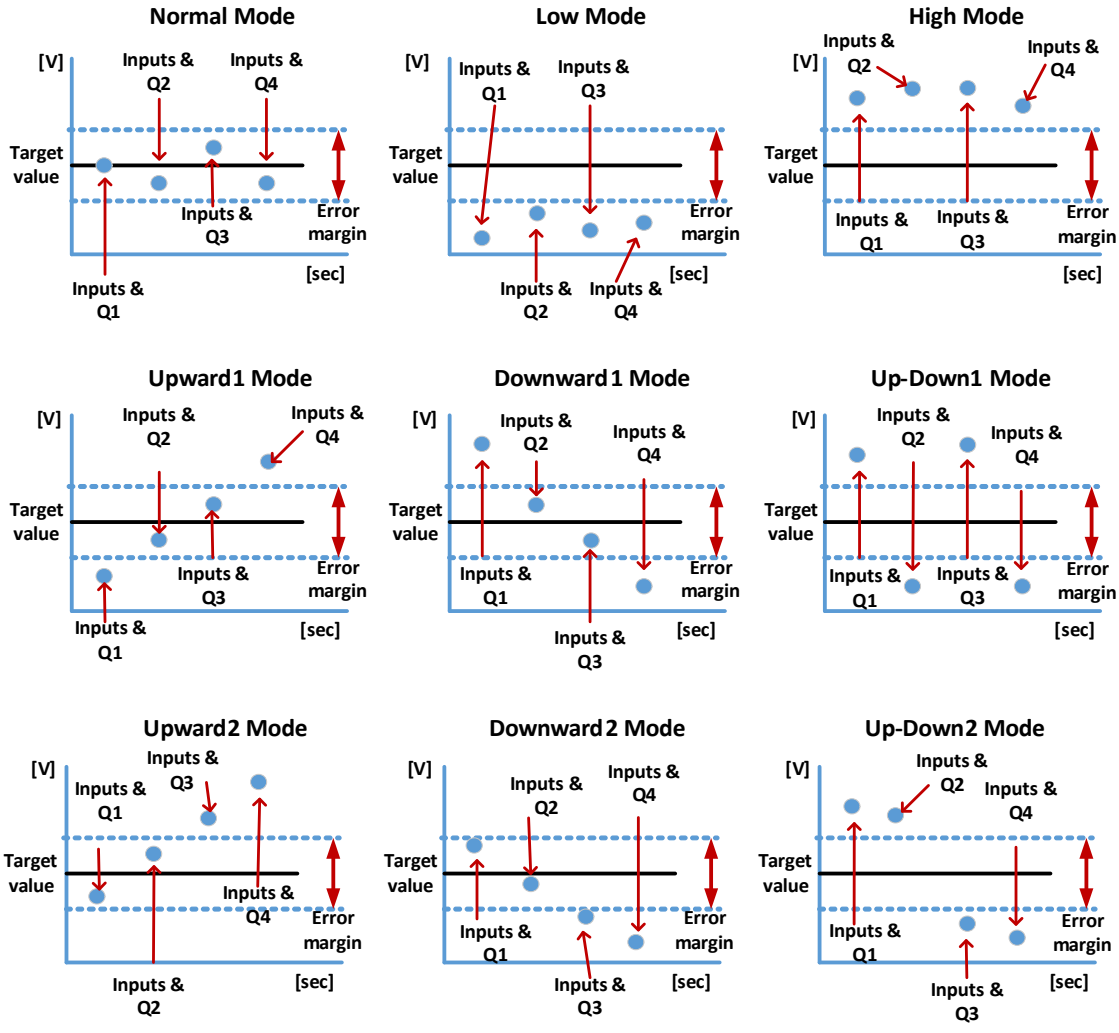

그림 5. 입력 데이터 셋 패턴
Fig. 5. Patterns of input data set

두 번째로 강화학습인 딥-Q 모드는 딥 모드로 학습한 딥-러닝 엔진에 Q-러닝 알고리즘으로 생성한 입력 데이터 셋을 인가하여 학습을 진행한다[9][10] 이 모드가 실행되는 절차는 다음과 같다.

