모바일 시스템 프로그래밍

07 Power Manager

2017 1학기

강승우

Computer Science and Engineering, KOREATECH

모바일 기기와 배터리

- 모바일 → 휴대성
  - 기기 동작을 위해서 외부 전원 소스가 아니라 전원을 내장하고 있어야 함
  - 내장 전원으로 배터리 사용

- 배터리 동작 시간이 모바일 기기 사용에 중요
  - 배터리 완충 후 방전까지 얼마나 사용할 수 있나

- 노트북 PC vs. 스마트폰 (태블릿)
  - 휴대성 측면
  - 이동성 측면
- 피처폰 vs. 스마트폰

파워 관리 (Power Management)

- 배터리의 성능
  - 기본적으로는 단위 부피당 용량을 늘리는 것이 중요
  - 지금까지 꾸준히 성능 향상이 있어왔지만, 모바일 기기 동작을 위해 요구되는 배터리 소모량도 증가
  - 제한된 용량을 갖는 배터리를 최대한 효율적으로 활용하기 위한 H/W, S/W 측면의 관리가 매우 중요
- 파워 관리 (power management)
  - 배터리로 동작하는 모바일 기기의 사용 시간을 증가하기 위해서 소프트웨어 및 하드웨어 측면에서 활용되는 방법/기술/시스템

파워의 정의

- 파워
  - 물리학에서 단위 시간당 일의 양으로 정의
  - 일률이라고도 함
  - 단위: W(와트)

  - 파워를 전기 관점에서 표현하면?
  - 전력
  - 전력량

  - 전력, 전력량을 어떻게 계산할까?

파워의 정의

- 전력
  - 단위 시간당 전류가 할 수 있는 일의 양
  - 1W = 1V x 1A
    - 1V 전압을 통해 흐르는 1A의 전류에 의한 일의 양
  - $P = VI = I^2R$
- 전력량
  - 일정한 시간 동안 사용한 전력의 양
  - Wh
  - 1kWh: 1kW 소비전력을 갖는 전기 제품을 1시간 동안 사용했을 때의 전력량

배터리

- 배터리
  - 물질의 화학적 혹은 물리적 반응을 이용하여 이들의 변화로 방출되는 에너지를 전기 에너지로 변환하는 소형 장치
  - 이온화 경향이 다른 두 금속 간 산화 환원 반응 및 이 과정에서 물질 간 이동하는 전자의 흐름을 이용
    - 이온화 경향: 전자를 버리려는 성질의 정도
    - 산화: 전자를 잃는 것
    - 환원: 전자를 얻는 것
- 종류
  - 전기 에너지로 변환하는 방법
  - 화학 전지: 화학적 반응 이용
  - 물리 전지: 물리적 반응 이용

배터리

- 화학 전지
  - 1차 전지: 일회용 배터리
    - 리튬 망간
    - 알카리 망간
  - 2차 전지: 충전 가능한 배터리
    - 납축 전지
    - 니켈 카드뮴, 니켈 아연, 니켈 수소
    - 리튬 이온
    - 리튬(이온) 폴리머
      - 셀 디자인이 쉬움 (모양을 비교적 자유롭게 만들 수 있음)
      - 전해질이 준고체 상태이기 때문에 용액이 잘 새어 나오지 않음

[표] 리튬-폴리머 배터리와 리튬-이온 배터리 비교

구분	리튬-폴리머 배터리	리튬-이온 배터리
음극	탄소	
양극	$LiCoO_2$, $LiNiO_2$, $LiMn_2O_4$ 등 금속산화물	
전해질	Polymer(고분자)	액체
전압	3.7V	
에너지 밀도	높다	
라이프 사이클	좋다	매우 좋다
전온특성	강하다	약하다
안정성	좋다	나쁨(폭발위험)
셀 디자인 다양성	쉽다	어렵다
장점	· 높은 안정성 · 종이처럼 가벼워 어떤 모양도 가능	· 광범위한 온도특성 (저온도 강함)
단점	· 저온에서 사용특성 떨어짐	· 폭발사고

배터리 용량

- mAh
  - 한 시간 동안 표시된 숫자만큼의 전류를 제공할 수 있다는 의미

3.7VDC, 1800mAh

3.7V, 5.55Wh 이것은 몇 mAh?

