Vol.2016-CDS-16 No.11
2016/6/2

Alarm 機能の観察による携帯端末の無操作時消費電力の増加の原因となるアプリケーションの推定

栗原駿[†1]　福田翔貴[†1]　濱中真太郎[†1]　小口正人[†2]　山口実靖[†1]

スマートフォンが普及し，その重要性が増している．スマートフォンの最大の不満点はバッテリの持ち時間であるとの報告もあり，スマートフォンの消費電力の低減は非常に重要な課題と言える．Android 搭載のスマートフォンには，ユーザが操作していない時間帯にもアプリケーションが動作する仕組みが備わっており，無操作状態でも電力が消費される．本研究では無操作状態における Android 端末の電力消費に着目し，無操作時消費電力を増加させる量が大きいアプリケーションの推定方法について考察する．具体的には，Android OS を改変し，無操作状態で 24 時間放置した時のアプリケーション毎の Alarm の起動回数を観察することで，無操作時消費電力を大きく増加させるアプリケーションを推定する手法を提案する．そして，性能評価により提案手法の有効性を検証する．

キーワード：Android OS，消費電力

Identifying Battery-Draining Applications by Monitoring Alarm in Screen-Off State in Android

Kurihara Shun[†1]　Fukuda Shoki[†1]　Hamanaka Shintaro[†1]
Oguchi Masato[†2]　Yamaguchi Saneyasu[†1]

Abstract: Android OS has function with which an application can work in screen-off state without user's operation. Some applications frequently work in screen-off state, and consume battery. In this paper, we propose a method for identifying applications which largely drain battery in Screen-off state. The method monitors setting and invoking alarm, which is a common method for executing an application in screen-off state, and estimate the power consumption of each application with the monitoring results. We evaluate our estimation and demonstrate that our method can identify power draining applications more correctly than the standard method of Android operating system.

Keywords: Android, Power Consumption

1. はじめに

近年，スマートフォンやタブレット PC が普及し，これらは重要な情報端末プラットフォームとなっている．スマートフォンの最大の課題は「バッテリの持続時間である」との報告[1]があり，スマートフォンにおける消費電力の低減は非常に重要な課題であると考えられる．多くのスマートフォン OS は指定時刻にアプリケーションを起動させる仕組みが用意されており，多くのアプリケーションにおいてユーザが直接操作を行わなくても動作する機能が備わっている．これらのユーザの直接的操作を伴わないアプリケーションの起動により端末のバッテリが消費されるが，ユーザが直接操作をしていない状態における端末やアプリケーションの動作の把握はユーザには困難であると予想できる．よって，無操作状態において多くの電力を消費するアプリケーションの特定は，重要な課題の一つであると考えられる．

スマートフォン上で動作する代表的なソフトウェアプラットフォームの一つに Android OS がある．Android OS の 2015 年第 2 四半期におけるスマートフォン OS 世界市場におけるシェアは 82.8%に達しており[2]，スマートフォンにおける代表的なプラットフォームの一つとなっている．本稿では，Android OS 搭載端末における無操作時消費電力に着目し，これを大きく増加させるアプリケーションの特定手法について考察を行う．具体的には，端末が Sleep 状態でも機能する AlramManager に着目し，Alarm の起動回数の観察による無操作時消費電力の大きいアプリケーションを推定する手法を提案する．

本稿の構成は以下の通りである．2 章にて，無操作状態の Android 端末における電力消費の原因となる仕組みである AlarmManager について説明する．3 章にて，無操作時消費電力の調査結果を述べ無操作時消費電力の主な原因はアプリケーションによる Alarm であることを示す. 4 章にて，Alarm の観察による無操作時消費電力を増加させるアプリケーションの推定手法を提案し，5 章にて推定手法の評価を示す．6 章で考察を，7 章で関連研究を述べ，8 章にて本稿をまとめる．

2. 無操作時消費電力

2.1 無操作時消費電力の把握

Android 端末では無操作時においてもアプリケーションが動作し多くの電力が消費されるが，ユーザが直接観察していない状況におけるアプリケーションの動作によるものであるため，ユーザによるその状況の把握は困難である．

†1 工学院大学
Kogakuin University

†2 お茶の水女子大学
Ochanomizu University

また，OS の標準機能としてアプリケーション別の消費電力を調査する機能も提供されているが，5 章で述べるように必ずしも正確には見積もることができず，無操作時消費電力を大きく増加させるアプリケーションの特定は困難である．