$$P_a = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_{i=1}^{N} e^{\frac{Q(s,i)}{T}}} \quad (1)$$

① 딥 모드의 출력인 Q-값을 식 (1)의 볼츠만 분포 함수에 대입하여 각 뉴런의 확률분포를 계산한 후에, 랜덤 샘플링으로 13개의 값을 생성한다,

② 딥-러닝 엔진에 이 값들을 첫 번째 상태의 입력 데이터 셋으로 인가하여 첫 번째 상태의 Q-값을 결정한다.

③ 동일한 과정으로 두 번째, 세 번째, 네 번째 상태의 입력 데이터 셋을 생성하여, 두 번째, 세 번째 및 네 번째 상태의 Q-값을 결정한다.

④ 각각의 셋에서 Q-값을 저장된 배열에서 최대값을 나타내는 인덱스 번호와 이에 해당하는 입력 데이터 셋 모드의 번호를 비교한다.

⑤ 두 개의 번호가 3개 이상 일치하는 경우의 에

피소드를 최적의 정책으로 선택하여 “+” 리워드 값을 부여하고, 3개 이상 불일치하는 경우는 최악의 정책으로 선택하여 “-” 리워드 값을 부여한다.

⑥ 이 리워드를 식 (2)의 Q-러닝 함수에 대입하여 새로운 Q-값을 결정한 후에, 역전파 알고리즘 실행하여 딥-러닝 엔진을 학습시킨다.

$$Q_{NXT} = Q_{CUR} + \alpha[R + \gamma Q_{MAX} - Q_{CUR}] \quad (2)$$

여기서 α는 학습률(Learning Rate), γ는 보상계수(Discount Factor), R은 리워드를 나타낸다.

첫 번째 딥 모드는 원핫 인코딩으로 생성한 데이터 셋을 입력뉴런에 인가하여 역전파 알고리즘을 통해 출력뉴런과의 상관관계를 유지토록 학습하는 과정이다. 입력 데이터 셋과 관련이 없는 출력뉴런에 대한 역전파 알고리즘에 의한 학습은 진행하지 않는다. 따라서 이 과정을 통해 부하가 접속된 스마트 전원장치의 출력전압을 변환하여 입력뉴런 값으로 인가한 후에 딥-러닝 엔진을 통해 얻어지는 최대 Q-값의 인덱스 번호와 입력 데이터 셋의 번호가 낮은 상관관계를 유지토록 한다.

두 번째 딥-Q 모드는 강화학습을 통해 두 종류의 인덱스 번호가 3번 이상 일치하는 경우에 최적의 에피소드로 간주하고 “+” 리워드 값을 부여하므로써, 딥-러닝 엔진이 효과적으로 학습되도록 한다. 부하가 접속된 상태에서 스마트 전원장치의 출력전압을 모니터링하는 과정은 학습이 완료한 딥-러닝 엔진의 하이퍼 파라메터를 저장한 파일을 호출하여 딥-러닝 엔진을 구축한 후에 진행한다. 실행되는 절차는 다음과 같다.

① 블루투스 통신 쓰래드를 통해 1개의 에피소드에 해당하는 80개의 출력전압을 샘플링하여 버퍼에 저장한다.

② 버퍼에 저장된 80개의 데이터로부터 4개의 연속적인 입력뉴런 값을 생성한 후에 순차적으로 딥-러닝 엔진을 실행하여 출력 Q-값들을 결정한다.

③ 4개의 연속된 Q-값으로 그림 5에 나타낸 입력 데이터 셋의 패턴과 일치하는 모드를 결정한다.

이 모드 값들이 변화하는 과정을 통해 부하의 적합성 여부를 판별하거나, 또는 스마트 전원장치가 추후에 고장이 발생할 정도를 예측할 수 있는 데이터로 사용할 수 있다.