3.82V
6.91Whr
→ 1810mAh

배터리 사용 시간

- 배터리 용량과 전류 소모량 혹은 전력 소모량이 주어졌을 때 배터리 사용 시간은 어떻게 계산할 수 있나?

- 배터리 용량
  - 3.7V, 1800mAh

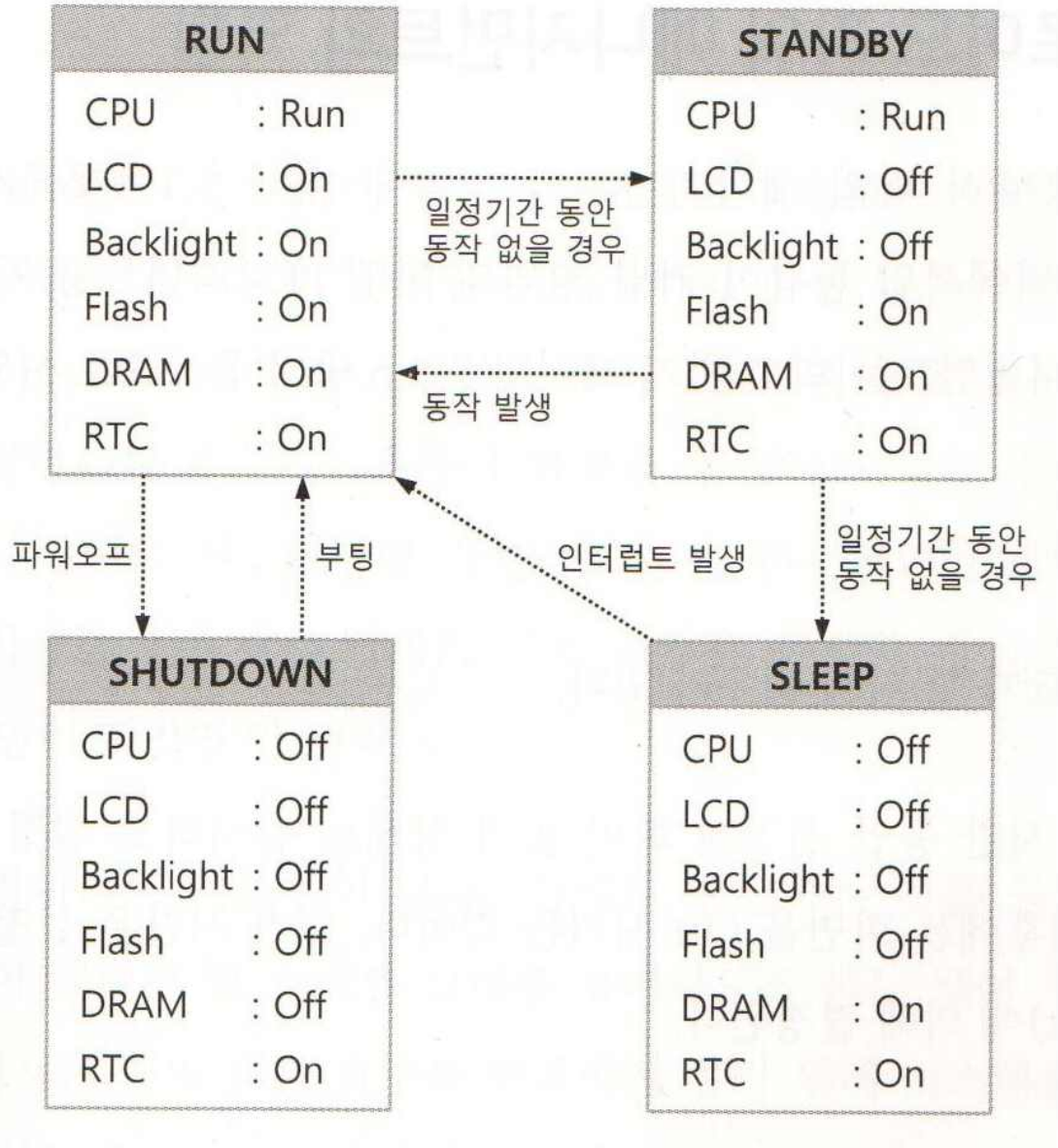
- 안드로이드 시스템과 애플리케이션의 전류 소모량
  - 600mA

- 배터리 사용 시간?