2.2 無操作時消費電力を増加させるアプリケーション

無操作状態が続くと Android 端末は Sleep 状態に入り，省電力モードとなる．Sleep 状態では，バッテリの主な消費原因である CPU 稼働，ディスプレイ点灯，通信が抑制される．この Sleep 状態への移行を禁止する機能として WakeLock 機能があり，Sleep 状態にてアプリケーションに指定の処理をさせる機能として Alarm 機能が用意されている．

WakeLock は端末が省電力モード(sleep 状態)に移行することを禁止する機能であり，本機能の使用が端末の消費電力を増加させると指摘されている[3]．また我々の過去の調査により，WakeLock の多くが Alarm 機能により起動された SystemServer プロセスによるものである[4][5]ことが分かっている．よって，「Alarm の設定や Alarm によるアプリケーションの起動が無操作時消費電力を大きく増加させる」と予想することができる．

2.3 AlarmManager

Alarm は，アプリケーションが将来のある時点に自身を起動させる仕組みである[6]．アプリケーションの起動を Alarm として OS に登録することにより，指定時刻における端末の状態(Sleep，Wake)に関わらず登録したアプリケーションを指定時刻に起動させることができる．起動時刻に端末が Sleep 状態である場合，必要に応じて端末が Wakeup され，アプリケーションが起動される．この機能は，定期的に情報を取得するアプリケーションなどが指定時刻に更新や同期を行うためなどに利用される．Alarm を管理する AlarmManager には，`set()`，`setRepeating()`，`setExact()`の 3 つのアラーム設定用メソッドが用意されている．`set()`メソッドで設定されたアラームは厳密に指定時刻にアプリケーションが起動されることを保証しておらず，起動時刻を OS が調整して遅らせることがある．`setRepeating()`メソッドは指定した間隔で繰り返しアラームを設定する際に用いられ，`set()`メソッド同様に厳密に指定時刻にアプリケーションが起動されることを保証していない．`setExact()`メソッドは厳密に指定時刻に実行させたい際に用いられる．`set()`メソッドとは異なり OS による調整が行われずアプリケーションが指定時刻に起動されることを保証している．また，時刻設定用の引数を表 1 に示す．本稿では，Sleep 状態であっても端末を WAKEUP するように設定を行う“RTC_WAKEUP”と“ELAPSED_REALTIME_WAKEUP”に注目する．

表 1 時刻設定用の引数

RTC	UTC で表現される時刻で設定する
RTC_WAKEUP	RTC に加え，端末を WAKEUP させるように設定する
ELAPSED_REALTIME	reboot からの経過時刻で設定する
ELAPSED_REALTIME_WAKEUP	ELAPSED_REALTIME に加え，端末を WAKEUP させるように設定する

3. 無操作状態での端末の消費電力

本章にて無操作状態端末の消費電力の調査結果を示す．

3.1 測定方法

Android OS を改変し，無操作状態の Android 端末のバッテリ残量の推移と，アプリケーションが登録した Alarm の起動が行われた時刻と，その Alarm を登録したアプリケーションを調査した．測定は 24 時間(1440 分間)行い，測定対象アプリケーションセットは 2014 年 12 月 26 日における Google Play Store のウィジェット無料アプリケーションランキングの上位 50 件とした(これは[4][5][7]と同一である)．ただし，50 件の内 8 件のアプリケーションは端末に非対応であったため対象から外し，測定は 42 件のアプリケーションがインストールされている状態で行った．また，無操作状態となり 1 分後にディスプレイ表示がオフとなる設定とした．測定に用いた端末は Nexus7 (2013)，CPU Qualcomm Snapdragon S4 Pro 1.5GHz，メモリ 2GB，OS Android 5.0.1 である．

表 2 アプリケーション毎の Alarm 起動回数ランキング (ウィジェット)

順位	アプリケーション名	起動回数 [回]
1	A	3233
2	B	719
3	C	134
4	D	97
5	E	95
6	F	48
7	G	26
8	H	4
9	I	3
10	J	2
11	K	2
12	L	2
13	M	1
14	N	1
15	O	1
16	P	1
17	Q	1
18	R	1