III. 실험 결과

본 연구에서 제작한 스마트 플러그인 스위칭 모드 전원장치는 그림 6과 같다. AC-DC 전원장치 및 5종류의 모듈형 보드, 아두이노 제어장치 및 제어회로는 내부에 배치하였고, 태블릿은 외부의 지지대에서 별도로 사용할 수 있도록 하였다.

그림 7은 딥-러닝 엔진 학습 및 입력 데이터 패턴을 검증할 수 있는 기능이 있는 앱 프로그램을 태블릿에서 실행한 결과를 나타내었다. 화면 왼쪽에는 상하의 순서로 스마트 전원장치에 탑재된 블루투스 모듈과의 통신을 설정하는 이미지 버튼, 딥 모드 학습을 실행시키는 이미지 버튼, 딥-Q 모드 학습을 실행시키는 이미지 버튼, 학습이 완료된 딥-러닝 엔진을 파일로 출력하는 이미지 버튼, 입력 데이터 패턴의 검증을 위해 학습된 딥-러닝 엔진을 호출하는 이미지 버튼으로 구성하였다.

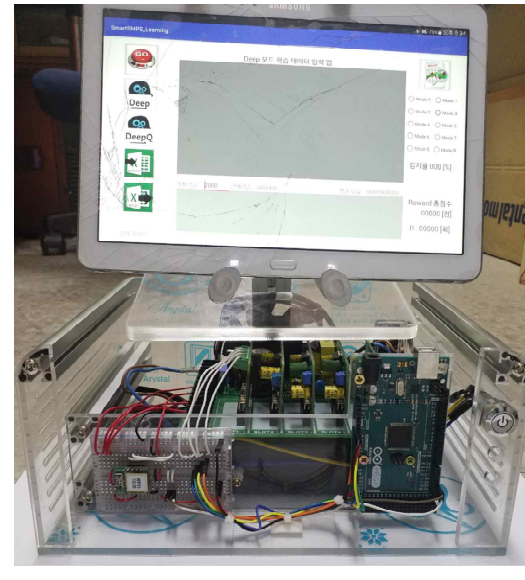

그림 6. 스마트 플러그인 스위칭 모드 전원장치
Fig. 6. Smart plug-in SMPS

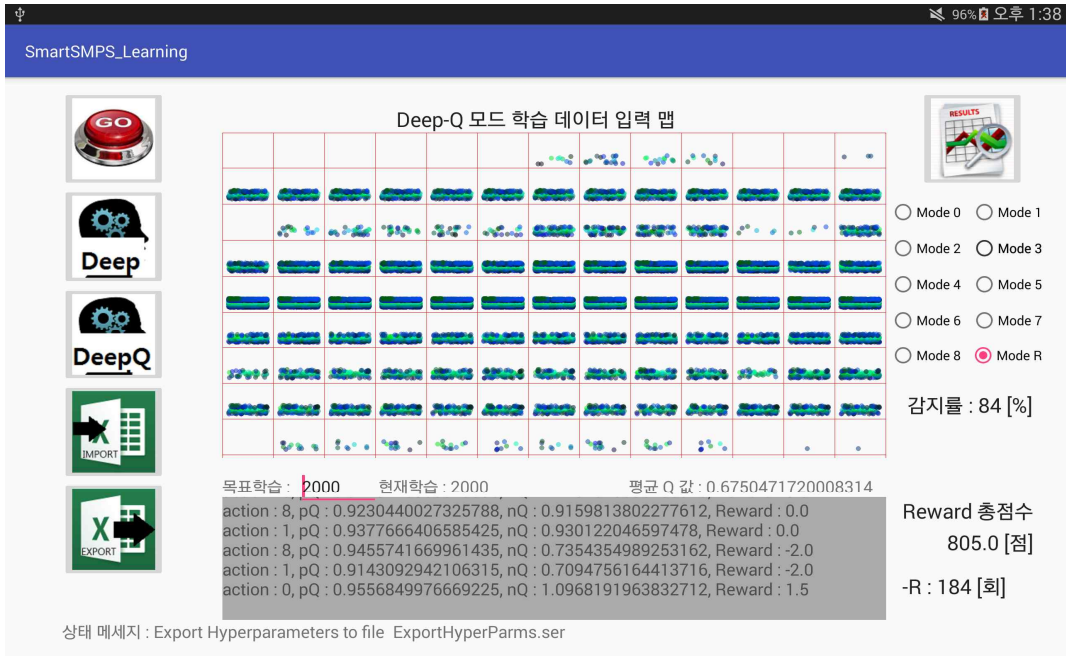

그림 7. 앱 프로그램 실행 결과
Fig. 7. Executing result of app. program

화면 중간의 상단에 위치한 이미지 뷰로 구성한 학습 데이터 입력 맵은 원핫 인코딩 방식을 적용하여 스마트 전원장치에서 발생한 검증용 입력 데이터 또는 랜덤 샘플링으로 발생된 학습용 입력 데이터를 9x13 배열의 2차원 맵으로 표시하여 데이터의 분포를 확인할 수 있도록 하였다. 반면에 하단에 위치한 텍스트 뷰는 프로그램이 실행되는 결과에 대한 모니터링, 목표학습 및 현재학습 횟수, 평균 Q-값 등을 확인할 수 있도록 하였다. 