スマートフォンの消費電流量推定手法の提案と評価

井原 卓也[†1] 土岐 卓[†2] 田坂 和之[†2] 大岸 智彦[†2] 小花 貞夫[†1]

近年，スマートフォンの普及が著しく進んでいるが，その電池持ちは，スマートフォンユーザの大きな関心事である．ユーザやアプリケーション開発者がアプリケーション実行時に実際どれだけの電池が消費されたかを知ることは，両者が電池持ちの課題を認識する上で，非常に重要なことであるが，電池テスタなどの専用の測定器を利用して実際に測定することは容易ではなく，また何が原因で電池を消費したのかを知ることは難しい．

そこで本稿では，専用の機材を用いずに，ソフトウェアベースで，電池消費量を推定する手法を提案，評価した．従来手法に対し，電池消費量の計算時に液晶の表示色を考慮していることを特徴とする．実験で使用した YouTube の動画の場合，液晶の表示色を考慮しない従来手法では，最大 40%の誤差率が確認されたのに対し，提案手法では，最大 16.1%以下に抑えられ，特に白・黒のみの動画の場合，誤差率を 8%以下に抑えられることを確認し，提案手法の有効性を示した．また，推定手法は Android だけでなく，近年注目され始めているモバイル OS である Firefox OS にも適用できることを確認した．

Proposal and Evaluation of Electric Current Consumption Estimation Method of Smartphones

TAKUYA IHARA[†1] SUGURU DOKI[†2] KAZUYUKI TASAKA[†2]
TOMOHIKO OGISHI[†2] SADAO OBANA[†1]

1. はじめに

近年，スマートフォンの普及が著しく進んでいる．スマートフォン選定時の重視点[1]から分かるように，スマートフォンの電池持ちは，ユーザが最も関心がある要素であり，不満を持つ要素でもある．ユーザが電池持ちに不満を感じる要因として，フィーチャーフォンより液晶画面が大きくなりその分電池消費が激しくなったこと，常駐アプリによる意図しないバックグラウンドでの通信動作により未使用状態でも絶えず電池を消費することなどが考えられる．

ユーザやアプリケーション開発者がアプリケーション実行時に実際どれだけの電池が消費されたかを知ることは，両者が電池持ちの課題を認識し，使い方および開発手法の双方の観点で解決策を図っていく上で非常に重要である．電池消費量を測定する技術として，電池テスタを利用する方法があるが，専用の機材が必要である上，誰でも簡単に使いこなせるものではない．また，電池消費量は分かったとしても，何が原因で電池を消費したのかを知ることはできない．

そこで本稿では，専用の機材を用いずに，ソフトウェアベースで，電池消費量(電流量の積算をベースとしているため，以下，消費電流量と呼ぶ)を推定する手法を提案する．従来手法に対して，消費電流計算時に液晶の表示色を考慮していることを特徴とする．また，提案手法の有効性を確認するため，YouTube での動画再生において，従来手法との比較評価を実施する．さらに，スマートフォンの OS として，Android だけでなく，近年注目され始めている Firefox OS についても同じ推定手法が適用できるのか否かを確認する．以下，提案手法の詳細ならびに従来手法をベースとした予備実験を含めた実験評価結果について述べる．

†1 電気通信大学 大学院情報理工学研究科 情報・通信工学専攻
Department of Communication Engineering and Informatics, Graduate School of Informatics and Engineering, The University of Electro-Communications
†2 株式会社 KDDI 研究所
KDDI R&D Laboratories, Inc.