3.2 測定結果

無操作状態端末の Alarm の起動回数と消費電力の関係の調査結果を図 1 に示す．“Alarm 起動回数”は，端末の状態

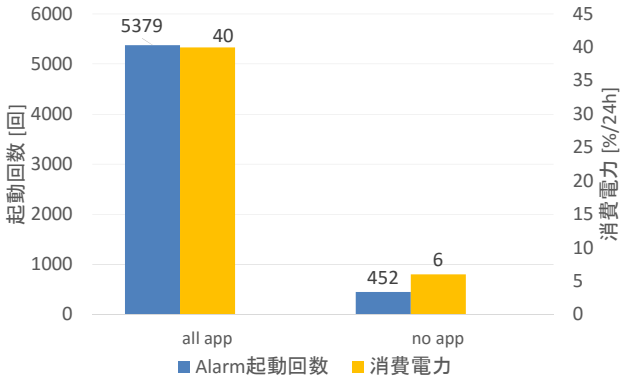

図 1 無操作状態端末の Alarm の起動回数及び消費電力

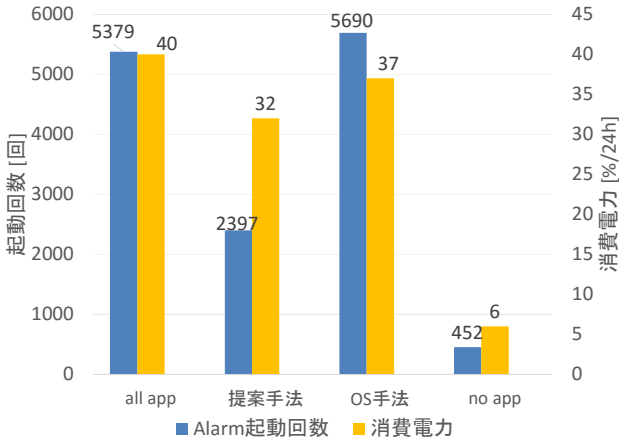

図 2 Alarm の起動回数と消費電力の関係
(ウィジェット)

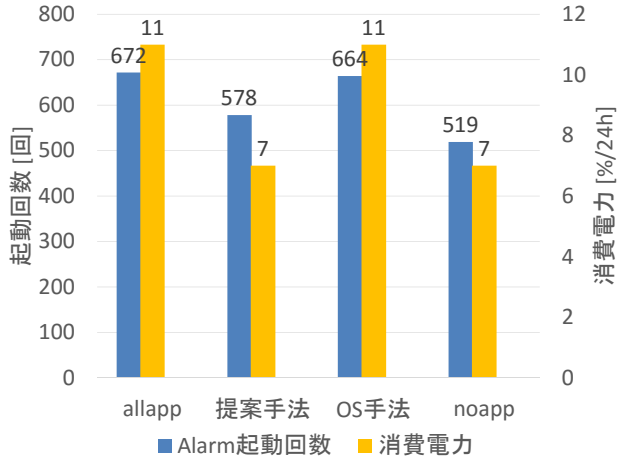

図 3 Alarm の起動回数と消費電力の関係
(ショッピング)

(Sleep, Awake)に関わらず Alarm が起動した回数を意味している．“消費電力”は，24 時間におけるバッテリ残量の減少量[%]を示している．“allapp”は OS 標準(AOSP 配布 OS 添付)のアプリケーションと上記 42 件のアプリケーションを端末に導入した状態を，“noapp”は OS 標準のアプリケーションのみをインストールした状況を示す．また表 2 に，

表 3 OS 標準機能による消費電力の大きいアプリケーションランキング(ウィジェット)

順位	アプリケーション	消費電力[%]
1	S	14
2	A	7
3	T	4

allapp のアプリケーション別 Alarm 起動回数ランキングを示す．

図 1 より，“noapp”より“allapp”の方が消費電力，Alarm 起動回数ともに大幅に上回っていることがわかる．また表 2 から，Alarm 起動の多くはアプリケーションによるものであることがわかる．以上より，無操作時消費電力の増加の原因はアプリケーションのインストールであることが分かり，Alarm の動作が大きく影響していると予想できる．

4. 無操作時消費電力の大きいアプリケーションの推定法の提案

本章で，AlarmManager によるアプリケーションの起動を観察し，無操作時消費電力が多いアプリケーションを推定する手法を提案する．

4.1 推定手法

2.2 節および 3.2 節にて，無操作状態での電力消費と Alarm によるアプリケーションの起動に関連があると予想できることを示した．我々は「Alarm 起動回数が多いアプリケーションは無操作時消費電力を大きく増加させるアプリケーションである」との仮説を立て，この仮説に基づき無操作時消費電力を大きく増加させるアプリケーションを推定する手法を提案する．

