

모바일 응용 프로그램들의 전력 소모 정보
제공 서비스

Power scanner

목 차

1. 서 론	1
2. 파워스캐너 프레임워크	3
2.1. 파워스캐너 프레임워크 구조	3
2.2. 파워스캐너 응용프로그램 구조	4
3. 파워스캐너 응용프로그램	6
3.1. 유즈케이스 다이어그램	6
3.2. 클래스 다이어그램	11
3.3. 시퀀스 다이어그램	30
3.4. 응용프로그램 실행 화면	35
4. 웹서버 및 데이터베이스	36
4.1. 웹 서버	36
4.2. 데이터베이스	38
5. 파워스캐너 테스트	40
5.1. 테스트 목적	40
5.2. 테스트 환경	40
5.3. 테스트 결과	40
6. 결 론	43
7. 참고문헌	45

1. 서 론

스마트폰의 하드웨어·소프트웨어 성능의 발전으로 사람들은 컴퓨터를 사용해서 하던 작업들을 스마트폰으로 처리하고 있다. 이런 높은 성능의 스마트폰도 배터리를 사용하는 유한한 전력 환경에서 동작하고 있다.

현재 안드로이드 마켓에 등록된 모든 응용 프로그램이 동일한 전력 소모량을 갖지 않는다. 그렇기 때문에 더 오랜 시간동안 스마트폰을 사용하기 위해서는 전력 소모량이 적은 응용 프로그램을 선택할 필요가 있다. 뿐만 아니라 필요 이상으로 많은 전력을 소비하는 응용 프로그램의 경우에는 사용자가 스마트폰을 사용하는데 있어서 큰 피해를 주게 된다. 사용자에게 피해를 주는 바이러스, 악성코드의 경우에는 백신과 같은 응용 프로그램이 존재해서 그 피해를 사전에 예방할 수 있다. 하지만 전력 소모에 대해서는 사전에 응용 프로그램의 전력 소모 정보를 제공하지 않기 때문에 사용자들의 피해와 불편함을 예방하지 못하고 있다.

기존의 연구들은 단순히 로컬 환경에서 전력 소모량 측정만을 제공하며 사용자에게 서비스를 제공하지 않는다. 그러므로 사용자가 전력 소모 정보를 알기 위해서는 많은 응용 프로그램의 전력 소모 정보를 직접 수집해야하는 어려움과 수집된 정보를 이용하여 유효한 결과를 도출해내는 데에 어려움이 있다. 그리고 새로운 응용 프로그램을 선택할 때 유사한 기능을 하는 응용 프로그램들을 매년 측정해야 하는 불편함이 있다.

본 연구에서는 이러한 문제점들을 해결하기 위해 로컬 환경에서 측정하는 것뿐만 아니라, 응용 프로그램 별 전력 소모 정보 데이터베이스 구축을 통한 파워스캐너 프레임워크를 제안한다. 본 연구를 통해서 사용자는 측정한 전력 소모 데이터를 서버로 전송할 수 있고 전력 소모 정보가 필요한 경우 별도의 측정 없이 서버로부터 정보를 받아서 확인할 수 있다. 본 연구를 진행하기 위해서 기존에 진행된 연구 결과인 Power Tutor를 활용하는 방법을 제안한다. Power Tutor는 스마트 폰에서 실행중인 응용 프로그램의 전력 소모 정보를 측

정하는 기능을 갖고 있다. 파워스캐너 프레임워크는 Power Tutor의 측정 기능을 활용해서 스마트 폰에서 실행 중인 응용 프로그램들의 전력 소모량을 측정한다. 또한 측정된 결과를 데이터로 저장하며, 데이터베이스가 구축된 서버로 데이터 전송을 가능하게 할 것이다. 위 과정을 통해 수집한 전력 소모 정보를 단순히 데이터베이스에 보관하는 것이 아니라, 사용자에게 필요한 정보로 가공하여 제공할 수 있는 환경을 만드는 것이 목표이다.

본 연구의 결과를 도출하기 위해서 40개 이상의 응용 프로그램에 대한 전력 소모 데이터를 수집하여 응용 프로그램들 사이에 전력 소모량의 차이가 있음을 확인할 것이며 동일 유형의 응용 프로그램 사이에서도 그 차이가 있음을 보일 것이다.

2. 파워스캐너 프레임워크

본 장에서는 전력 소모 정보 제공을 위한 클라이언트 어플리케이션, 서버, 데이터베이스를 포함하는 파워스캐너 프레임워크와 클라이언트 어플리케이션인 파워스캐너 어플리케이션에 대해서 소개한다.

- 파워스캐너 프레임워크
- 파워스캐너 어플리케이션 (안드로이드)

2.1. 파워스캐너 프레임워크 구조

그림 1은 본 연구에서 제안하는 파워스캐너 프레임워크의 전체적인 시스템 구조를 도식화 한 그림이다. 사용자들은 안드로이드에서 동작하는 스마트폰에서 파워스캐너 어플리케이션을 사용하여 사용자가 사용하는 응용 프로그램에 대한 전력 소모 데이터를 수집한 후 서버로 전송 할 수 있다. 서버(웹 서버)에서는 다수의 사용자가 측정 후 전송한 데이터를 수집하며 수집된 데이터를 사용해서 각각의 응용 프로그램에 대한 평균 전력 소모량을 계산한 후 데이터베이스에 저장한다. 이렇게 해서 파워스캐너 프레임워크가 사용자들에게 제공할 전력 소모 데이터를 생성, 관리하는 것이다. 사용자는 측정에 사용한 파워스캐너 어플리케이션을 사용해서 서버로부터 전력 소모 데이터를 받을 수 있다. 또한 서버로부터 받은 데이터를 이용하여 사용자의 스마트폰에 설치되어 있는 응용 프로그램들에 대한 전력 소모 정보를 확인할 수 있다. 서버로부터 전송받은 데이터는 일회성으로 사용되는 것이 아니라 사용자의 필요에 따라 사용자의 스마트폰에 저장이 가능하며 이를 통해서 서버와의 통신 없이도 사용자는 언제든지 전력 소모 정보를 확인할 수 있다.

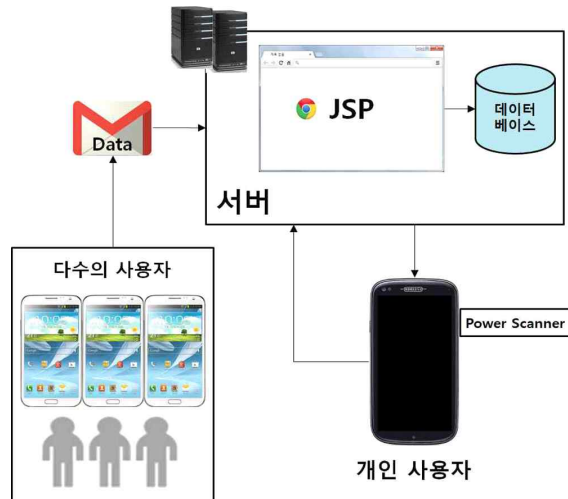


그림 1. 파워스캐너 프레임워크

2.2. 파워스캐너 응용 프로그램 구조

파워스캐너 응용 프로그램은 크게 다섯 가지 기능을 갖는다.

- 전력 소모 측정 기능
- 데이터 출력 기능
- 데이터 송신 기능
- 데이터 수신 기능
- 데이터 저장 기능

전력 소모 측정 기능은 하나의 응용프로그램에 대해서 다음과 같은 컴포넌트에 대해서 전력 소모량을 측정하며, 컴포넌트로는 LCD, CPU, Wifi, GPS, Audio가 있다. 측정 기능을 시작하는 당시 사용자의 스마트폰에서 실행되어지고 있는 응용 프로그램들을 uid로 식별하여 각각의 전력 소모량 정보를 측정하는 기능을 한다.

데이터 출력 기능은 사용자가 직접 측정한 결과 또는 서버로부터 받아온 데이터를 이용하여, 사용자의 스마트폰에 설치된 응용프로그램들의 평균 전력소모 데이터를 사용자에게 보여주는 기능을 한다.

데이터 송신 기능은 사용자가 측정한 전력 소모 데이터를 서버로 전송하는 기능을 한다.

데이터 수신 기능은 서버에게 평균 전력 소모 데이터를 요청하여 사용자의 스마트폰에 다운로드를 할 수 있도록 하는 기능이다.

데이터 저장 기능은 사용자가 직접 측정한 전력 소모 데이터 또는 서버로부터 다운로드 한 평균 전력 소모 데이터를 사용자의 스마트폰에 저장하는 기능을 한다.

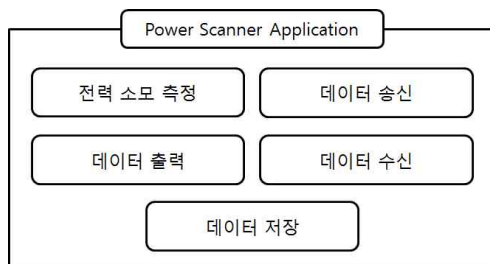


그림 2. 파워스캐너 응용 프로그램 구조

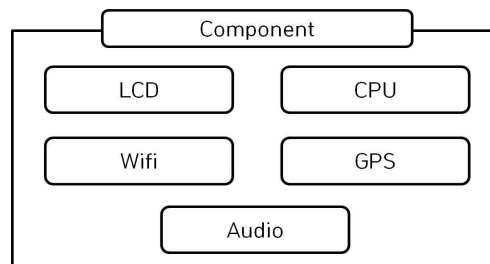


그림 3. 파워스캐너 응용 프로그램 구조

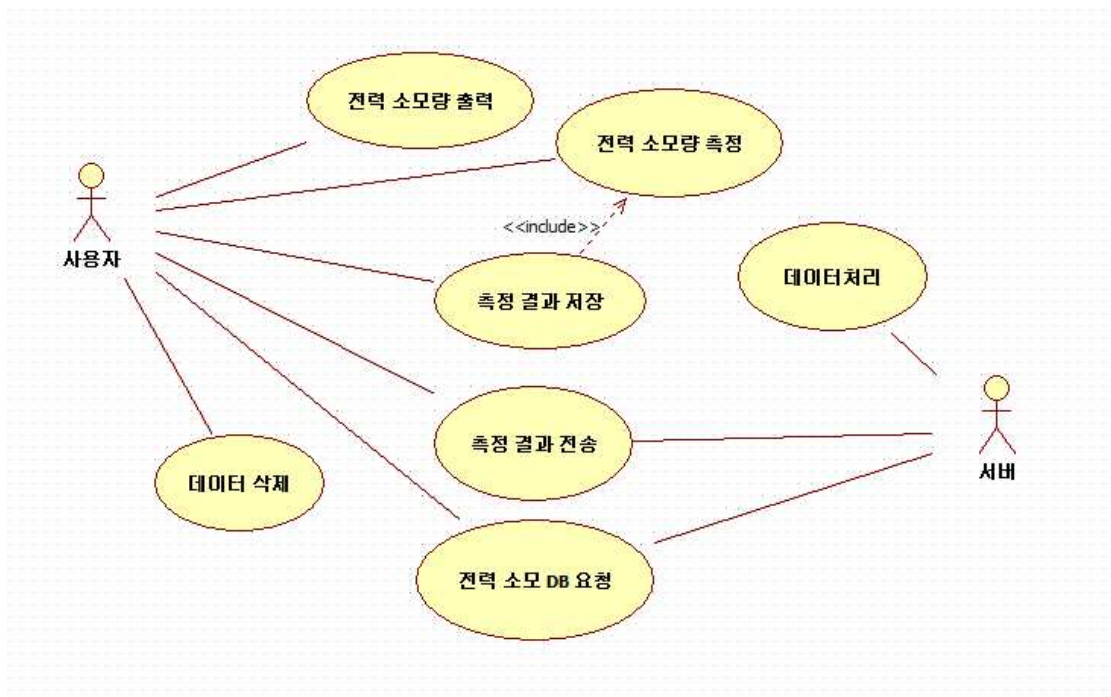
3. 파워스캐너 어플리케이션

이 장에서는 클라이언트가 사용할 파워스캐너 어플리케이션 개발을 위한 유즈케이스, 클래스, 시퀀스 다이어그램과 다이어그램에 대한 상세화에 대해서 설명한다.