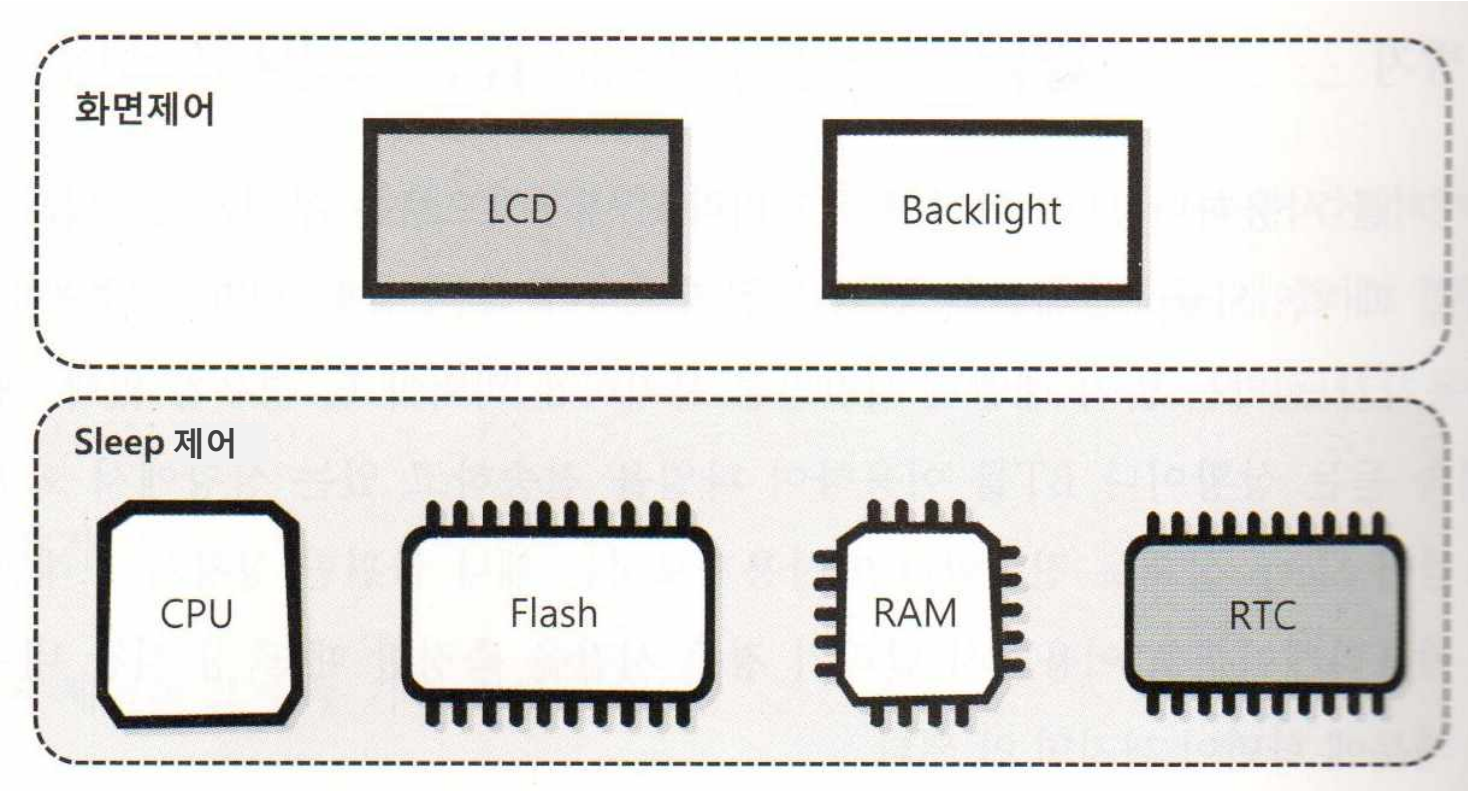
기본적인 파워 상태와 파워 관리

- Run
  - 시스템이 동작 중인 상태
  - 전류 소모가 최고에 이르는 상태
- Standby
  - 시스템이 비활성화 된 상태
  - LCD, Backlight가 Off, CPU 클록 감소
- Sleep
  - 시스템이 Standby 상태가 지속될 경우 대부분 주변장치들의 전원 공급 차단
  - CPU도 Sleep 상태 진입
  - 램이나 레지스터 같은 휘발성 메모리는 전원의 공급이 선택적
- Shutdown
  - 시스템 파워 off 된 상태

파워 상태도


RUN
CPU : Run
LCD : On
Backlight : On
Flash : On
DRAM : On
RTC : On
STANDBY
CPU : Run
LCD : Off
Backlight : Off
Flash : On
DRAM : On
RTC : On
일정기간 동안
동작 없을 경우
동작 발생
파워오프
부팅
인터럽트 발생
일정기간 동안
동작 없을 경우
SHUTDOWN
CPU : Off
LCD : Off
Backlight : Off
Flash : Off
DRAM : Off
RTC : On
SLEEP
CPU : Off
LCD : Off
Backlight : Off
Flash : Off
DRAM : On
RTC : On

파워 관리의 2가지 측면



기본적인 파워 절약 정책과 예외 상황

- Run → Standby → Sleep
  - 사용자의 조작이 없을 때 화면이나 키보드 조명을 차단 (Standby 상태)
  - Standby 상태로 일정 시간이 지나면 CPU off, WiFi off

- 이러한 정책에 따라 동작할 수 없는 앱 존재
  - 예) 동영상 플레이어, 네비게이션
  - 사용자가 화면을 터치하지 않는다고 화면을 끄면 안 됨

  - 화면이 꺼지더라도 CPU는 계속 동작해서 백그라운드 작업을 수행해야 하는 앱들도 있음

안드로이드 PowerManager 클래스