화면 오른쪽 상단에 데이터 검증용 이미지 버튼은 10개의 라디오 버튼의 선택 결과에 따라서 해당되는 입력 모드를 검증토록 하였고, 하단에 텍스트 뷰는 감지률과 딥-Q 모드가 실행되는 경우에 리워드 값을 확인할 수 있도록 하였다.

딥-러닝 엔진을 생성하기 위한 실험은 다음과 같은 절차로 진행하였다.

① 앱 프로그램을 실행시켜 랜덤 모드로 초기화된 하이퍼 파라메터를 갖는 비학습 상태의 딥-러닝 엔진을 생성한다.

② 목표학습을 2000회로 설정한 후에 딥 모드 학습을 실행한다. 역전파 알고리즘은 1회 학습당 최대 100번의 반복 실행을 하거나, “0.001”의 오차 범위 이내로 출력뉴런 값이 결정되면 다음 학습을 진행한다.

③ 현재 실행 중인 앱 프로그램을 종료하고 재실행시킨 후에, 이전 단계학습에서 하이퍼 파라메터를 저장한 파일을 호출하여 새로운 딥-러닝 엔진을 생성한다.

④ 목표학습을 2000회로 설정한 후에 딥-Q 모드 학습을 실행한다. Q-러닝 알고리즘을 통해 결정된 출력뉴런 값에 따라서 학습률을 향상시키기 위해 역전파 알고리즘은 1회 실행한다.

⑤ 동일한 과정으로 딥 모드와 딥-Q 모드 학습을 교대로 진행한다.

이와 같은 실험을 진행하여 비학습 상태 엔진을 포함한 20000회 학습, 26000회 학습, 32000회 학습, 38000회 학습, 46000회 학습 상태의 6개 딥-러닝 엔진을 생성하였다. 스마트 전원장치를 통해 9가지 종류의 입력 데이터 패턴 모두를 생성하는 것이 용이하지 않아서 “mode 0” 및 “mode 1”에 해당하는 일부 입력 데이터 패턴만을 생성시켰다.