- 유즈케이스 다이어그램 및 상세화
- 클래스 다이어그램 및 상세화
- 시퀀스 다이어그램 및 상세화

3.1. 유즈케이스 다이어그램

3.1.1. 유즈케이스 다이어그램



3.1.2. 유즈케이스 상세화

3.1.2.1. 전력 소모량 측정

전력 소모량 측정	
액터	사용자
설명	현재 사용중인 어플리케이션들의 전력사용량을 측정한다.
시작조건	메인 화면에서 측정 버튼을 선택한다.
종료조건	전력 소모량 측정에서 종료 버튼을 선택한다.
정상흐름	① 전력 소모량 측정에서 시작 버튼을 선택한다.
	② 측정하고자 하는 어플리케이션을 활성화 시킨다.
	③ 사용중인 어플리케이션 별 전력소모량을 측정한다.
	④ 측정을 종료하기 위해서 파워스캐너를 활성화 한다.
대체흐름	② 측정하고자 하는 어플리케이션이 활성화 되지 않은 경우 정상흐름 2번에서 다시 시작한다.

3.1.2.2. 측정 결과 저장

측정 결과 저장 (<include> 전력 소모량 측정)	
액터	사용자
설명	전력 소모량 측정 결과를 저장한다.
시작조건	전력 소모량 측정이 종료된 상태로, 전력 소모량 측정 화면이 보여지고 있다.
종료조건	전력 소모량 측정 결과가 저장된다.
정상흐름	① 전력 소모량 측정에서 저장 버튼을 선택한다.
	② 측정 결과를 저장한다.
대체흐름	① 전력 소모량 측정을 하지 않은 경우 전력 소모량 측정부터 시작한다.

3.1.2.3. 전력 소모량 출력

전력 소모량 출력	
액터	사용자
설명	어플리케이션 별 전력 소모량을 출력해준다.
시작조건	출력을 위한 데이터가 있고, 메인 화면이 보여지고 있다.
종료조건	선택된 데이터를 사용해서 화면에 출력된다.
정상흐름	① 전력 소모량 출력 버튼을 선택한다.
	② 데이터 목록에서 출력에 사용할 데이터를 선택한다.
	③ 확인 버튼을 선택한다.
	④ 화면에 출력한다.
대체흐름	② 다른 데이터를 선택한 경우, 다시 선택한다.
	② 데이터가 없는 경우, 전력 소모 데이터 요청기능을 사용해서 새로운 데이터를 받아 온 후 정상흐름 1번에서 다시 시작한다.
	④ 다른 데이터를 선택한 경우, 정상흐름 1번에서 다시 시작한다.

3.1.2.4. 전력 소모 데이터 요청

전력 소모 데이터 요청	
액터	사용자
설명	서버로 전력소모 데이터를 요청한다.
시작조건	데이터 관리 화면이 보여지고 있다.
종료조건	서버로부터 전력 소모 데이터 전송이 완료된다.
정상흐름	① 데이터 관리에서 데이터 요청 버튼을 선택한다.
	② 서버에 데이터를 요청한다.
	③ 서버로부터 전력소모 데이터를 받는다.
대체흐름	③ 데이터를 받는 도중에 전송이 끊기는 경우, 정상 흐름 1번에서 다시 시작한다.

3.1.2.5. 측정 결과 전송

측정 결과 전송	
액터	사용자
설명	전력소모량 측정 결과를 서버에 전송한다.
시작조건	전송할 측정 결과가 있어야 한다.
종료조건	측정 결과의 서버전송이 완료된다.
정상흐름	① 데이터 관리에서 서버 전송을 선택한다.
	② 전송할 측정결과 데이터를 선택한다.
	③ 전송 버튼을 선택해서 전송을 시작한다.
	④ 전력 소모량 측정 결과가 서버로 전송된다.
대체흐름	② 다른 파일을 선택한 경우, 다시 선택한다.
	③ 다른 파일을 전송한 경우, 정상 흐름 1번에서 다시 시작한다.

3.1.2.6. 측정 결과 전송

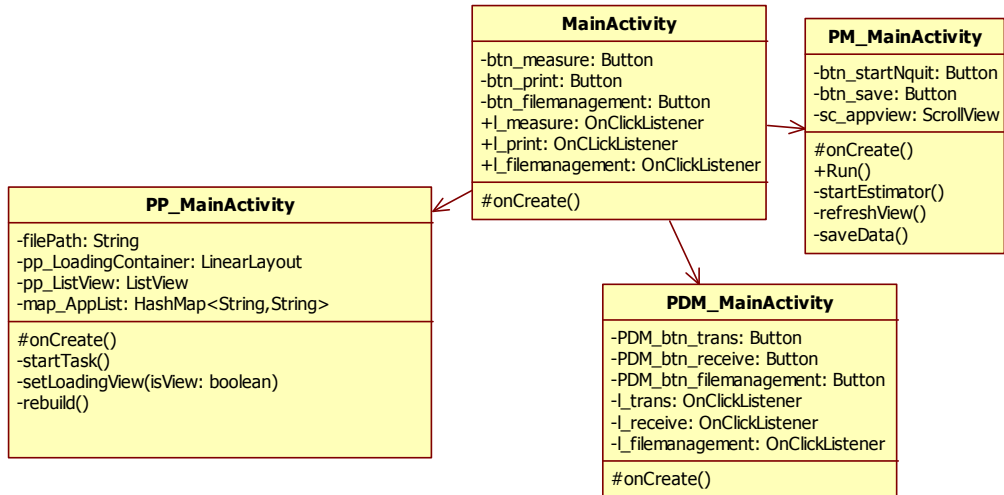
데이터 삭제	
액터	사용자
설명	전력 소모량 측정 파일, 전력 소모 데이터 파일을 삭제한다.
시작조건	저장된 파일이 있고, 데이터 관리 화면이 보여지고 있다.
종료조건	선택한 데이터가 삭제된다.
정상흐름	① 데이터 관리에서 데이터 삭제 버튼을 선택한다.
	② 파일 목록에서 삭제할 데이터를 선택한다.
	③ 삭제 버튼을 선택한다.
	④ 삭제 재확인 팝업 창에서 확인 버튼을 선택한다.
대체흐름	② 다른 파일을 선택한 경우, 다시 선택한다.
	④ 다른 파일을 선택한 경우, 취소 버튼을 선택 후 정상흐름 2번에서 다시 시작한다.

3.1.2.7. 측정 결과 전송

데이터 처리	
액터	서버
설명	사용자로부터 받은 데이터를 사용해서 어플리케이션 별로 평균 전력 소모량을 계산하여 DB에 저장한다.
시작조건	데이터를 처리 중이지 않고, 대기 상태이다.
종료조건	평균 전력 소모량 계산을 완료 후 대기 상태로 돌아간다.
정상흐름	① 처리할 데이터가 있는 지 확인한다.
	② 데이터가 있으면 계산을 시작한다.
	③ 처리가 완료된 데이터를 삭제한다.
대체흐름	① 수신되지 않은 경우 대기상태로 돌아간다.
	② 데이터가 없으면 정상 흐름 1번에서 다시 시작한다.

3.2. 클래스 다이어그램

3.2.1. 초기화면 클래스 다이어그램



3.2.1.1. MainActivity 상세화

MainActivity			
Class Diagram	<div> <div> MainActivity </div> <div> -btn_measure: Button -btn_print: Button -btn_filemanagement: Button +l_measure: OnClickListener +l_print: OnClickListener +l_filemanagement: OnClickListener #onCreate() </div> </div>		
Responsibility	Application에서 Main Activity이다. 측정, 출력, 관리 3개의 버튼을 갖고 있다. 3개의 버튼에 대한 OnClickListener가 있으며 각 리스너는 PM_MainActivity, PP_MainActivity, PDM_MainActivity를 시작한다.		
Attribute	Type	Name	Description
	Button	btn_measure	측정 기능을 위한 버튼
	Button	btn_print	출력 기능을 위한 버튼
	Button	btn_filemanagement	파일 관리 기능을 위한 버튼
Attribute	OnClickListener	l_measure	측정 버튼 클릭시 수행되는 onClick()이 포함된 객체

	OnClickListener	I_print	출력 버튼 클릭시 수행되는 onClick()이 포함된 객체
	OnClickListener	I_filemanagement	파일 관리 버튼 클릭시 수행되는 onClick()이 포함된 객체
Operation			
	Return Type	Method Name	Parameter Type
		onCreate	
	Description	화면을 구성하기 위해서 작성한 xml 레이아웃을 보여주기 위해서 연결한다. 또한 선언된 버튼 3개에 xml에 작성한 버튼을 연결한다. 마지막으로 버튼 선택 시 실행되는 OnClickListener를 각각의 버튼에 연결해준다.	

3.2.1.2. 상세화 - PM_MainActivity

PM_MainActivity			
Class Diagram	<div> <div>PM_MainActivity</div> <div> -btn_startNquit: Button -btn_save: Button -sc_appview: ScrollView </div> <div> #onCreate() +Run() -startEstimator() -refreshView() -saveData() </div> </div>		
Responsibility	<p>측정 버튼을 선택 시 시작되는 Activity이다. 이 클래스에서 전력소모량을 측정한다. 측정 시작, 종료를 위한 버튼과 저장을 위한 버튼을 갖고 있으며 측정 시 측정되는 값을 표현하기 위한 ScrollView를 가지고 있다. 측정을 위해서 스레드가 필요하기 때문에 Runnable 인터페이스를 구현했다. 또한 측정에 필요한 각 컴포넌트들에 대한 라이브러리를 사용하기 위해서 오픈소스 PowerTutor를 사용했다. PowerTutor에는 전력소모량 측정을 위한 컴포넌트들의 파워 모델, 배터리 정보, 시스템 정보 등을 위한 라이브러리가 있다.</p>		
Attribute	Type	Name	Description
	Button	btn_startNquit	측정 시작, 종료하기 위한 버튼으로 버튼의 텍스트가 시작시에는 종료로 종료시에는 시작으로 바뀐다.

	Button	btn_save	저장을 위한 버튼으로 측정 전에는 비활성상태이고, 측정을 완료한 후에 활성화상태로 바뀐다.
	ScrollView	sc_appview	측정 값을 실시간으로 보여주는 뷰로 측정되는 어플리케이션이 많아서 ListView로 표현하기 어렵기 때문에 ScrollView를 사용한다.