2. 先行技術

従来，スマートフォンの消費電流量を推定する手法に関する取り組みがある． Google 社は，CPU や液晶などのスマートフォンのリソース(以下，リソース)毎の消費電流量の測定方法を定義している[2]. 表 1 に，リソース毎の消費電流量の一例を示す．スマートフォンの各機種に応じたリソース毎の消費電流量の一覧は，一般的にパワープロファイルと呼ばれる．消費電流量の測定は，CPU のクロック周波数，液晶の点灯/消灯，Wi-Fi アクセスポイントを経由して通信中か否かといったリソースの状態毎に行う．また，Lide Zhang らの論文[3]では，スマートフォンのリソース状態毎の消費電流量を独自の手法で実際に測定し，その値を利用してスマートフォンの消費電流量を推定する方式の検討を行っている．また Android でアプリケーションを作成し，推定した消費電流量と電流計で測定した実際の消費電流量との差異を見ることによる精度評価を行っている．ここでは，リソース状態毎の消費電流量を利用した推定方法を示す．

表 1 Nexus S のパワープロファイル

測定項目		内容	Android (mA)
（リソース）	（状態）		
CPU	cpu.idle	CPU アイドル状態	1.4
	cpu.active1	CPU クロック周波数 (200 MHz)	82.1
	cpu.active2	〃 (400 MHz)	113.7
	cpu.active3	〃 (800 MHz)	205.4
	cpu.active4	〃 (1000 MHz)	259.0
液晶	screen.on	液晶(輝度最小時)	49.0
	screen.full	液晶(輝度最大時)	260.0
GPS	gps.on	GPS オン	50.0
Wi-Fi	wifi.scan	Wifi スキャン	220.0
	wifi.on	Wifi 接続(データ未通信)	4.0
	wifi.active	Wifi 通信(データ通信時)	120.0

図 1 消費電流係数の例

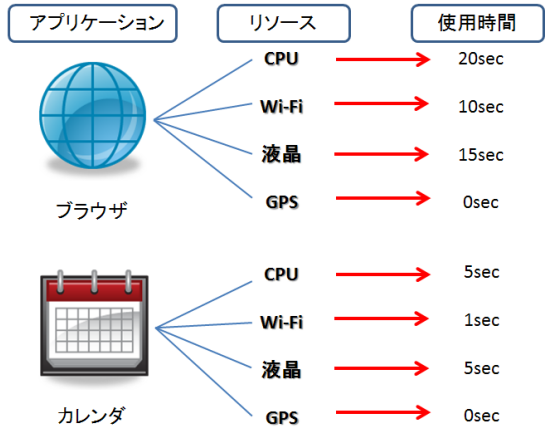

図 2 アプリケーション実行時のリソース使用状況例

【消費電流量の推定方法】

リソース状態毎の毎秒の消費電流量(これを消費電流係数と呼び，単位は mA とする)と，リソースの状態毎の使用時間から実測せずにスマートフォンの消費電流量を推定する．この時，スマートフォン全体の消費電流量は以下の式(1)で定義する．

$$I_{total} = \Sigma I_j \cdot T_j \quad \cdots (1)$$

I_j：リソース状態 j の消費電流係数

T_j：リソース状態 j の使用時間

I_{total}：スマートフォン全体の消費電流量

図 1 と図 2 を用いて消費電流モデルの簡単な例を示す．例では，Wi-Fi，液晶，CPU が使用されていた場合，スマートフォン全体の消費電流量は，726 mA(166 + 260 + 300 + 0)と推定することができる．このため，予めリソース毎の消費電流係数を実測しておき，図 2 のようなアプリケーション実行時のリソース状態毎の使用時間を測定することで，当該アプリケーションを実行したときに消費するスマートフォン全体の消費電流量を推定できる．

3. 予備実験

3.1 予備実験 I

まず，Nexus S が持つパワープロファイル（表 1）を用いた消費電流量の推定精度，誤差率(実測値を基準とした際の，推定値との差の比率と定義する)を評価した．このパワープロファイルベースの推定方式を，従来方式 I とする．