표 1. 입력 패턴 감지률
Table 1. Detection ratio of input pattern

Learning Method	Iteration Numbers		Detection Mode									Mode Dectection Rate
			mode 0	mode 1	mode 2	mode 3	mode 4	mode 5	mode 6	mode 7	mode 8	
Random mode (no learning)	1	Input A	31	13	16	25	30	25	21	18	21	6.5
		Input B	3	0	0	4	3	0	0	0	3	
		Ave. Q	0.91544	NaN	NaN	0.914006	0.930292	NaN	NaN	NaN	0.920931	
	2	Input A	29	26	24	15	24	22	21	19	20	5.5
		Input B	4	0	0	1	1	0	0	0	5	
		Ave. Q	0.896881	NaN	NaN	0.896529	0.923664	NaN	NaN	NaN	0.93632	
	3	Input A	21	13	25	26	24	17	23	30	21	6
		Input B	0	0	0	6	3	0	0	1	2	
		Ave. Q	NaN	NaN	NaN	0.899958	0.917386	NaN	NaN	0.910725	0.940833	
	4	Input A	24	22	16	17	27	22	30	23	19	4.5
		Input B	2	2	0	1	2	0	0	0	2	
		Ave. Q	0.910515	0.927812	NaN	0.909649	0.933181	NaN	NaN	NaN	0.935853	
	5	Input A	20	11	27	27	23	22	19	25	26	3.5
		Input B	6	0	0	0	1	0	0	0	0	
		Ave. Q	0.927221	NaN	NaN	NaN	0.916896	NaN	NaN	NaN	NaN	
	Ave. Rate		12	2.35	0	10.91	7.81	0	0	0.87	11.21	5.2
20000 times learning (Deep mode 10000 DeepQ mode 10000)	1	Input A	18	30	21	23	19	26	22	29	12	54
		Input B	1	30	0	22	0	0	22	29	4	
		Ave. Q	0.97763	0.987792	NaN	0.980276	NaN	NaN	0.982154	0.983862	0.973357	
	2	Input A	15	24	15	22	26	25	21	33	19	46.5
		Input B	2	23	0	19	0	0	21	26	2	
		Ave. Q	0.983154	0.988579	NaN	0.980269	NaN	NaN	0.98091	0.982147	0.976737	
	3	Input A	18	20	30	15	24	24	25	19	25	40
		Input B	2	19	0	12	0	1	24	18	4	
		Ave. Q	0.983817	0.988054	NaN	0.977252	NaN	0.966501	0.981625	0.983961	0.973405	
	4	Input A	17	26	18	18	34	24	18	22	23	42
		Input B	1	26	0	14	0	0	18	20	5	
		Ave. Q	0.978307	0.987609	NaN	0.979759	NaN	NaN	0.982666	0.983359	0.973323	
	5	Input A	14	22	28	23	26	14	21	23	29	48
		Input B	2	22	0	21	0	0	21	22	8	
		Ave. Q	0.982247	0.98677	NaN	0.979643	NaN	NaN	0.981886	0.982697	0.975272	
	Ave. Rate		9.76	98.36	0	87.13	0	0.88	99.07	91.27	21.3	46.1
26000 times learning (Deep mode 14000 DeepQ mode 10000)	1	Input A	21	22	21	21	17	18	32	28	20	64.5
		Input B	6	21	20	19	3	4	22	26	8	
		Ave. Q	0.680951	0.735412	0.688095	0.632227	0.633288	0.625394	0.621112	0.776734	0.626793	
	2	Input A	25	19	24	23	21	24	22	20	22	63
		Input B	10	18	24	18	2	14	12	18	10	
		Ave. Q	0.655725	0.743288	0.724463	0.631773	0.648869	0.622846	0.623813	0.742773	0.63007	
	3	Input A	26	17	24	27	19	20	22	17	28	61
		Input B	8	17	23	19	2	5	19	17	12	
		Ave. Q	0.667155	0.750177	0.710945	0.620739	0.615554	0.605996	0.622543	0.76913	0.635802	
	4	Input A	21	25	32	19	23	20	23	22	15	72.5
		Input B	10	24	31	17	6	10	18	22	7	
		Ave. Q	0.648331	0.7184	0.697037	0.625378	0.628884	0.625947	0.628115	0.744369	0.633429	
	5	Input A	21	17	23	27	15	27	18	24	28	65.5
		Input B	11	16	23	23	4	10	11	22	11	
		Ave. Q	0.66422	0.775104	0.706004	0.624475	0.622359	0.612903	0.636761	0.758247	0.637285	
	Ave. Rate		39.47	96	97.58	82.05	17.89	39.45	70.09	94.59	42.48	65.3