- 안드로이드 디바이스의 파워 상태를 제어할 수 있도록 하는 API 제공
- 사용법
  - Context.getSystemService() 메소드를 이용하여 PowerManager 객체를 생성
  - PowerManager의 newWakeLock() 메소드를 이용하여 PowerManager.WakeLock 객체 생성
  - WakeLock의 메소드를 이용하여 파워 상태 제어
    - acquire(), release()
- permission 필요
  - android.permission.WAKE_LOCK
  - 메니페스트 파일에 <uses-permission android:name="android.permission.WAKE_LOCK"/> 추가

- ***** 주의사항 *****
  - PowerManager API 사용으로 인해 배터리 사용 시간에 큰 영향을 받을 수 있음
  - 반드시 필요한 경우에만 wakelock을 사용하고, 가장 낮은 레벨로 사용하며, 사용 후 반드시 release 해야 함

https://developer.android.com/reference/android/os/PowerManager.html?hl=ko

PowerManager와 wake lock 사용법

```
PowerManager pm;
PowerManager.WakeLock wl;

pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
wl = pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, "Tag: partial wake lock");

wl.acquire();

// CPU on
// doing a job you want

wl.release();
```

newWakeLock 메소드

public PowerManager.WakeLock newWakeLock (int levelAndFlags, String tag)