具体的には，AlarmManager によるアプリケーションの起動を観察し，起動が多いアプリケーションを無操作時消費電力が大きいアプリケーションと推定する手法を提案する．

4.2 実装

実装方法は，Android OS の `frameworks/base/services/core/java/com/android/server/AlarmManagerService.java` 内の `deliverAlarmsLocked()` メソッドを修正し，「メソッドの呼び出し時刻」と「`Alarm` 処理を行ったアプリケーション名」を記録するような改変を加えた．具体的には，上記メソッド内で取得した上記情報を `/data/data/` 内のテキストファイルに記録する修正を施した．

5. 性能評価

本章では，提案手法の性能評価を行う．

5.1 アプリケーション(ウィジェット)による評価

前章の改変を施した Android OS を用いて，アプリケーションによる alarm の起動時刻を調査，測定した．測定に用いた端末およびアプリケーションセットは 3.1 節と同様である．

提案手法により無操作時消費電力を最も大きく増加させると予想されるアプリケーションは，表 2 におけるアプリケーション A となる．OS 標準機能のアプリケーション

別の消費電力調査機能により推定されたアプリケーションは，表 3 におけるアプリケーション S であり，これは表 2 には登場していない．

Alarm の起動回数と消費電力の関係の調査結果を図 2 に示す．“提案手法”は“allapp”から提案手法により推定されたアプリケーション(表 2 の A)をアンインストールした状態を，“OS 手法”は“allapp”から OS 標準機能により推定されたアプリケーション(表 3 の S)をアンインストールした状態を示す．

図 2 の提案手法と OS 手法を比べると，Alarm の起動回数，無操作時消費電力ともに提案手法の方が大きく低減できていることが分かる．このことから，提案手法により無操作時消費電力の大きいアプリケーションをより正確に推定できたことが分かる．

5.2 アプリケーション(ショッピング)による評価

同様の評価を，対象アプリケーションを 2016 年 3 月 30 日の Google Play Store 無料アプリケーションランキング(ショッピング)の上位 50 件に変更して行った．ただし，50 件の内 8 件のアプリケーションは端末に非対応であったため対象から外し，42 件のアプリケーションを用いて評価を行った．評価結果を図 3，表 4，表 5 に示す．OS 標準の報告手法は消費電力量が 1%未満のアプリケーションはランキング内に表示しない仕様となっているため，すべてのアプリケーションを表示する様に OS 標準の調査機能の表示部を改変し表 5 を得た．

図 3 より，本アプリケーション群においても提案手法は OS 標準の調査機能より正確に無操作時消費電力の大きいアプリケーションを推定できたことが分かる．

5.3 アプリケーション(スポーツ)による評価

同様の評価を，対象アプリケーションを 2016 年 4 月 7 日の Google Play Store 無料アプリケーションランキング(スポーツ)の上位 50 件に変更して行った．このうち 2 件は端末非対応であったため，48 件のアプリケーションを用いて評価を行った．評価結果を図 4，表 6，表 7 に示す．

図 4 より，提案手法の推定によるアプリケーションのアンインストールよりも，OS 標準の調査機能の推定のアプリケーションのアンインストールの方がより大きな消費電力の削減に成功していることが分かる．提案手法が適切に推定をできなかったことに関しては，次章にて考察を行う．

6. 考察

6.1 アプリケーション実装と推定精度

5.3 節の性能評価にて，スポーツカテゴリのアプリケーション群にて提案手法が必ずしも適切に推定をできないことが確認された．表 6 を見ると，Alarm はアプリケーション a においてのみ使用されており，起動回数も 24 時間にて 4 回と非常に少ないことが分かる．よって，アプリケーション a の Alarm に起因する消費電力は非常に小さく，提案手法はアラーム起動回数が少ないアプリケーション間における推定は必ずしも正確にできないことが予想できる．これは，推定手法を「Alarm の起動回数のみで推定を行う」手法から，「OS 標準の手法と Alarm 起動回数の両方を考慮して推定を行う」手法に改善するなどにより，解決できると期待される．