3.1.1 実験方法

① 消費電流量の測定環境

図3に消費電流量の測定機器の接続図を示す． テスタにはキーエンス社 NR-2000 を，また，安定化電源には，アルインコ社 DM-310MV をそれぞれ使用した．抵抗は 50mΩ のものを使用した．NR-2000 での測定により取得した消費電流量をパソコンに取り込むロガーとして，WAVESHOT!2000 ソフトウェアを用いた．安定化電源から 3.7V の電圧をかけ，測定を行い，ロガーで消費電流量を収集した．消費電流量は 1 秒に 10 回の頻度で測定できるように収集条件を設定した.また，試験機種として，Android(ver 4.0.4)のスマートフォン Nexus S を用いた．

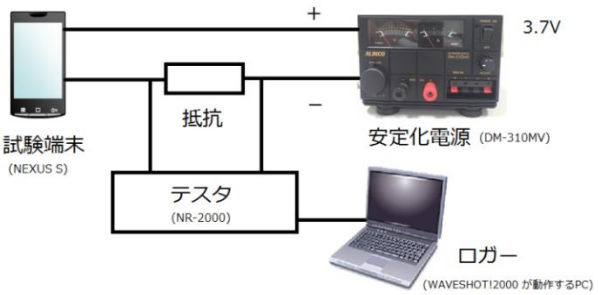

図 3 消費電流量の測定機器の接続図

② 測定対象のアプリケーション

測定対象のアプリケーションを YouTube[4]とし，動画再生時の消費電流量を測定した．測定に用いた動画は以下の 2 つとした．なお，画質は HD，音はミュートとし，最大画面表示として再生した．

動画 A：Blank Screens の WhiteScreen(16:9)
(http://www.youtube.com/watch?v=Jr3tlqXH7is)
白画面を表示し続ける動画の 60 秒間(0:30~1:30)

動画 B：firefoxchannel の Firefox OS The Jorney
(http://www.youtube.com/watch?v=LizAjaOXi4U)
画面が移り変わる暗めな動画 60 秒間(0:30~1:30)

③　推定値の計算方法

動画再生時に稼働するリソースを，CPU，液晶，Wi-Fi の 3 つと仮定した．CPU クロック周波数は，動画再生時の CPU クロック周波数の値(/sys/devices/system/cpu/cpu0/cpufreq/scaling_cpu_freq)を測定し，その値を使用した．また，Wi-Fi については，通信量の測定結果より，常に wifi.active(データ通信状態)として計算した．液晶については，パワープロファイルには中間の輝度に関する項目がないため，最小輝度を意味する項目（screen.on）を用いた場合と最大輝度を意味する項目(screen.full)を用いた場合の 2 通りについて計算した．

3.1.2 実験結果と考察

予備実験 I による消費電流量測定結果を図 4 に示す．動画 A の誤差率は，31%~116%程度，動画 B の誤差率は 87%~217%程度となり，どちらの場合も推定精度が非常に悪い結果となった．これより，デフォルトのパワープロファイルの値はあまり正確でなく，従来手法 I では，アプリケーション実行時の消費電流量の推定に問題が生じることが分かった．

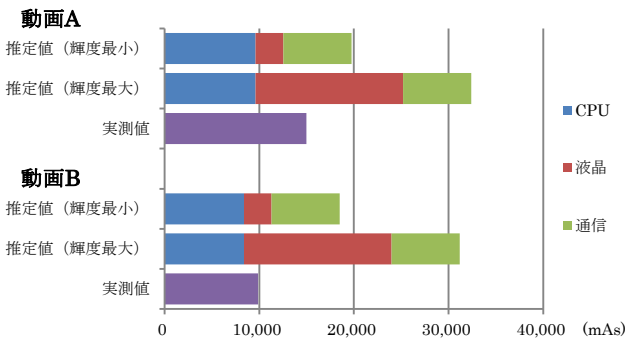

図 4 従来方式 I を用いた推定値と実測値の比較

3.2 予備実験 II

次に，Google 社が定義する測定方法[2]と先行研究[3]を参考に，リソース状態毎の消費電流量を実測し，その結果を用いて推定値を計算した場合の推定精度の評価を行った．この推定方式を，従来方式 II とする．なお，消費電流量の測定方法や，測定対象のアプリケーションは，予備実験 I と同じとした．