- Wake lock을 생성
- levelAndFlags 파라미터
  - wake lock 레벨과 옵션 플래그 지정 (logical OR operator 사용)
  - 레벨 (PowerManager 클래스에 정의된 constant) - 반드시 하나의 레벨만 지정해야 함
    - PARTIAL_WAKE_LOCK
    - FULL_WAKE_LOCK
    - SCREEN_DIM_WAKE_LOCK
    - SCREEN_BRIGHT_WAKE_LOCK
  - 플래그 (PowerManager 클래스에 정의된 constant) – 스크린 동작에만 영향, partial wake lock의 사용에는 아무 영향 없음
    - ACQUIRE_CAUSES_WAKEUP
    - ON_AFTER_RELEASE

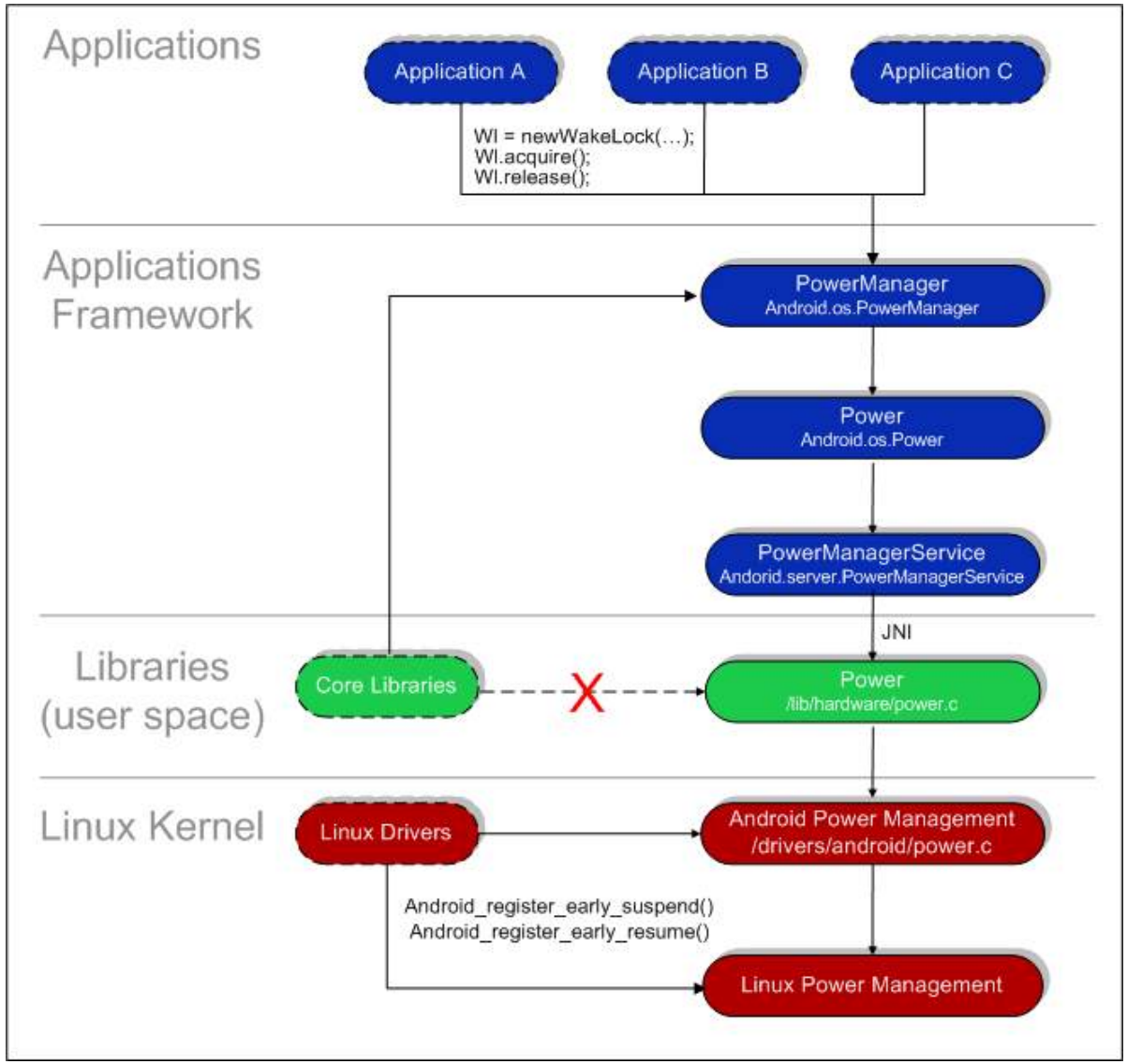
```
PowerManager.WakeLock wl =
pm.newWakeLock(PowerManager.SCREEN_DIM_WAKE_LOCK|PowerManager.ON_AFTER_RELEASE, TAG);
```