Operation	<table><tr><th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr><tr><td></td><td>onCreate</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">화면을 구성하기 위해서 작성한 xml 레이아웃을 보여주기 위해서 연결한다. 또한 선언된 버튼 2개에 xml에 작성한 버튼을 연결한다. 마지막으로 버튼 선택시 실행되는 OnClickListener를 각각의 버튼에 연결해준다.</td></tr></table>	Return Type	Method Name	Parameter Type	Parameter Name		onCreate			Description	화면을 구성하기 위해서 작성한 xml 레이아웃을 보여주기 위해서 연결한다. 또한 선언된 버튼 2개에 xml에 작성한 버튼을 연결한다. 마지막으로 버튼 선택시 실행되는 OnClickListener를 각각의 버튼에 연결해준다.		
	Return Type	Method Name	Parameter Type	Parameter Name									
		onCreate											
	Description	화면을 구성하기 위해서 작성한 xml 레이아웃을 보여주기 위해서 연결한다. 또한 선언된 버튼 2개에 xml에 작성한 버튼을 연결한다. 마지막으로 버튼 선택시 실행되는 OnClickListener를 각각의 버튼에 연결해준다.											
	<table><tr><th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr><tr><td></td><td>Run</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">Runnable 인터페이스의 Run 함수를 오버라이딩한 것으로 측정을 위해서 스레드가 동작할 때 수행할 측정 함수를 작성한다.</td></tr></table>	Return Type	Method Name	Parameter Type	Parameter Name		Run			Description	Runnable 인터페이스의 Run 함수를 오버라이딩한 것으로 측정을 위해서 스레드가 동작할 때 수행할 측정 함수를 작성한다.		
	Return Type	Method Name	Parameter Type	Parameter Name									
		Run											
	Description	Runnable 인터페이스의 Run 함수를 오버라이딩한 것으로 측정을 위해서 스레드가 동작할 때 수행할 측정 함수를 작성한다.											
	<table><tr><th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr><tr><td></td><td>startEstimator</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">측정을 위한 함수이다. 파워 튜터의 라이브러리를 사용해서 각 컴포넌트의 사용량을 측정, 전력 소모량을 계산한다. 이 함수에서 DataBuffer 객체를 사용해서 어플리케이션 별 전력소모량을 관리한다.</td></tr></table>	Return Type	Method Name	Parameter Type	Parameter Name		startEstimator			Description	측정을 위한 함수이다. 파워 튜터의 라이브러리를 사용해서 각 컴포넌트의 사용량을 측정, 전력 소모량을 계산한다. 이 함수에서 DataBuffer 객체를 사용해서 어플리케이션 별 전력소모량을 관리한다.		
	Return Type	Method Name	Parameter Type	Parameter Name									
	startEstimator												
Description	측정을 위한 함수이다. 파워 튜터의 라이브러리를 사용해서 각 컴포넌트의 사용량을 측정, 전력 소모량을 계산한다. 이 함수에서 DataBuffer 객체를 사용해서 어플리케이션 별 전력소모량을 관리한다.												
<table><tr><th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr><tr><td></td><td>refreshView</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">이 함수에서는 실시간으로 측정되는 전력소모량을 ScrollView에 반영하기 위해서 사용되는 함수이다. 스레드를 사용하기 때문에 Handler와 post 함수를 사용해서 측정하는 모든 어플리케이션의 전력 소모량을 스레드를 사용해서 계속해서 업데이트 한다.</td></tr></table>	Return Type	Method Name	Parameter Type	Parameter Name		refreshView			Description	이 함수에서는 실시간으로 측정되는 전력소모량을 ScrollView에 반영하기 위해서 사용되는 함수이다. 스레드를 사용하기 때문에 Handler와 post 함수를 사용해서 측정하는 모든 어플리케이션의 전력 소모량을 스레드를 사용해서 계속해서 업데이트 한다.			
Return Type	Method Name	Parameter Type	Parameter Name										
	refreshView												
Description	이 함수에서는 실시간으로 측정되는 전력소모량을 ScrollView에 반영하기 위해서 사용되는 함수이다. 스레드를 사용하기 때문에 Handler와 post 함수를 사용해서 측정하는 모든 어플리케이션의 전력 소모량을 스레드를 사용해서 계속해서 업데이트 한다.												
<table><tr><th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr><tr><td></td><td>saveData</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">이 함수는 저장 버튼을 선택 시에 수행되는 기능으로 DataBuffer 객체에 저장된 전력 소모량 데이터를 파일로 저장한다.</td></tr></table>	Return Type	Method Name	Parameter Type	Parameter Name		saveData			Description	이 함수는 저장 버튼을 선택 시에 수행되는 기능으로 DataBuffer 객체에 저장된 전력 소모량 데이터를 파일로 저장한다.			
Return Type	Method Name	Parameter Type	Parameter Name										
	saveData												
Description	이 함수는 저장 버튼을 선택 시에 수행되는 기능으로 DataBuffer 객체에 저장된 전력 소모량 데이터를 파일로 저장한다.												

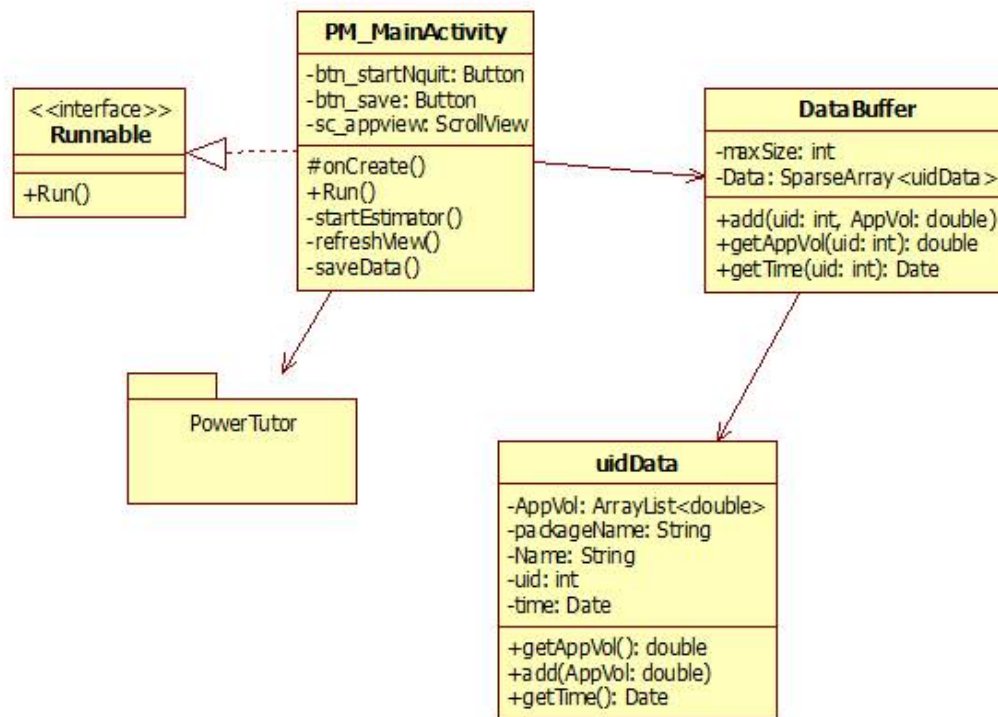
3.2.1.3. 상세화 - PDM_MainActivity

PDM_MainActivity	
Class Diagram	<div style="border: 1px solid black; padding: 5px; margin: 10px auto; width: fit-content;"> <p style="text-align: center;">PDM_MainActivity</p> <p>-PDM_btn_trans: Button -PDM_btn_receive: Button -PDM_btn_filemanagement: Button -l_trans: OnClickListener -l_receive: OnClickListener -l_filemanagement: OnClickListener</p> <p>#onCreate()</p> </div>
Responsibility	<p>데이터 관리 기능 액티비티. 데이터 관리는 측정한 데이터 전송, 데이터 삭제, 데이터 요청 기능이 있다. 이 클래스에는 각 기능에 대한 버튼이 있으며 각 버튼을 선택하면 기능에 맞는 액티비티가 시작된다.</p> <p>attribute, operation은 다른 MainActivity와 유사하다.</p>

3.2.1.4. 상세화 - PP_MainActivity

PP_MainActivity			
Class Diagram	<div><div>PP_MainActivity</div><div><div>-filePath: String</div><div>-pp>LoadingContainer: LinearLayout</div><div>-pp>ListView: ListView</div><div>-map_AppList: HashMap<String,String></div></div><div><div>#onCreate()</div><div>+onResume()</div><div>-startTask()</div><div>-setLoadingView(isView: boolean)</div><div>-rebuild()</div></div></div>		
Responsibility	<p>출력 버튼 선택 시 시작되는 Activity이다. FileListActivity를 불러서 출력에 사용될 파일을 선택한 후 선택한 파일의 값을 사용해서 스마트폰에 설치된 어플리케이션의 전력소모량을 출력해준다.</p> <p>출력을 위해 값을 계산하는 중에는 로딩화면을 출력해주고, 계산이 끝나면 ListView를 사용해서 어플리케이션의 전력소모량을 출력한다.</p>		
Attribute	Type	Name	Description
	String	filePath	선택한 파일의 경로가 저장된다.
	LinearLayout	pp>LoadingContainer	출력을 위해서 값을 계산하는 중에 보여줄 로딩화면.

3.2.2. 전력소모 측정 및 저장기능 클래스 다이어그램



3.2.2.1. 상세화 - DataBuffer

DataBuffer			
Class Diagram	<div> <div>DataBuffer</div> <div> -maxSize: int -Data: SparseArray<uidData> </div> <div> +add(uid: int, AppVol: double) +getAppVol(uid: int): double +getTime(uid: int): Date </div> </div>		
	이 클래스는 어플리케이션 전력소모량 측정 결과를 관리하기 위한 클래스이다. 이 클래스에서는 측정 중인 어플리케이션의 개수만큼 어플리케이션에 대한 패키지명, 어플리케이션명, 전력 소모량, 측정 시간을 위한 uidData 객체를 갖는다.		
Attribute	Type	Name	Description
	int	maxSize	측정하는 어플리케이션의 최대 개수를 정하기 위한 변수로 스마트폰에 설치된 어플리케이션의 개수가 저장된다.
	SparseArray<uidData>	Data	실제 측정된 결과가 저장된다.

Operation	Return Type	Method Name	Parameter Type	Parameter Name
		add	int double	uid AppVol
	Description	측정된 값을 저장하기 위한 함수로 synchronized 로 선언하여 쓰레드 간에 동기를 유지할 수 있다.		
	Return Type	Method Name	Parameter Type	Parameter Name
	Date	getTime	int	uid
	Description	Uid에 해당하는 어플리케이션의 측정 시간을 반환하는 함수로 getAppVol과 함께 측정시 실시간으로 뷰를 업데이트하는데 사용된다.		
	Return Type	Method Name	Parameter Type	Parameter Name
	double	getAppVol	int	uid
	Description	Uid에 해당하는 어플리케이션의 전력소모량을 반환하는 함수로 add와 마찬가지로 synchronized로 선언되어서 쓰레드 간에 동기를 유지할 수 있다.		

3.2.2.2. 상세화 - uidData

uidData			
Class Diagram	<div> uidData -AppVol: ArrayList<double> -packageName: String -Name: String -uid: int -time: Date +getAppVol(): double +add(AppVol: double) +getTime(): Date </div>		
Responsibility	이 클래스는 어플리케이션 전력소모량 측정 결과 값이 저장되는 클래스이다. 패키지 명, 어플리케이션 명, 측정 시간, 측정값이 저장된다.		
Attribute	Type	Name	Description
	int	uid	어플리케이션의 id.
	String	packageName	패키지 명.
	String	Name	어플리케이션 명.
	ArrayList<double>	AppVol	전력 소모량이 저장된다.
	Date	time	측정 시간을 저장한다.

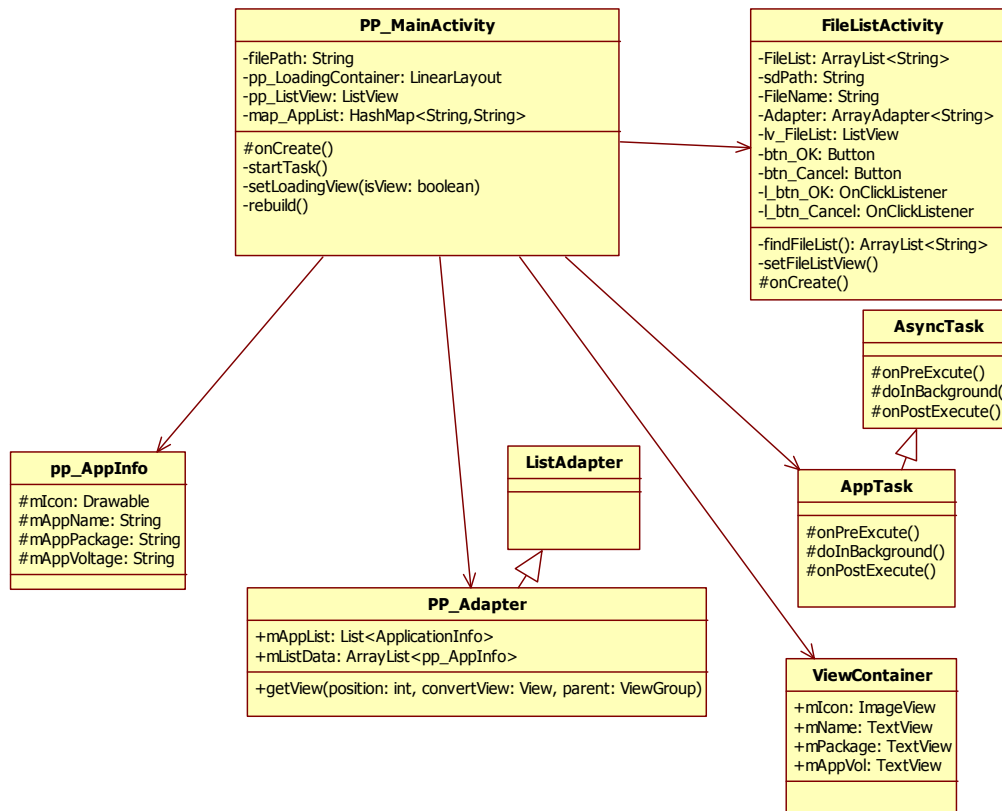
Operation

Return Type	Method Name	Parameter Type	Parameter Name
	add	double	AppVol
Description	DataBuffer의 add 함수가 호출하는 함수로 측정된 값을 저장한다.		