표 2. 입력 패턴 감지률

Table 2. Detection ratio of input pattern

Learning Method	Iteration Numbers		Detection Mode									Mode Dectection Rate
			mode 0	mode 1	mode 2	mode 3	mode 4	mode 5	mode 6	mode 7	mode 8	
32000 times learning (Deep mode 20000 DeepQ mode 12000)	1	Input A	17	26	23	30	20	23	18	24	19	67.5
		Input B	11	26	22	22	0	14	13	16	11	
		Ave. Q	0.636625	0.778436	0.714374	0.615731	NaN	0.607846	0.608144	0.714026	0.635215	
	2	Input A	26	17	22	18	16	16	28	32	25	67.5
		Input B	16	16	19	12	0	14	23	26	9	
		Ave. Q	0.624446	0.766682	0.715665	0.605243	NaN	0.619205	0.606568	0.740037	0.633319	
	3	Input A	25	19	26	21	23	19	25	20	22	71.5
		Input B	14	19	26	18	0	15	20	18	13	
		Ave. Q	0.613724	0.784467	0.712621	0.613407	NaN	0.607923	0.608099	0.731519	0.647782	
	4	Input A	31	23	23	15	21	20	19	21	27	63.5
		Input B	17	23	21	12	0	16	8	15	15	
		Ave. Q	0.621531	0.783397	0.70278	0.626715	NaN	0.614941	0.605252	0.702217	0.632108	
	5	Input A	23	35	16	22	19	26	28	16	15	69.5
		Input B	13	35	16	17	0	20	12	16	10	
		Ave. Q	0.625286	0.801552	0.725041	0.610564	NaN	0.605074	0.601148	0.746094	0.64617	
	Ave. Rate		58.2	99.17	94.55	76.42	0	75.96	64.41	80.53	53.7	67.9
38000 times learning (Deep mode 24000 DeepQ mode 14000)	1	Input A	19	21	20	25	29	14	26	27	19	65.5
		Input B	13	18	20	21	13	7	7	25	7	
		Ave. Q	0.634676	0.724931	0.743807	0.637289	0.595676	0.593672	0.599921	0.7262	0.605324	
	2	Input A	30	16	28	16	23	19	24	23	21	72.5
		Input B	20	16	28	13	12	15	8	23	10	
		Ave. Q	0.666171	0.72114	0.720364	0.638679	0.595897	0.603135	0.6102	0.72409	0.610986	
	3	Input A	18	13	26	25	25	28	25	23	17	60
		Input B	11	12	25	15	4	14	9	23	7	
		Ave. Q	0.648697	0.726483	0.718665	0.634189	0.582168	0.59828	0.606555	0.729142	0.629718	
	4	Input A	33	22	21	20	22	20	23	25	14	65.5
		Input B	21	21	21	15	8	10	9	22	4	
		Ave. Q	0.665699	0.735335	0.717036	0.64266	0.597818	0.60112	0.605502	0.736687	0.621	
	5	Input A	25	21	19	21	16	26	19	26	27	66.5
		Input B	17	20	19	15	7	16	7	23	9	
		Ave. Q	0.658502	0.755021	0.720394	0.641769	0.600357	0.597673	0.608804	0.716047	0.59404	
	Ave. Rate		65.6	93.55	99.12	73.83	38.26	57.94	34.19	93.55	37.76	66
46000 times learning (Deep mode 30000 DeepQ mode 16000)	1	Input A	16	24	23	24	26	19	17	21	30	54
		Input B	12	18	21	14	16	0	0	14	13	
		Ave. Q	0.672534	0.66399	0.709475	0.627526	0.625694	NaN	NaN	0.711222	0.637534	
	2	Input A	17	25	26	25	18	24	15	30	20	53.5
		Input B	15	23	20	12	13	0	0	14	10	
		Ave. Q	0.669277	0.671828	0.688576	0.619023	0.62308	NaN	NaN	0.692315	0.640253	
	3	Input A	27	24	24	30	23	16	17	22	17	58
		Input B	23	21	19	17	10	0	2	17	7	
		Ave. Q	0.681531	0.668091	0.711169	0.627283	0.630853	NaN	0.595065	0.70142	0.643885	
	4	Input A	29	24	16	22	23	23	23	20	20	54
		Input B	25	21	13	9	15	0	0	16	9	
		Ave. Q	0.684295	0.67555	0.714211	0.636979	0.618519	NaN	NaN	0.707149	0.664338	
	5	Input A	18	27	28	25	23	18	22	18	21	54.5
		Input B	15	24	24	15	10	0	0	12	9	
		Ave. Q	0.664772	0.683497	0.713232	0.629243	0.615856	NaN	NaN	0.670084	0.647169	
	Ave. Rate		84.11	86.29	82.91	53.17	56.64	0	2.13	65.77	44.44	54.8