OS 標準の機能が推定したアプリケーション b は，無操作状態で Alarm 機能を用いずに GPS 機能を使用し続けるアプリケーションであった．一般に Sleep 状態にてアプリケーションが起動するには Alarm 機能を用いて実装することが推奨されている[8]が，本提案手法はこれらのガイドラインに従わないアプリケーションの消費電力の推定は正確に行うことができないことが確認された．ただし，これらガイドラインに従わないアプリケーションの数は多くなく，また今後も減少していくと期待できるため，提案手法が有効である状況は少なくないと期待できる．

表 4 アプリケーション毎の Alarm 起動回数ランキング (ショッピング)

順位	アプリケーション	起動回数[回]
1	a	99
2	b	24
3	c	24
4	d	1
5	e	1

表 5 OS 標準機能による消費電力の大きいアプリケーションランキング(ショッピング)

順位	アプリケーション	消費電力[%]
1	f	0.124439
2	g	0.056983
3	a	0.042802
4	h	0.028872
5	i	0.017549
6	j	0.016118
7	c	0.012532
8	k	0.008516
9	l	0.008162
10	m	0.003794
11	n	0.003791
12	o	0.003642
13	b	0.003342
14	p	0.003196
15	q	0.003055
16	r	0.000817
17	s	0.000673
18	e	0.000627
19	t	0.000401
20	d	0.000241
21	u	0.000897
22	v	0.000832
23	w	0.000515
24	x	0.000474

6.2 適用領域

次に，本手法の適用領域，使用方法についての考察を行う．本手法は，Android OS に対して修正を行っており，修正 OS を使用できる環境において使用することが必要となる．Android OS がサポートする端末には，エクスペリエンス端末などユーザが自由に OS をインストールすることが可能である端末が存在し，それらの端末のユーザが本提案手法を適用した Android OS を使用して本提案手法を活用する使用方法が考えられる．本稿の実験環境もこれに近い環境と考えることができる．また，これらの機器用のコンパイル済み OS バイナリ(カスタム ROM)の配布も広く行われており，それらの作成者が本提案手法を用いたバイナリの作成，配布を行い，カスタム ROM ユーザが本提案手法を使用される方法も考えられる．

端末の開発，販売を行っている端末ベンダにより本手法が用いられ，市販端末に適用される方法も考えられ，その様な場合はエクスペリエンス端末の様な OS を自由にインストール可能である端末でなくても使用可能となる．そして，端末を購入したユーザが特別な技術なく用いることが可能となる．

また，アプリケーション開発者やアプリケーションマーケットの運営者がリファレンス端末などを用いてアプリケーション評価環境を構築し，自身が開発したアプリケーションや自サイトで配布するアプリケーションの評価を行う使用法なども有効な適用領域であると期待できる．

6.3 観察時間の短縮

本手法では，アプリケーションの 24 時間分の動作の観察には，実時間で 24 時間を要することとなる．この短縮手法として OS のカーネルの時間管理部の実装を修正し，OS 内のアプリケーションが認識する時間の流れを実時間より早くする手法が考えられる[11]．この手法の適用により Alarm 起動の観察などが実時間より短い時間で行えることが確認されている．

7. 関連研究

スマートフォンや Android 端末の消費電力の調査に関して，以下の様な研究がある．Carroll らは携帯電話機の各構成要素に流れる電流を調査している[12]．彼らは各コンポーネントの電源供給線に検出抵抗器(sense resistor)を挿入して電流を計測し，コンポーネントごとの電流計測を実現している．そして，電力消費モデルを開発し，使用方法ごとの電力消費の解析を行っている．また，CPU クロック周波数の動的制御の消費電力への影響についての解析を行っている．Zhuang らは位置情報を用いるアプリケーションに着目し，位置情報処理による大きな電力消費を指摘している[13]．また，適応型位置情報取得フレームワークによる省電力削減を提案しており，彼らの手法により GPS の使用率の低減やバッテリ持続時間の拡大が可能であることを示している．

Kolin Paul らは Android 仮想マシンと通常の Java 仮想マシンの性能を比較し，消費電力に関する考察を行っている[14]．Rahul Murmuria らは Android 端末の様々なデバイス，機能による消費電力の調査を行い，ディスプレイや Wi-Fi などの機能ごとの消費電力を明らかにしている[15]．文献[16]においては，クランプメータを用いた Android スマートフォンの消費電力計測方法が示され，CPU クロック周波数と消費電力の関係が示されている．また，CPU クロック周波数をアプリケーションが要求する性能にあわせて調整し不要な電力消費を抑制する手法が提案されている．文献[17]において，電力消費が大きい装置であるディスプレイに着目し，RGB 値の制御による省電力手法が提案されている．