Return Type	Method Name	Parameter Type	Parameter Name
double	getAppVol		
Description	DataBuffer의 getAppVol함수 내부에서 호출하는 함수로 전력 소모량을 반환한다.		

Return Type	Method Name	Parameter Type	Parameter Name
Date	getTime		
Description	DataBuffer의 getTime함수 내부에서 호출하는 함수로 측정 시간을 반환한다.		

3.2.3. 전력소모 출력 클래스 다이어그램



3.2.3.1. 상세화 - FileListActivity

FileListActivity	
Class Diagram	<div> <div>FileListActivity</div> <div> -FileList: ArrayList<String> -sdPath: String -FileName: String -Adapter: ArrayAdapter<String> -lv_FileList: ListView -btn_OK: Button -btn_Cancel: Button -l_btn_OK: OnClickListener -l_btn_Cancel: OnClickListener -findFileList(): ArrayList<String> -setFileListView() #onCreate() </div> </div>
Responsibility	출력, 전송, 삭제 기능을 사용할 때 파일을 선택하기 위해서 파일 목록을 보여주는 액티비티이다. 목록에서 파일을 선택 후 확인 버튼을 선택한 경우 선택한 파일 명을 FileListActivity를 호출한 액티비티로 넘겨준다.

Attribute	Type	Name	Description	
	String	sdPath	파일이 저장된 경로.	
	String	FileName	사용자가 선택한 파일명.	
	ListView	lv_FileList	파일 목록을 보여줄 ListView.	
	ArrayAdapter<String>	Adapter	ListView에 사용될 Adapter.	
	ArrayList<String>	FileList	sdPath에 있는 파일 목록.	
Operation				
	Return Type	Method Name	Parameter Type	Parameter Name
		onCreate		
	Description	화면을 구성하기 위해서 작성한 xml 레이아웃을 보여주기 위해서 연결한다.		
	Return Type	Method Name	Parameter Type	Parameter Name
	ArrayList<String>	findFileList		
	Description	파워스캐너에 사용되는 파일이 저장된 경로(sdPath)에 있는 파일의 이름을 가져온다.		
	Return Type	Method Name	Parameter Type	Parameter Name
		setFileListView		
	Description	findFileList 함수에서 작성된 파일 목록을 ListView로 보여주기 위해서 adapter를 사용해서 ListView와 연결한다.		

3.2.3.2. 상세화 - AppTask

AppTask													
Class Diagram	<div><div>AppTask</div><div>#onPreExcute() #doInBackground() #onPostExecute()</div></div>												
Responsibility	<p>안드로이드의 AsyncTask 클래스를 상속받아서 구현한 클래스이다. AsyncTask를 상속받아서 구현하면 onPreExcute, doInBackground, onPostExecute 3개의 함수를 오버라이딩하는데 이를 통해서 UI의 처리와 다른 작업을 하나의 클래스에서 작업할 수 있게 된다. 이 클래스를 사용할 때는 AsyncTask.Execute()를 호출하면 된다.</p> <p>AppTask를 사용해서 로딩 UI와 ListView 출력 그리고 출력에 사용할 View를 생성하는 작업을 처리할 수 있다. PP_MainActivity의 함수를 사용하기 때문에 내부 클래스로 구현한다.</p>												
Operation	<table><tr><th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr><tr><td></td><td>onPreExcute</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">LoadingView를 화면에 출력하게 한. 이 과정에서 ListView를 disable 시킨다.</td></tr></table>	Return Type	Method Name	Parameter Type	Parameter Name		onPreExcute			Description	LoadingView를 화면에 출력하게 한. 이 과정에서 ListView를 disable 시킨다.		
	Return Type	Method Name	Parameter Type	Parameter Name									
		onPreExcute											
	Description	LoadingView를 화면에 출력하게 한. 이 과정에서 ListView를 disable 시킨다.											
	<table><tr><th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr><tr><td></td><td>doInBackground</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">PP_MainActivity의 rebuild 함수를 호출한다.</td></tr></table>	Return Type	Method Name	Parameter Type	Parameter Name		doInBackground			Description	PP_MainActivity의 rebuild 함수를 호출한다.		
	Return Type	Method Name	Parameter Type	Parameter Name									
		doInBackground											
	Description	PP_MainActivity의 rebuild 함수를 호출한다.											
	<table><tr><th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr><tr><td></td><td>onPostExecute</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">doInBackground 함수가 완료되면 LoadingView를 disable시킨 후 ListView를 화면에 출력하게 한다.</td></tr></table>	Return Type	Method Name	Parameter Type	Parameter Name		onPostExecute			Description	doInBackground 함수가 완료되면 LoadingView를 disable시킨 후 ListView를 화면에 출력하게 한다.		
Return Type	Method Name	Parameter Type	Parameter Name										
	onPostExecute												
Description	doInBackground 함수가 완료되면 LoadingView를 disable시킨 후 ListView를 화면에 출력하게 한다.												

3.2.3.3. 상세화 - ViewContainer

ViewContainer			
Class Diagram	<div> <div>ViewContainer</div> <div> +mIcon: ImageView +mName: TextView +mPackage: TextView +mAppVol: TextView </div> </div>		
Responsibility	<p>출력에는 ListView를 사용하는데 ListView의 각 item은 ViewContainer로 구성되어 있다. 이 클래스에는 출력에 필요한 아이콘, 어플리케이션 명, 패키지 명, 평균 전력 소모량이 저장된다. 클래스의 각 속성은 모두 View로 이루어져있다.</p>		
Attribute	Type	Name	Description
	ImageView	mIcon	어플리케이션의 아이콘
	TextView	mAppName	어플리케이션 이름
	TextView	mAppPackage	어플리케이션 패키지 명
	TextView	mAppVoltage	평균 전력 소모량
Operation			

3.2.3.4. 상세화 - pp_Adapter

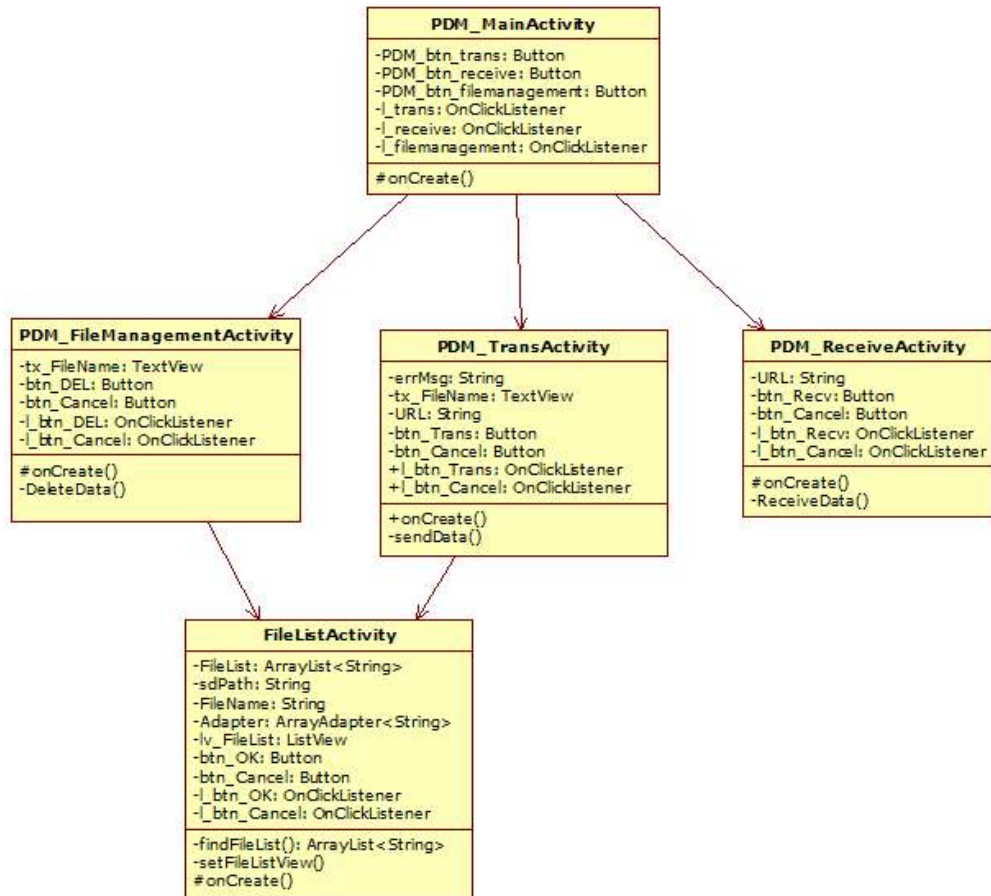
PP_Adapter	
Class Diagram	<div> <div>PP_Adapter</div> <div> +mAppList: List<ApplicationInfo> +mListData: ArrayList<pp_AppInfo> +getView(position: int, convertView: View, parent: ViewGroup) </div> </div>
Responsibility	<p>출력에 사용되는 listView를 위한 Adapter로 ListAdapter를 상속받아서 구현한다. ArrayList를 사용해서 사용자가 선택한 파일(전력 소모량이 저장되어 있다.)에 저장된 데이터를 가져온다. 그리고 getView 함수에서 ArrayList를 이용해서 출력을 위한 ViewContainer를 생성한다.</p>

Attribute	Type	Name	Description	
	List<ApplicationInfo>	mAppList	스마트폰에 설치된 어플리케이션의 목록	
	ArrayList<pp_AppInfo>	mListData	파일에서 읽어온 어플리케이션 별 전력 소모량	
Operation				
	Return Type	Method Name	Parameter Type	Parameter Name
	View	getView		
	Description	mAppList와 mListData를 비교해서 파일에 저장된 값중 스마트폰에 설치된 어플리케이션에 대해서 ViewContainer 객체를 생성한다.		

3.2.3.5. 상세화 - pp_Appinfo

pp_AppInfo			
Class Diagram	<div> pp_AppInfo #mIcon: Drawable #mAppName: String #mAppPackage: String #mAppVoltage: String </div>		
Responsibility	출력에 사용할 파일을 선택한 후, 출력을 위한 뷰를 생성하는 과정에서 파일에서 읽어온 어플리케이션 명, 패키지 명, 전력소모량과 아이콘을 갖는 클래스이다.		
Attribute	Type	Name	Description
	Drawable	mIcon	어플리케이션의 아이콘
	String	mAppName	어플리케이션 이름
	String	mAppPackage	어플리케이션 패키지 명
Operation	String	mAppVoltage	평균 전력 소모량

3.2.4. 데이터 관리기능 클래스 다이어그램



3.2.4.1. 상세화 - PDM_ReceiveActivity

PDM_ReceiveActivity	
Class Diagram	<div> <div>PDM_ReceiveActivity</div> <div> -URL: String -btn_Recv: Button -btn_Cancel: Button -l_btn_Recv: OnClickListener -l_btn_Cancel: OnClickListener #onCreate() -ReceiveData() </div> </div>
Responsibility	서버에 전력 소모 데이터를 요청하는 클래스이다.