표 1 및 표 2는 6개 딥-러닝 엔진을 사용하여 9가지 종류의 입력 데이터를 200개 생성하여, 이 입력 데이터 패턴의 감지률을 측정한 결과를 나타내었다. 여기서 “Input A”는 딥-러닝 엔진에 인가한 입력이고, “Input B”는 딥-러닝 엔진에 의하여 감지된 입력을 나타낸다. 본 실험을 통해 딥-러닝 엔진의 학습 횟수를 증가시키면 전원장치의 정상상태를 나타내는 “mode 0” 입력 데이터 패턴에 대한 감지률이 9.76%에서 84.11%까지 증가함을 확인할 수 있었다. 반면에 모든 종류의 입력 데이터 패턴에 대한 모드 감지률은 최소 5.2%에서 최대 67.9%까지 증감이 반복됨을 확인할 수 있었다. 또한 “mode 1” 및 “mode 2” 입력 데이터에 대한 감지률은 26000회 이상의 학습을 진행하면 82%이상을 유지함을 확인할 수 있었다. 이 결과는 통해 지도 학습 기반의 딥-러닝 학습과 강화 학습 알고리즘을 혼용하여 구현된 딥-러닝 엔진을 사용하면 전원장치의 특이한 상태를 감지할 수 있음을 확인할 수 있다.

Ⅳ. 결 론

본 연구에서는 AC-DC 컨버터, 플러그인 방식으로 탈착이 가능한 4 종류의 DC-DC 컨버터 모듈 및 전압 센싱 모듈, 아날로그 출력전압을 디지털 신호로 변환하여 태블릿으로 전송하는 제어장치 프로그램, 태블릿에서 실행되는 딥-러닝 엔진을 내장한 앱 프로그램 등을 포함하는 스마트 플러그인 스위칭 모드 전원장치를 제작하였다. 실험 결과를 통해 이 전원장치는 다음과 같은 특징을 지니고 있음을 알 수 있었다.

첫째로 부하에 인가된 출력전압에 대한 모니터링을 통해 진단 기능을 제공함과 동시에 플러그인 방식으로 탈착이 가능한 구조로 제작되어 있어 부하에 필요한 전원만을 공급하는 효율적인 전원장치로 사용할 수 있다.

둘째로 46000회 학습을 통해 “mode 0”인 정상상태를 평균 84.11%로 감지할 수 있는 딥-러닝 엔진을 사용할 수 있다. 하지만 이 엔진은 전체 입력 데이터에 대한 모드 감지률이 54.8%로 높지 않은 수준이다. 따라서 전체 모드 감지률을 향상시키기 위해 새로운 방식의 딥-러닝 엔진 구조를 사용하거나, 하이퍼 파라메터를 정교하게 설정하여 반복적인 실험을 진행할 필요가 있다.