久保らは，ネットワーク層の情報も用いた，クロスレイヤ技術の適用により低遅延化を実現する省電力マルチホップ方式を提案しており[18]，実機において低遅延化と同時に省電力効果を示している[19]．Roy Friedman らは電力やスループットの観点から Wi-Fi や Bluetooth を評価し，これらを考慮したクロスレイヤ最適化を検討している[20]．これらの研究では，ユーザやアプリケーションによる資源要求量は既与であると仮定し，資源要求量決定後の装置による消費電力量の見積もりや削減に関する手法を行っている．しかし，資源消費の原因となるアプリケーションに関する考察は行っておらず，本研究とは目的が異なっている．

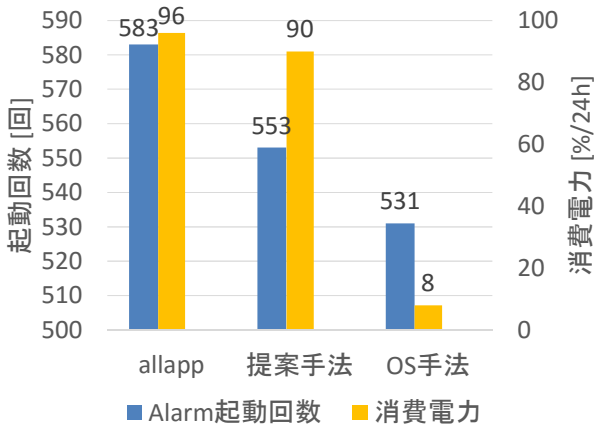

図 4　Alarm の起動回数と消費電力の関係(スポーツ)

表 6　アプリケーション毎の Alarm 起動回数ランキング(スポーツ)

順位	アプリケーション	起動回数[回]
1	a	4

表 7　OS 標準機能による消費電力の大きいアプリケーションランキング(スポーツ)

順位	アプリケーション	消費電力[%]
1	b	45
2	c	9
3	d	1
4	e	1

スマートフォンの無操作時における電力消費に関する研究としては，以下のものがある．小西らは，ブロードキャストインテントを用いた定時実行機構により大域的な同時通信が発生することを指摘している[21]. そして, バックグラウンドタスクを時間厳守型タスクと非時間厳守型タスクに分類し，非時間厳守型タスクを時間厳守型タスクと同時に実行する制御によって，端末内における複数のバックグラウンドタスクの同時実行や，端末間での実行タイミングの分散を実現し, 省電力や通信集中の回避を実現している．文献[22]は，ブロードキャストインテント発行による通信量の増加の調査や，それによる消費電力増加に関する考察が行われている．しかし，電力消費を誘発するアプリケーションに関する考察や Alarm についての考察などは行われていない．

文献[5][7]にて，WakeLock 実行や Alarm 起動予定の観察による無操作時消費電力を増加させるアプリケーションの推定手法が提案されている．しかし，Alarm の起動の観察や，様々なアプリケーション事例を用いての詳細な評価は行われていない．

8. おわりに

本稿では，無操作状態の Android 端末における消費電力に着目し，Alarm の起動回数に基づく無操作時消費電力の増加の原因となるアプリケーションの推定手法の提案を行った．また，本手法により推定したアプリケーションをアンインストールしたところ大幅な消費電力の低減が確認され，本手法に有効性があることが確認された．

今後は，Alarm による WakeUp 後の処理と無操作時消費電力の関係，観察機能による消費電力量について調査していく予定である．

謝辞 本研究は JSPS 科研費 25280022，26730040，15H02696 の助成を受けたものである．

本研究は、JST、CREST の支援を受けたものである．

参考文献

[1] 日本経済新聞 2013 年 4 月 1 日 http://www.nikkei.com/article/DGXNASFK2600W_W3A320C1000000/

[2] Smartphone OS Market Share, 2015 Q2, http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[3] PowerManager http://developer.android.com/reference/android/os/PowerManager.html

[4] 栗原 駿，福田 翔貴，小柳 文乃，小口 正人，山口 実靖，" Alarm の観察による無操作状態携帯端末の消費電力の増加の原因となるアプリケーションの推定"，データ工学と食メディア，2015-9