Attribute	Type	Name	Description
	String	URL	웹 서버의 주소가 저장되어 있다.
	Button	btn_Recv	요청 버튼으로 선택한 파일이 전송된다.
	Button	btn_Cancel	이전 화면으로 돌아간다.
	OnClickListener	l_btn_Recv	요청 버튼에 연결된 OnClickListener
	OnClickListener	l_btn_Cancel	취소 버튼에 연결된 OnClickListener
Operation			
	Return Type	Method Name	Parameter Type
		onCreate	
	Description	화면을 구성하기 위해서 작성한 xml 레이아웃을 보여주기 위해서 연결한다.	
	Return Type	Method Name	Parameter Type
		ReceiveData	
	Description	URL의 주소를 사용해서 웹 서버에 전력 소모 데이터를 요청한다.	

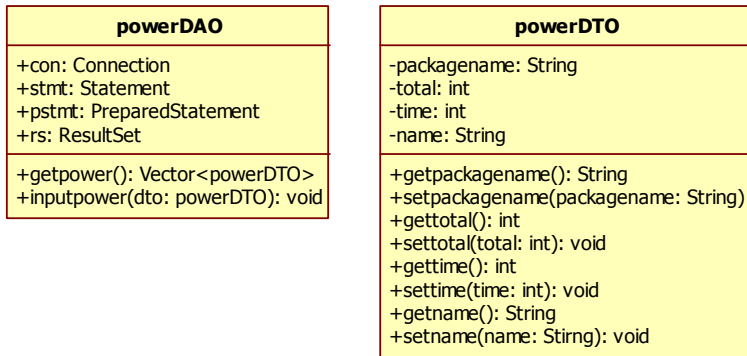
3.2.4.2. 상세화 - PDM_TransActivity

PDM_TransActivity	
Class Diagram	<div> PDM_TransActivity -errMsg: String -tx_FileName: TextView -URL: String -btn_Trans: Button -btn_Cancel: Button +l_btn_Trans: OnClickListener +l_btn_Cancel: OnClickListener +onCreate() -sendData() </div>
Responsibility	측정한 결과를 서버로 전송하기 위한 기능이다. 삭제 기능처럼 FileListActivity를 사용해서 파일 목록을 사용자에게 보여준다. 파일 목록

3.2.4.3. 상세화 - PDM_FileManagement

PDM_FileManagement															
Class Diagram	<div><div>PDM_FileManagementActivity</div><div><div>-tx_FileName: TextView</div><div>-btn_DEL: Button</div><div>-btn_Cancel: Button</div><div>-l_btn_DEL: OnClickListener</div><div>-l_btn_Cancel: OnClickListener</div></div><div><div>#onCreate()</div><div>-DeleteData()</div></div></div>														
	현재 스마트폰에 저장된 측정 결과 및 서버로부터 받은 데이터를 삭제하기 위한 기능이다. 출력 기능처럼 FileListActivity를 사용해서 파일 목록을 사용자에게 보여준다. 파일 목록에서 삭제할 파일을 선택하면 FileListActivity가 종료되면서 해당 Activity로 선택한 파일 명을 전달한다.														
Attribute	Type	Name	Description												
	TextView	tx_FileName	FileListActivity에서 전달 받은 파일 명을 출력												
	Button	btn_DEL	삭제 버튼으로 선택한 파일을 삭제된다.												
	Button	btn_Cancel	이전 화면으로 돌아간다.												
	OnClickListener	l_btn_DEL	삭제 버튼에 연결된 OnClickListener												
	OnClickListener	l_btn_Cancel	취소 버튼에 연결된 OnClickListener												
Operation	<table><tr><th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr><tr><td></td><td>onCreate</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">화면을 구성하기 위해서 작성한 xml 레이아웃을 보여주기 위해서 연결한다.</td></tr></table>			Return Type	Method Name	Parameter Type	Parameter Name		onCreate			Description	화면을 구성하기 위해서 작성한 xml 레이아웃을 보여주기 위해서 연결한다.		
	Return Type	Method Name	Parameter Type	Parameter Name											
		onCreate													
	Description	화면을 구성하기 위해서 작성한 xml 레이아웃을 보여주기 위해서 연결한다.													
	<table><tr><th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr><tr><td></td><td>DeleteData</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">사용자가 선택한 파일을 삭제한다.</td></tr></table>			Return Type	Method Name	Parameter Type	Parameter Name		DeleteData			Description	사용자가 선택한 파일을 삭제한다.		
	Return Type	Method Name	Parameter Type	Parameter Name											
		DeleteData													
	Description	사용자가 선택한 파일을 삭제한다.													

3.2.5. 웹서버 클래스 다이어그램



3.2.5.1. powerDAO

powerDAO																
Class Diagram	<table><tr><th colspan="4">powerDAO</th></tr><tr><td colspan="4">+con: Connection +stmt: Statement +pstmt: PreparedStatement +rs: ResultSet</td></tr><tr><td colspan="4">+getpower(): Vector<powerDTO> +inputpower(dto: powerDTO): void</td></tr></table>				powerDAO				+con: Connection +stmt: Statement +pstmt: PreparedStatement +rs: ResultSet				+getpower(): Vector<powerDTO> +inputpower(dto: powerDTO): void			
	powerDAO															
+con: Connection +stmt: Statement +pstmt: PreparedStatement +rs: ResultSet																
+getpower(): Vector<powerDTO> +inputpower(dto: powerDTO): void																
Responsibility	웹 서버에서 데이터베이스에 접근하기 위해서 사용하는 클래스이다. 이 클래스에는 연결을 맺고, 끊기 위한 함수가 Private 으로 구현되어 있으며 데이터베이스에 데이터를 삽입하고 가져오기 위한 함수가 구현되어 있다.															
Attribute	Type	Name	Description													
	Connection	con	데이터베이스와 연결을 맺기 위한 객체													
	Statement	stmt	쿼리문을 사용하기 위한 객체													
	ResultSet	rs	쿼리 실행 결과를 가져오는 객체													
Operation	Return Type	Method Name	Parameter Type	Parameter Name												
	Vector<powerDTO>	getpower														
	Description	데이터베이스에서 값을 가져오기 위한 함수로 데이터베이스에 저장된 모든 어플리케이션에 대한 데이터를 가져오는 데 한 번에 하나의 어플리케이션에 대한 데이터를 가져와서 powerDTO객체를 생성한 후 리스트에 삽입한다. 모든 데이터를 가져오면 리스트를 반환한다.														

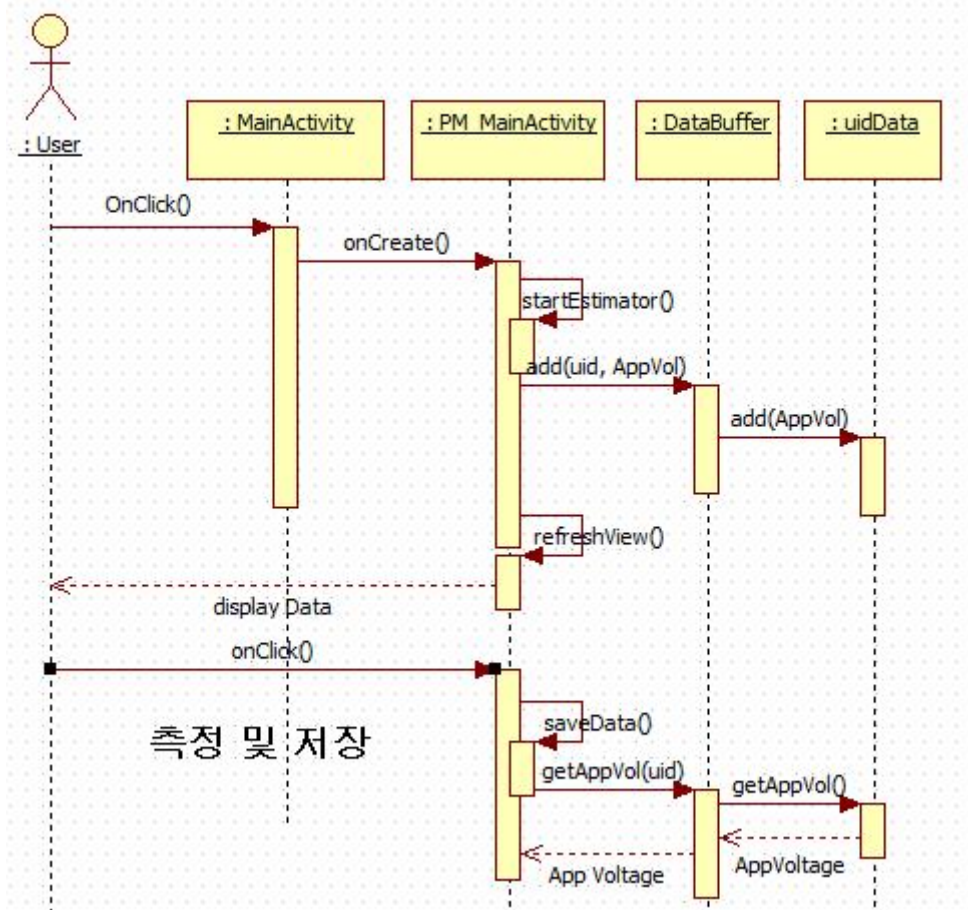
Return Type	Method Name	Parameter Type	Parameter Name
	inputpower	powerDTO	dto
Description	사용자가 전송한 데이터를 웹 서버에서 파싱, 객체 리스트를 생성 후 데이터베이스에 삽입할 때 사용하는 함수로 데이터베이스와 연결한 후 데이터베이스에 리스트의 데이터를 삽입한 후 연결을 종료한다.		

3.2.5.1. powerDTO

powerDTO			
Class Diagram	<div> <div>powerDTO</div> <div> -packagename: String -total: int -time: int -name: String +getpackagename(): String +setpackagename(packagename: String) +gettotal(): int +settotal(total: int): void +gettime(): int +settime(time: int): void +getname(): String +setname(name: String): void </div> </div>		
Responsibility	웹 서버에서 전송받은 데이터를 파싱하는데 사용하는 객체로 패키지명, 측정 시간, 전력 소모량, 어플리케이션 명이 들어간다.		
Attribute	Type	Name	Description
	String	packagename	패키지명
	int	total	전력 소모량
	int	time	측정시간
	String	name	어플리케이션 명