Wake lock 레벨과 플래그

Value	CPU	Screen	Keyboard
PARTIAL_WAKE_LOCK	On	Off	Off
SCREEN_DIM_WAKE_LOCK	On	Dim	Off
SCREEN_BRIGHT_WAKE_LOCK	On	Bright	Off
FULL_WAKE_LOCK	On	Bright	Bright

- PARTIAL_WAKE_LOCK의 경우 디스플레이 타임아웃이나 스크린 상태에 상관 없이 CPU가 계속 on (사용자가 파워 버튼을 눌러서 화면을 끄더라도 CPU는 계속 돌아간다)
- 다른 wake lock의 경우 파워 버튼을 누르면 sleep 상태로 변경된다
  - 사용자가 파워 버튼을 누르면, 시스템에 의해서 lock이 release됨. screen과 CPU는 off

- ACQUIRE_CAUSES_WAKE_UP
  - 보통 wake lock은 켜진 화면을 계속 켜져 있게는 하지만 꺼진 화면을 강제로 켜지는 않는다
    이 플래그가 사용되면 wake lock이 acquire() 되면 스크린과 키보드가 바로 켜진다
    notification 등이 있을 때 주로 사용
- ON_AFTER_RELEASE
  - wake lock이 release된 후 화면 타이머를 reset하여 화면이 꺼지는 시간을 좀 늦춘다



http://www.netmite.com/android/mydroid/development/pdk/docs/power_management.html

화면 제어

- 화면과 관련된 wake lock은 현재 deprecated
  - FULL_WAKE_LOCK
  - SCREEN_DIM_WAKE_LOCK
  - SCREEN_BRIGHT_WAKE_LOCK

- wake lock을 안 쓰고 화면을 on 상태로 유지하는 방법
  - 동영상 플레이어, 게임 앱 등
  - Activity에서 FLAG_KEEP_SCREEN_ON 사용
    - service 혹은 다른 app component에서 사용하면 안 됨
  - 장점
    - permission을 필요로 하지 않음
    - 안드로이드 플랫폼이 알아서 관리해줌 (app에서 명시적으로 release 하지 않아도 됨)

FLAG_KEEP_SCREEN_ON 사용법

- Activity onCreate에서 코드로 설정

```
public class MainActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
  }
```

- 레이아웃 XML 파일에서 설정

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true">
    ...
</RelativeLayout>
```

PowerManager 이용

- 실내 장소에 대한 근접 경보 예제를 다시 살펴보자
  - 예제 프로젝트 이름
    - MSP06IndoorProximityAlert
  - 기본 기능
    - Wifi AP 정보를 이용하여 proximity alert을 발생할 장소를 등록
    - 주기적으로 Wifi scan을 수행하여 현재 위치가 alert 등록 장소인지 검사
    - 현재 위치가 alert 등록 장소로 판단되면 alert
  - 구현 시 생각해 볼 점에서 4번 문제에 대한 해결 방법은??
    - 4. 화면이 꺼진 경우에도 동작하게 하고 싶음. 어떻게?
  - 위 예제 프로젝트를 수정하여 화면이 꺼진 경우에도 근접 경보가 동작하도록 만들어보자