셋째로 동작 상태를 실시간으로 학습 또는 진단하기 위한 과정은 아두이노 제어장치의 A/D 변환 처리에 의한 지연, 블루투스 통신에 의한 데이터 전송 지연, 앱 프로그램의 역전파 알고리즘 실행에 의한 지연 등으로 본 연구에서 제안한 하드웨어 구조로는 용이하지 않는 경우가 있다. 일반적으로 클라우드 기반의 상용 딥-러닝 플랫폼을 사용하면 간편하게 전체 시스템을 구축할 수 있지만 플랫폼에 대한 의존성이 높은 단점이 있다. 반면에 임베디드 시스템 전용으로 출시된 “ENVIDIA TX2”와 유사한 제어장치를 사용하면 독립된 환경에서 고유한 특성을 지닌 딥-러닝 엔진의 설계가 가능할 뿐만 아니라, 실시간 제어가 가능한 전체 시스템을 제작할 수 있다.

향후에 본 연구에서 개발한 앱 프로그램을 2개의 모듈로 분리해서 “ENVIDIA TX2”와 같은 인공지능용 제어장치에서는 딥-러닝 엔진 학습 및 진단 기능을 제공하는 모듈을 실행시키고, 태블릿은 U/I용 모듈을 실행토록 시스템 구조를 변경하면 딥-러닝 기반의 진단 기능을 탑재한 새로운 전원장치를 개발할 수 있다. 특히 대용량의 신뢰성을 요구하는 시스템에 사용되는 전원장치에 이 기능을 탑재되면 상품으로써 가치가 충분히 있을 것이다.

References

[1] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel, "End-to-End Training of Deep Visuomotor Policies", Journal of Machine Learning Research, No. 17, pp. 1-40, Oct. 2016.

[2] Dong-Ha Shin and Chang-Bok Kim, "A Study on Deep Learning Input Pattern for Summer Power", Journal of KIIT, Vol. 14, No. 11, pp. 127-134, Nov. 2016.

[3] Ho-Bum Song and Jun Jeong, "The Implementation of smart DNC depaneling router", Journal of KIIT, Vol. 15, No. 3, pp. 11-22, Mar.

2017.

[4] Wei Li, Matthias Breier, and Til Aach, "Vision-based Auto-Teaching for automated PCB depaneling", Industrial Informatics, 2012 10th IEEE International Conference, pp. 910-915, Sep. 2012.

[5] Lerrel Pinto and Abhinav Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours", Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 3406-3413, May 2016.

[6] Alonso Marco, Felix Berkenkamp, Philipp Hennig, Angela P. Schoellig, Andreas Krause, Stefan Schaal, and Sebastian Trimpe, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization", Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 1557-1563, May 2017.

[7] Rejin John Varghese, Pierre Berthet-Rayne, Petros Giataganas, Valentina Vitiello, and Guang-Zhong Yang, "A framework for sensorless and autonomous probe-tissue contact management in robotic endomicroscopic scanning", Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 1738-1745, May 2017.

[8] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei1, and Ali Farhadi, "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning", Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 3357-3364, May 2017.

[9] M. Deisenroth, G. Neumann, and J. Peters, "A survey on policy searchfor robotics", Foundations and Trends in Robotics, Vol. 2, No. 1-2, pp. 1-142, 2013.

[10] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates", Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 3389-3396, May 2017.

Authors

송 호 범 (Ho-Bum Song)

1984년 2월 : 서울대학교 전기공학과(공학사)
1986년 2월 : 서울대학교 전기공학과(공학석사)
1986년 ~ 1994년 : 삼성전자 및 삼성종합기술원 선임 연구원
2005년 : 성균관대학교 정보통신 공학부 박사 수료
1995년 3월 ~ 현재 : 동양미래대학교 로봇자동화공학부 교수
관심분야 : 인공지능, 임베디드 시스템, 모바일 프로그램

정 준 (Jun Jeong)

1994년 2월 : 연세대학교 기계공학과(공학사)
1996년 8월 : 연세대학교 기계공학과(공학석사)
2001년 8월 : 연세대학교 기계공학과(공학박사)
2002년 ~ 2005년 : 삼성전자 스토리지사업부 책임연구원
2006년 3월 ~ 현재 : 동양미래대학교 로봇자동화공학부 부교수
관심분야 : 임베디드 시스템, 자동제어