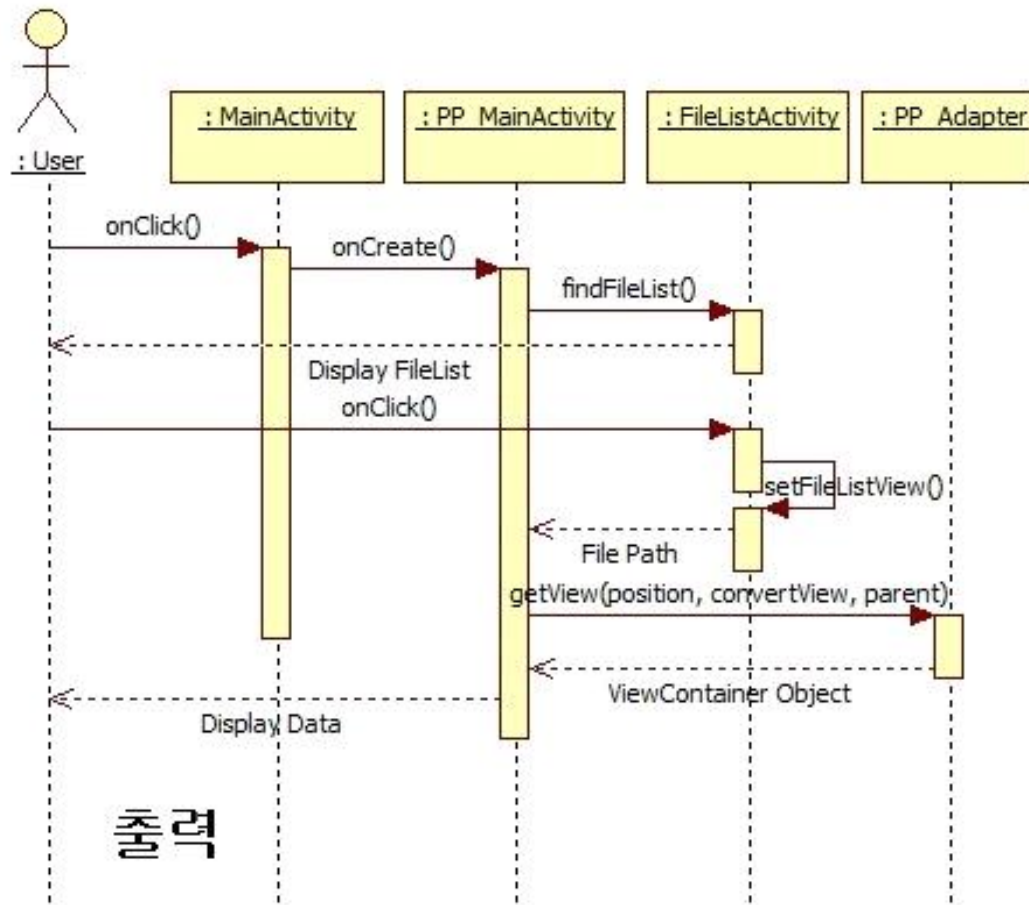
3.3. 시퀀스 다이어그램

3.3.1. 측정 및 저장 시퀀스 다이어그램



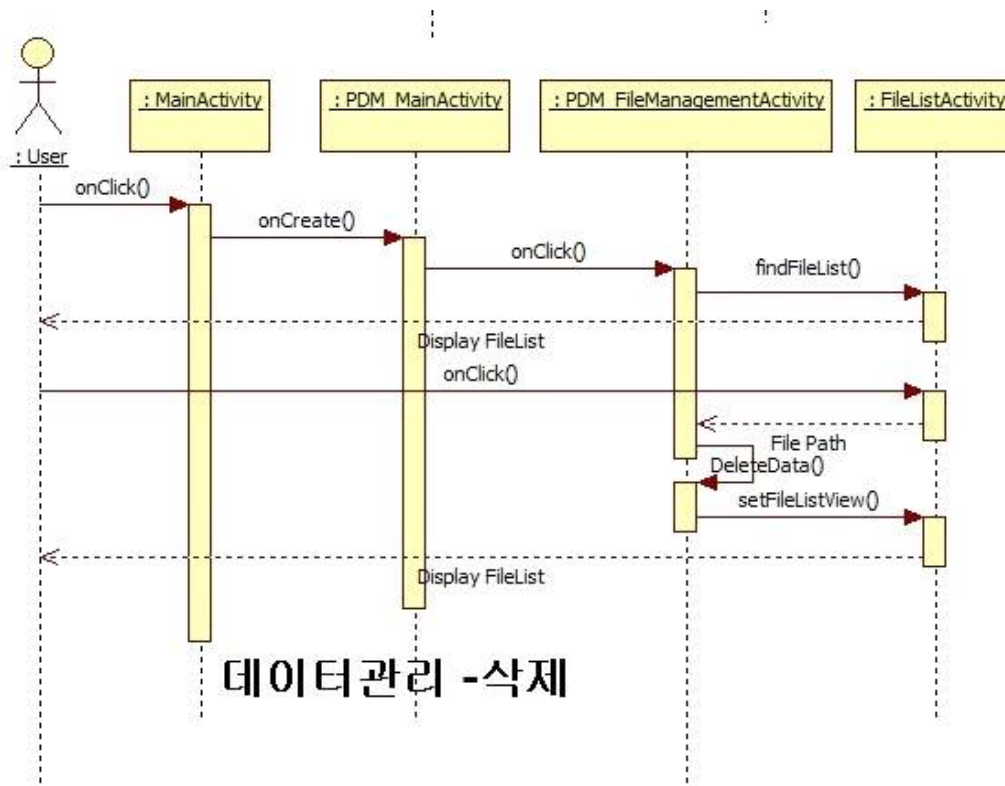
USE-CASE	사용자 - 전력 소모량 측정, 저장
설 명	<p>① User는 MainActivity의 측정버튼을 클릭하여 PM_MainActivity의 onCreate를 호출하여 xml레이아웃을 출력</p> <p>② startEstimator를 실행하여 각 컴포넌트의 사용량을 측정 계산</p> <p>③ DataBuffer는 전력소모량 측정결과를 관리하며 측정 중인 어플리케이션 개수만큼의uidData객체를 가짐.</p> <p>④ add() operation을 통하여 uidData객체에 측정값을 저장</p> <p>⑤ refreshView()를 통해 실시간으로 측정되는 값을 업데이트하여 사용자에게 보여줌</p> <p>⑥ 측정 후에 저장버튼이 활성화 되고 User가 저장버튼클릭을 통해 저장가능</p> <p>⑦ saveData는 저장버튼을 클릭시 호출되며 DataBuffer에 저장된 전력소모량 데이터를 얻어와 파일로 저장</p>

3.3.2. 출력 시퀀스 다이어그램



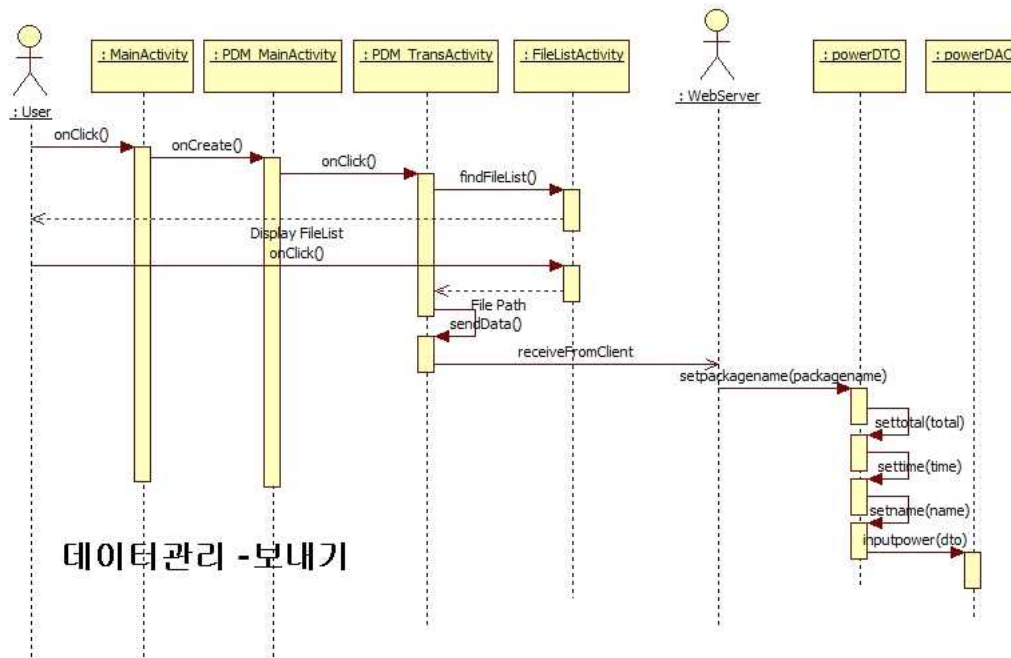
USE-CASE	사용자 - 전력 소모량 출력
설 명	<p>① User는 MainActivity의 출력버튼을 클릭하여 PP_MainActivity의 onCreate를 호출하여 xml레이아웃을 출력</p> <p>② PP_MainActivity의 onCreate는 FileListActivity의 onCreate를 intent로 호출</p> <p>③ findFileList를 사용하여 지정된 sd memory에 존재하는 파일의 이름을 가져와서 User에게 출력</p> <p>④ User가 파일을 선택하면 setFileListView를 사용하여 ListView를 구성.</p> <p>파일의 경로를 반환받아PP_Adapter의 getView를 통해 스마트폰에 설치된 어플리케이션에 대해 viewContainer 형태로 만들어 반환</p> <p>⑤ 사용자에게 전력사용량 출력</p>

3.3.3. 데이터삭제 다이어그램



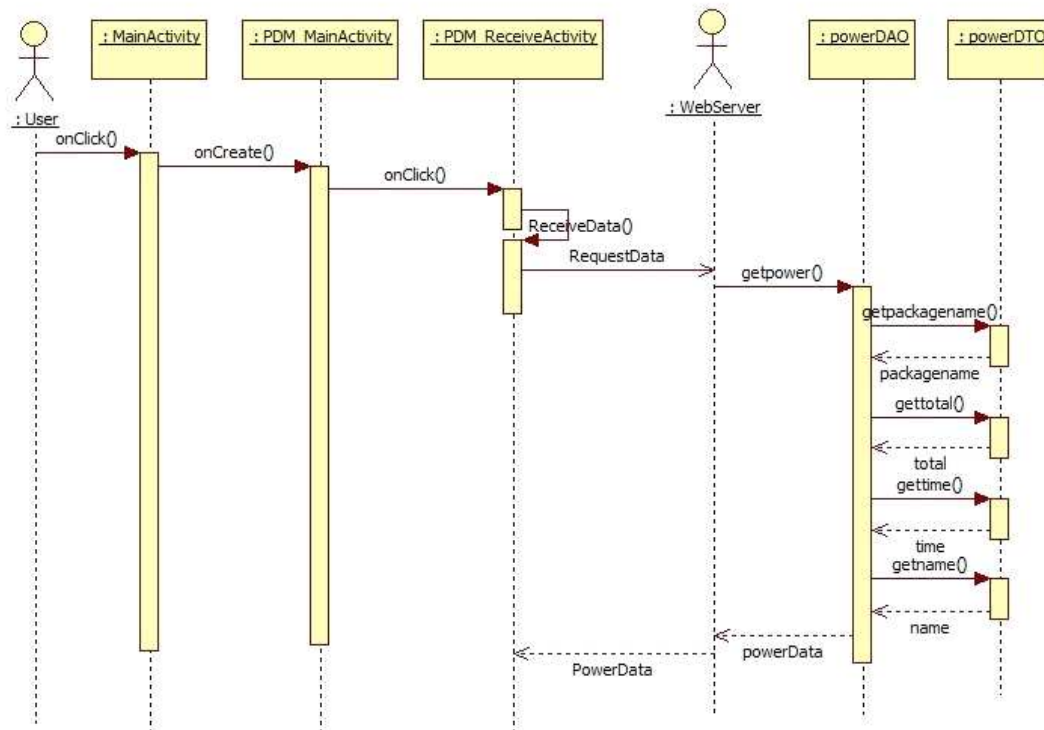
USE-CASE	사용자 - 데이터 삭제
설 명	<p>① User는 MainActivity의 데이터관리버튼을 클릭하여 PDM_MainActivity의 onCreate를 호출하여 xml레이아웃을 출력</p> <p>② PDM_MainActivity는 3개의 버튼을 가지고 있으며 삭제버튼을 클릭하면 PDM_FileManagementActivity의 onCreate가 호출되면서 FileListActivity의 onCreate를 호출</p> <p>③ FileListActivity의 findFileList함수를 이용해 저장된 파일리스트를 User에게 출력</p> <p>④User는 삭제할 파일을 선택하고 삭제버튼을 누르면 해당파일이 존재하는 파일의 경로를 반환</p> <p>⑤ DeleteData를 사용하여 파일을 삭제하고 파일목록을 업데이트 하여 사용자에게 출력한다</p>