[5] Shun Kurihara，Shoki Fukuda，Ayano Koyanagi，Ayumu Kubota，Akihiro Nakarai，Masato Oguchi，Saneyasu Yamaguchi，“A Study on Identifying Battery-Draining Android Applications in Screen-Off State”，2015 IEEE 4th Global Conference on Consumer Electronics，2015-10

[6]AlarmManager http://developer.android.com/reference/android/app/AlarmManager.html

[7] Shun Kurihara，Shoki Fukuda，Shintaro Hamanaka，Masato Oguchi，Saneyasu Yamaguchi，“Identifying Battery-Draining Applications by Monitoring Behavior in Screen-Off State in Android”，IEEE International Conference on Consumer Electronics-Taiwan，2016-5

[8] Android アプリ作成ガイドライン https://www.nttdocomo.co.jp/binary/pdf/service/developer/smart_phone/etc/Android_app_guide_2_1.pdf

[9] Android Open Source Project http://source.android.com/

[10] 無料トップ Android https://play.google.com/store/apps/collection/topselling_free

[11] 福田翔貴，栗原駿，濱中真太郎，小口正人，山口実靖，“Android アプリケーション観察の加速環境の構築”，第 15 回コンシューマ・デバイス＆システム(CDS)研究発表会，CDS15 2016

[12] Aaron Carroll, Gernot Heiser, “An analysis of power consumption in a smartphone,” USENIXATC'10 Proceedings of the 2010 USENIX conference on USENIX annual technical conference, 21-21, 2010.

[13] Zhenyun Zhuang, Kyu-Han Kim, Jatinder Pal Singh, “Improving energy efficiency of location sensing on smartphones,” MobiSys '10 Proceedings of the 8th international conference on Mobile systems, applications, and services, 315-330, 2010.

[14] Kolin Paul, Tapas Kumar Kundu, “Android on Mobile Devices: An Energy Perspective,” 10th IEEE International Conference on Computer and Information Technology, 2010.

[15] Rahul Murmuria, Jeffrey Medsger, Angelos Stavrou, Jeffery M. Voas, “Mobile Application and Device Power Usage Measurements”, Energy aware self-adaptation in mobaile systems,USA,2013

[16] Nagata Kyosuke, Saneyasu Yamaguchi, Hisato Ogawa, "A Power Saving Method with Consideration of Performance in Android Terminals", The 9th IEEE International Conference on Autonomic and Trusted Computing (IEEE ATC 2012), ATC7-3

[17] 坂本 寛和, 中村 優太, 野村 駿, 濱中 真太郎, 山口 実靖, 小林 亜樹 “Android 端末における照度と消費電力の関係を考慮した読みやすさの低減を抑えたディスプレイ消費電力の低減”，第 12 回コンシューマ・デバイス＆システム(CDS)研究発表会, CDS12-10 2015

[18] 久保祐樹，柳原健太郎，野崎正典，“省電力リスニング MAC プロトコルの遅延と消費電力の削減手法”，電子情報通信学会技術研究報告. USN, ユビキタス・センサネットワーク 109(47), 19-24, 2009-05-15

[19] 久保祐樹，柳原健太郎，野崎正典，福永茂，中井敏久，“データ収集型無線センサネットワークに適した低遅延省電力マルチホップ通信”，電子情報通信学会論文誌. B，通信 J92-B(8), 1225-1235, 2009-08-01

[20] Roy Friedman, Alex Kogan,Yevgeny Krivolapov, “On Power and Throughput Tradeoffs of WiFi and Bluetooth in Smartphones”, Mobile Computing, IEEE Transactions on (Volume:12， Issue: 7)

[21] 小西 哲平, 稲村 浩, 川崎 仁嗣, 神山 剛, 大久保 信三, 太田 賢，“画面オフ状態におけるバックグラウンドタスク同時実行による Android 端末の省電力化”，情報処理学会論文誌, 55 巻, 2 号, pp. 587 － 597, 2014

[22] 早川 愛, 磯村 美友, 竹森 敬祐, 山口 実靖, 小口 正人, "Android 端末省電力化のためのブロードキャストインテント情報の調査", マルチメディア、分散、協調とモバイル DICOMO2014 シンポジウム, pp. 1461-1468, 2014.