3.3.4. 데이터 보내기 다이어그램



USE-CASE	사용자 - 측정결과 전송, 서버 - 데이터 처리, 측정 결과 전송
설 명	<p>① User가 데이터관리 버튼을 클릭하여 PDM_MainActivity의 onCreate를 호출</p> <p>② 보내기 버튼을 클릭하여 findFileList를 호출하여 저장된 파일리스트를 출력</p> <p>③ User는 보낼파일을 선택하면 FileListActivity는 선택한 파일의 경로를 반환하고 sendData를 실행</p> <p>④ 웹서버는 receiveFromClient를 이용하여 데이터를 받고 서버는 powerDTO클래스에 전송받은 값을 설정</p> <p>⑤ powerDTO는 설정된 값을 powerDAO의 inputpower를 통해 보내고 powerDAO는 DB에 저장한다</p>

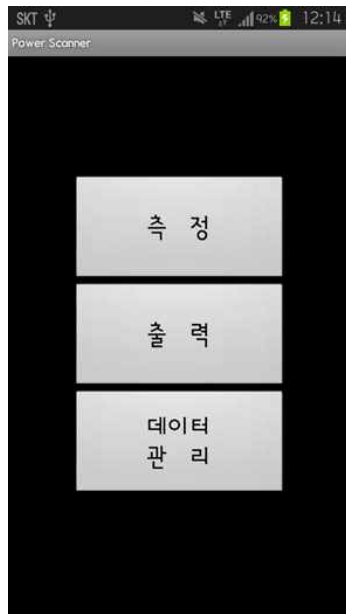
3.3.5. 데이터 받기 다이어그램



데이터관리 - 받기

USE-CASE	사용자 - 전력소모DB요청 서버 - 전력소모DB요청, 데이터 처리
설 명	① User가 데이터관리버튼을 클릭하여 PDM_MainActivity의 onCreate를 호출 ② PDM_MainActivity의 수신버튼을 클릭하여 ReceiveData 실행 ③ 웹서버로 데이터를 요청하며 웹서버는 getPower를 통하여 powerDAO에 데이터를 ④ 요청하고 powerDTO에 접근하여 값을 읽어온 후 반환 ⑤ 웹서버는 데이터를 Client로 전달

3.4 응용프로그램 실행 화면



<초기 화면>



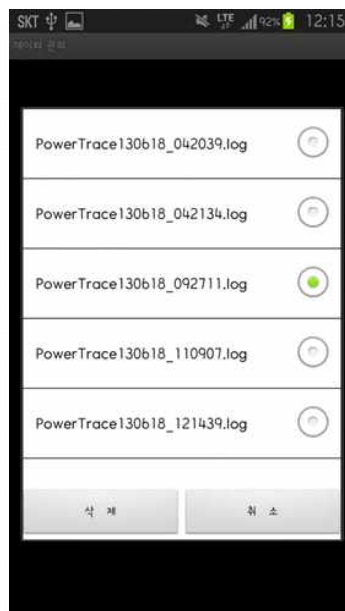
<측정 메인 화면>



<측정 시작 화면>



<출력 화면>



<파일 선택 화면>



<데이터관리>

4. 웹 서버 및 데이터베이스

본 논문에서는 사용자가 측정한 데이터를 수집하고 사용자에게 전력 소모 정보를 제공하기 위해서 웹 서버를 사용했다. 클라이언트 어플리케이션에서 데이터 송·수신하기 위한 방법이 여러 가지 있지만 HTTP를 이용해서 웹서버와 통신하는 방법을 채택했다.

웹 서버는 Tomcat과 JSP를 사용해서 구현했다. JSP는 웹 응용프로그램을 만드는 기술로 웹 서버 안에서 서블릿을 이용해서 웹 페이지의 모양과 특정 로직을 실행하든지 서버에서 DB에 접근할 때 자바 프로그램을 호출하는 내용을 생성한다. JSP는 PHP, ASP와 다르게 자바를 사용해서 개발하기 때문에 함수, 인터페이스 부분이 잘 되어 있어서 응용프로그램을 개발하기도 용이하고 개발 후 유지 보수가 쉽다. JSP로 웹 서버에서 사용할 페이지를 만들었다면 Tomcat은 JSP로 만든 페이지를 서비스해주는 서버 역할을 한다.

데이터베이스는 사용자가 웹 서버로 전송한 결과를 파싱해서 저장할 공간이다. 데이터베이스는 Mysql을 사용해서 구현했다.

4.1. 웹 서버

4.1.1. Input Page

이 페이지는 클라이언트에서 HTTP를 사용해서 전송한 데이터를 받게 된다. 데이터를 받으면 토큰에 의해서 어플리케이션 명, 패키지 명, 실행 시간, 전력 소모량을 구분해서 하나의 객체로 생성한다. 또 다른 토큰으로 어플리케이션들을 구분해서 사용자가 측정한 어플리케이션을 하나의 어플리케이션 당 하나의 객체로 만들며 객체들은 리스트를 사용해서 관리된다. 전송 받은 데이터의 파싱이 끝나면 데이터베이스에 값을 삽입하는 절차가 진행된다. 데이터베이스와 연결을 맺고 리스트의 객체들을 하나씩 데이터베이스에 삽입한다. 삽입이 끝나면 데이터베이스와 연결을 종료한다. 객체를 생성하는 과정 없이

파싱하고 데이터베이스에 삽입하는 과정을 반복하는 방법도 있다. 하지만 이 방법을 사용하지 않은 이유는 파싱하는 시간만큼 데이터베이스에 계속 연결되어 있어야 한다. 데이터베이스와의 연결을 최소화하기 위해서 파싱 후 객체를 생성하고 데이터베이스에 삽입하는 방법을 선택했다.

4.1.2. Output Page

이 페이지는 서버에서 클라이언트로 데이터를 전송하는데 사용된다. 사용자가 이 페이지로 요청을 보내면 데이터베이스에서 데이터를 가져오는데 쿼리를 사용해서 한 번에 하나의 어플리케이션에 대한 데이터를 가져온 후 powerDTO클래스를 사용해서 객체로 생성한다. 모든 어플리케이션에 대한 객체가 생성되면 Output Page에서 클라이언트에게 HTTP를 사용해서 전송한다.

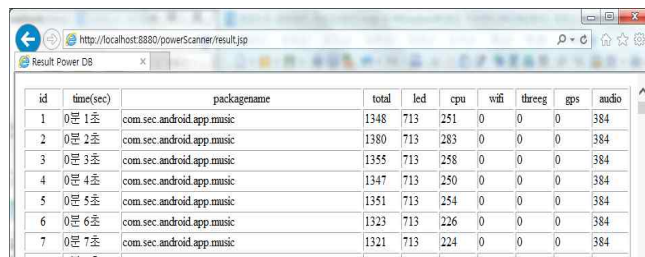
4.1.3. Result Page

이 페이지는 데이터베이스에 저장된 데이터를 어플리케이션 없이도 확인할 수 있도록 제공해주기 위한 페이지로 웹 브라우저에서 Result Page에 접속하면 데이터베이스에 저장된 데이터를 사용자에게 제공해준다. 2 종류의 페이지로 제공하는데 어플리케이션과 평균 전력 소모량 데이터가 첫 번째이고 그 다음은 매 초당 측정된 데이터를 제공하는 것이다.



id	packagename	avg_power
1	com.sec.android.app.music	1344.07
2	com.cjenm.ModooMarbleKakao	1368.17
3	com.cjenm.cowgirl	1368.39
4	com.facebook.katana	980.276666666
5	com.nhn.android.nmap	1359.08
6	com.sec.android.app.camera	963.333333333
7	com.google.android.youtube	1288.083333333
8	com.sec.android.app.videolayer	1319.67

그림 4. 어플리케이션
평균 전력 소모량



id	time(sec)	packagename	total	led	cpu	wifi	threeg	gps	audio
1	0분 1초	com.sec.android.app.music	1348	713	251	0	0	0	384
2	0분 2초	com.sec.android.app.music	1380	713	283	0	0	0	384
3	0분 3초	com.sec.android.app.music	1355	713	258	0	0	0	384
4	0분 4초	com.sec.android.app.music	1347	713	250	0	0	0	384
5	0분 5초	com.sec.android.app.music	1351	713	254	0	0	0	384
6	0분 6초	com.sec.android.app.music	1323	713	226	0	0	0	384
7	0분 7초	com.sec.android.app.music	1321	713	224	0	0	0	384

그림 5. 초당 측정된 데이터

4.2. 데이터베이스

4.2.1. 테이블 구성

데이터베이스는 2개의 테이블을 사용했다. 첫 번째 테이블인 App 테이블은 패키지명, 평균 전력 소모량, 평균 CPU 사용량, 평균 LED 밝기로 구성된다. 두 번째 테이블인 Power 테이블은 패키지명, 총 전력 소모량, 측정 시간, LED, CPU, WIFI, 3G, GPS, AUDIO의 전력 소모량으로 구성된다. 첫 번째 테이블은 실제로 사용자에게 제공할 평균 전력 소모량을 위한 테이블이고 두 번째 테이블은 테스트를 위해서 만든 테이블로 초당 사용량을 측정해서 사용량을 보기 위해서 만든 테이블로 실제로 사용자에게는 첫 번째 테이블을 사용해서 제공될 것이다.

The screenshot shows the MySQL Workbench interface. On the left, the 'Object Browser' displays the 'power' table structure with columns: id, package_name, total, led, cpu, wifi, threeg, gps, audio, and time. The main window shows the 'power' table data with 20 rows. The bottom panel shows the 'Output' window with query results.

id	package_name	total	led	cpu	wifi	threeg	gps	audio	time
1	com.sec.android.app.music	1348	713	251	0	0	0	384	1
2	com.sec.android.app.music	1380	713	283	0	0	0	384	2
3	com.sec.android.app.music	1355	713	258	0	0	0	384	3
4	com.sec.android.app.music	1347	713	250	0	0	0	384	4
5	com.sec.android.app.music	1351	713	254	0	0	0	384	5
6	com.sec.android.app.music	1323	713	226	0	0	0	384	6
7	com.sec.android.app.music	1321	713	224	0	0	0	384	7
8	com.sec.android.app.music	1335	713	238	0	0	0	384	8
9	com.sec.android.app.music	1343	713	246	0	0	0	384	9
10	com.sec.android.app.music	1324	713	227	0	0	0	384	10
11	com.sec.android.app.music	1365	713	268	0	0	0	384	11
12	com.sec.android.app.music	1320	713	223	0	0	0	384	12
13	com.sec.android.app.music	1318	713	221	0	0	0	384	13
14	com.sec.android.app.music	1375	713	278	0	0	0	384	14
15	com.sec.android.app.music	1327	713	230	0	0	0	384	15
16	com.sec.android.app.music	1317	713	220	0	0	0	384	16
17	com.sec.android.app.music	1311	713	214	0	0	0	384	17
18	com.sec.android.app.music	1321	713	224	0	0	0	384	18
19	com.sec.android.app.music	1324	713	227	0	0	0	384	19
20	com.sec.android.app.music	1317	713	220	0	0	0	384	20

power 5

Output

Time	Action	Message	Duration / Fetch
5 22:44:02	select * from power LIMIT 0, 99999	20700 row(s) returned	0.015 sec / 0.047 sec
6 22:44:05	select * from power_storage LIMIT 0, 99999	18900 row(s) returned	0.000 sec / 0.047 sec
7 22:44:10	select * from power LIMIT 0, 99999	20700 row(s) returned	0.000 sec / 0.031 sec

그림 6. Power Table

MySQL Workbench

SQL Editor (localhost) x

File Edit View Query Database Plugins Scripting Help

Object Browser

SCHEMAS

Search objects

powerscanner

Tables

app

Columns

id

package_name

avg_power

avg_cpu

avg_led

Indexes

Foreign Keys

Triggers

power

power_storage

Views

Routines

Information

No object selected

Object Info

Session

Query Completed

sqlquery*

Filter:

id	package_name	avg_power	avg_cpu	avg_led
1	com.sec.android.app.music	1344.07	247.07	713
2	com.cjenn.ModoMarbleKakao	1358.17	261.17	713
3	com.cjenn.cowgirl	1368.39	271.39	713
4	com.facebook.katana	980.276666666	229.896666666	713
5	com.nhn.android.nmap	1359.08	217.08	713
6	com.sec.android.app.camera	953.333333333	240.333333333	713
7	com.google.android.youtube	1286.093333333	207.013333333	713
8	com.sec.android.app.videoplayer	1312.67	215.67	713
9	com.sec.android.app.dmb	1419.236666666	319.193333333	713
10	net.cj.qhvw.gs.tving	1327.276666666	234.116666666	713
11	com.facebook.katana	715.34	248.426666666	452.83333...
12	daum.uid.shared:10139	1022.86	240.953333333	475
13	com.ftt.mystkakao_and	1121.74	287.06	475
14	com.sundaytoz.mobile.anipang...	1125.546666666	266.546666666	475
15	com.ktmusic.dosirakg	710.646666666	194.623333333	134.58333...
16	com.ubivelox.megabox	357.19	114.156666666	238
17	net.daum.android.air	753.38	198.43	545.41666...
18	ahnmobilesecurity.app.shared...	814.793333333	213.86	590.93333...
19	com.ms.coupleobserver	714.513333333	232.18	472.33333...
20	kr.co.seworks.sguard	631.206666666	240.773333333	380.56666...

app 6 x

Apply Cancel

Output

Action Output

Time	Action	Message	Duration / Fetch
6 22:44:05	select * from power_storage LIMIT 0, 99999	18900 row(s) returned	0.000 sec / 0.047 sec
7 22:44:10	select * from power LIMIT 0, 99999	20700 row(s) returned	0.000 sec / 0.031 sec
8 22:45:12	select * from app LIMIT 0, 99999	69 row(s) returned	0.000 sec / 0.000 sec

그림 7. App 테이블

5. 파워스캐너 테스트

5.1. 테스트 목적

파워스캐너 프레임워크의 필요성을 검증하기 위해 실험을 진행한다. 이 실험은 스마트폰에서 사용되는 응용 프로그램들의 전력 소모량이 다양한 분포를 갖는다는 점을 분석하고 동일 유형의 응용 프로그램들에 대한 전력 소모량을 측정함으로써 유사한 기능을 제공하는 응용 프로그램들 간에 전력 소모량이 다양함을 확인하기 위한 실험이다.

5.2. 테스트 환경

개발 OS : Windows 7

개발 도구 : eclipse-jee-kepler-R-win32-x86_64, Tomcat 7.0.42

테스트 기기 : 구글 레퍼런스폰, 안드로이드 버전 2.1

테스트 대상 : 안드로이드 마켓에서 상위권에 속하는 40개의 응용 프로그램.

5.3. 테스트 결과

실험을 위해서 파워스캐너의 측정 기능을 사용해서 얻은 전력 소모량을 서버로 전송하여 각 응용프로그램 별로 전력 소모 데이터에 대한 평균값을 계산하여 사용했다.

그림 8은 40개의 응용 프로그램에 대한 정규화된 평균 전력 소모 누적분포도이다. 그림 8에서 X축은 정규화된 평균 전력 소모율이며 Y축은 응용 프로그램의 누적 분포율을 나타낸다. X축의 값은 최대 전력 소모량, 응용 프로그램의 평균 전력 소모량 그리고 최소 전력 소모량을 이용하여 식 (1)로 계산한다.

$$\frac{AppAveragePower - MinPower}{MaxPower - MinPower} * 100\% \quad (1)$$

식 (1)에 의해서 X축은 0부터 100(%) 사이의 값으로 나타내고 100(%)에 가까워질수록 전력 소모가 심함을 나타낸다.

그림 8을 통해서 응용 프로그램들의 전력 소모량이 다양한 분포를 갖는다는 것을 알 수 있다. 그리고 일정 구간에서 그래프의 기울기가 클수록 응용 프로그램이 많이 분포한다는 것인데, 이를 통하여 평균 전력 소모율이 40%와 90%인 구간에 응용 프로그램이 많이 분포함을 알 수 있다. 응용 프로그램들 사이에도 전력 소모 차이가 크다는 것을 알 수 있다.

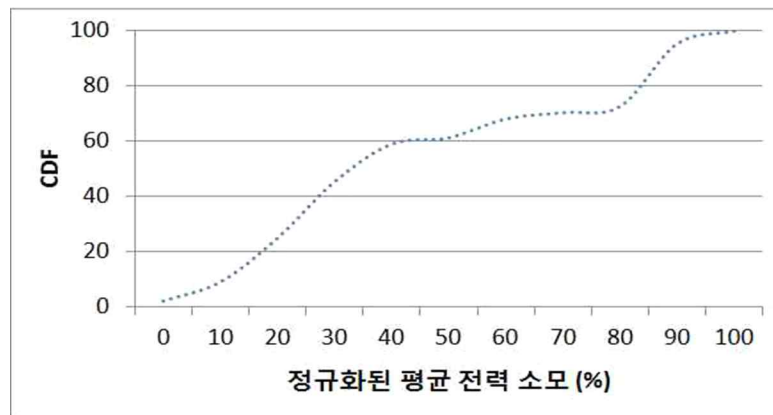


그림 8. 정규화된 평균 전력 소모 누적분포도

그림 9는 동영상 카테고리 응용 프로그램의 평균 전력 소모량을 그린 그래프이다. 빨간색으로 표시한 계열은 스트리밍 서비스 응용 프로그램이며, 파란색으로 표시한 계열은 로컬 서비스를 제공하는 응용 프로그램을 나타낸다. 실험에 사용된 데이터는 모두 동일한 동영상을 사용했으며, 실험 환경으로는 동일한 스마트폰을 이용했으며 LCD와 Audio 컴포넌트는 모두 동일하게 최대치를 유지시켰다.

그림 9를 통해서 유사한 기능을 하는 응용 프로그램들 사이에도 전력 소모량의 차이가 있음을 알 수 있었다. 평균 전력 소모량이 가장 많은 Video Player와 가장 적은 BS Player의 평균 전력 소모량의 차이는 약 73mW이다. 두 응용 프로그램이 동일한 로컬 서비스를 제공하는 데에도 전력 소모

차이가 나는 이유는 여러 가지가 있을 수 있다. 실험에 사용된 응용 프로그램의 경우 안드로이드 마켓에 있는 개발자의 설명을 참고하자면, 외부코덱을 사용하기 때문에 발생하는 CPU의 사용률의 차이가 있음을 알 수 있었다. 그리고 하드웨어 코덱을 사용하는 경우와 소프트웨어 코덱을 사용하는 경우에 따라 전력 소모의 차이가 있음을 확인했다. 부가적으로 같은 응용 프로그램에서도 광고가 삽입된 무료 버전과 광고가 없는 유료 버전의 경우에도 전력 소모량의 차이가 있었다. 이를 통해서 광고가 전력 소모량에 영향을 준다는 것을 확인했다.

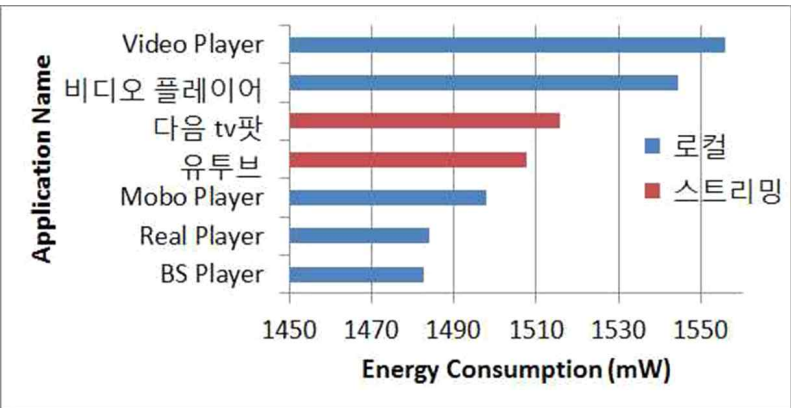


그림 9. 동영상 카테고리 응용 프로그램의 평균 전력 소모량

6. 결론

본 연구를 통해서 스마트폰에서 사용되는 응용 프로그램마다 전력 소모량의 차이가 있음을 확인했다. 또한 유사한 기능을 하는 응용 프로그램 사이에서도 전력 소모량의 차이가 있는 것을 확인했다. 이 결과를 통해서 본 논문이 제시하는 파워스캐너 프레임워크의 실효성을 검증할 수 있었다. 즉, 실제로 각각의 응용프로그램들 사이에서 전력 소모량의 차이가 발생하는 것을 확인했기 때문에 연구 결과인 파워스캐너 프레임워크가 제공해주는 전력 소모량 데이터는 사용자가 스마트폰의 전력 소모를 줄이기 위한 방법으로 사용이 가능하다. 서로 다른 응용 프로그램에서도 차이가 있지만 유사한 기능을 하는 응용 프로그램에서도 차이가 있다. 테스트 결과처럼 동영상 플레이어 사이에서도 서로 다른 전력 소모량을 갖고 있었다.

사용자가 스마트폰을 사용하는 동안에 전력 소모량이 많거나 다른 영향이 있다고 해도 쉽게 바꾸기 힘든 응용 프로그램이 있다. 예를 들면, 다른 사람과의 의사소통을 하는 메신저나 SNS의 경우는 나 혼자서 바꾸기가 어려운 경우에 속한다. 하지만 음악, 동영상 플레이어와 같은 개인적으로 사용하는 응용 프로그램들은 본연의 기능에 충실하다면 그 중에서 전력 소모량이 적은 응용 프로그램을 선택해서 스마트폰 사용 시간을 늘리는데 도움이 될 수 있다.

파워스캐너 프레임워크 적용에는 몇 가지 어려움이 있다. 그 중 하나는 사람들이 사용하는 스마트폰의 종류가 서로 다르기 때문이다. 스마트폰의 하드웨어 성능에 따라 LED, AUDIO 등 각 종 컴포넌트가 소비하는 전력량이 다르기 때문에 하드웨어에 대한 정확한 정보가 없다면 사용자에게 제공 되는 응용 프로그램의 전력 소모량 정보는 상대적인 비교에만 사용이 가능하다. 즉, 현재 존재하는 스마트폰에 대한 정보가 필요한 것이다. 또 하나의 어려움은 데이터 수집에 문제가 있다. 사용자에게 가장 정확한 전력 소모 정보를 제공하기 위해서는 수집 단계에서도 스마트폰 기종 별로 수집이 되어야 하며 제공 또한 그렇게 제공 되어야 한다. 하지만 수집 단계에서 사람들이 많이 사용하지 않는 오래

된 스마트폰이나 비인기 스마트폰의 경우에는 사용자가 적어서 수집에 어려움이 있다. 이 부분에 대한 해결방법이 필요 할 것 같다. 이런 경우에는 비슷한 하드웨어를 갖는 기계를 통합해서 제공하는 방법 등을 통해서 해결이 가능할 것 같다.

현재 파워스캐너 프레임워크는 사용자에게 전력 소모 정보만을 제공하고 있다. 수집된 정보를 단순히 제공만 하기 보다는 카테고리 별 전력 소모량을 제공하거나 현재 설치된 응용 프로그램을 대체할 수 있는 추천 목록을 제공하는 방법을 개발할 필요가 있다.

7. 참고문헌

- [1] L. Zhang, et al., "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in IEEE/ACM, CODES/ISSS, pp.105-114, 2010.
- [2] L. Jaymin, et al, "Smart Phone Power Model Generation Using Use Pattern Analysis," IEEE International Conference on Consumer Electronics, Jan, 2012.
- [3] P. Abhinav, et al., "Where is the energy spent inside my app? Fine Grained Energy Accounting on Smartphones with Eprof," EuroSys'12, pp.29-42, 2012.
- [4] C. Yoon, et al., "AppScope: Application Energy Metering Framework for Android Smartphone using Kernel Activity Monitoring," USENIX ATC'12.
- [5] PowerTutor, <http://ziyang.eecs.umich.edu/projects/powertutor/>