3.2.1 消費電流係数の測定

先行研究[3]では，Android のリソースとして，CPU，Wi-Fi，液晶，GPS，3G 回線，Audio などを対象としているが，本稿では，実験の目的や推定値の計算に影響すると考えられる CPU，Wi-Fi，液晶，GPS の 4 つを対象とした．消費電流係数を測定するリソース状態を表 2 に示す．それぞれのリソース状態の消費電流係数は以下の(1)~(4)に示す通りに 60 秒間測定し，その結果を 60 で割った値とした．測定はそれぞれ 10 回行った．

表 2 測定項目の内容

測定項目		内容
（リソース）	（状態）	
CPU	cpu.idle	CPU アイドル状態
	cpu.active1	CPU クロック周波数 (200 MHz)
	cpu.active2	〃 (400 MHz)
	cpu.active3	〃 (800 MHz)
	cpu.active4	〃 (1000 MHz)
液晶	screen.on	液晶(輝度 150cd/m^2)
GPS	gps.on	GPS オン
Wi-Fi	wifi.scan	Wifi スキャン
	wifi.on	Wifi 接続(データ未通信)
	wifi.active	Wifi 通信(データ通信時)

(1) CPU

スマートフォンを機内モードとし，タスクの実行中のアプリケーションを全て終了させ，画面をオフにした際の状態を CPU アイドル状態（cpu.idle）として定義した．この状態の CPU クロック周波数を 60 秒間, 10 回測定し, CPU クロック周波数が最小の値であることを確認した．これより，この結果を基準として，他のリソース状態の消費電流係数をこの値との差分として計測した．

CPU クロック周波数の測定は, PC と USB 接続し, Android Debug Brige(adb)[5]を用いて，adb で以下の①～④などのパスの値を取得，変更して測定した．なお，今回 CPU に高負荷をかけるプログラムは，三角関数の計算を 60 秒間以上無限に繰り返すプログラムを使用した．測定した CPU クロック周波数の値は，表 2 に示す通りである．

① /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq の現在最大に設定されている CPU クロック周波数の値を任意のクロック周波数に変更
② /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor の値を performance に変更
③ /sys/devices/system/cpu/cpu0/cpufreq/scaling_cpu_freq を確認し，現在のクロック周波数が測定項目の値となっていることを確認
④ CPU に高負荷をかけるプログラムを実行し，測定

(2) 液晶

液晶の輝度に関しては，パワープロファイル[2]では考慮されていないが，論文[3]では考慮されている．そのため，予備実験では, Android で白画面を表示した際の消費電流量が約 150cd/m^2 となる際の内部設定輝度(WindowManager の screenBrightness とする)を外部から輝度計を用いて測定した．輝度計には，トプコンテクノハウス社 BM-9M を用いた．測定の結果，Android の内部設定輝度が 0.76 の際に約 150cd/m^2 となることが分かった．消費電流係数は，この内部設定輝度で白画面を表示した際の値(screen.on)とした．

(3) GPS

Android の位置情報を取得するクラスである Location Manager を用いて，現在地を取得するプログラムを作成した．設定アプリで GPS の機能を許可し，実際に GPS のリソースが稼働している際の消費電流量（gps.on）を測定した．

(4) Wi-Fi

Wi-Fiに関しては，通信状態毎に以下の3つのリソース状態に分類し，測定を行った．

① wifi.scan：アクセスポイントをスキャンする状態
② wifi.on：アクセスポイントに接続しているが，データを通信していない状態
③ wifi.active：アクセスポイントに接続し，データを通信している状態

また，それぞれのリソース状態について以下の様にして測定した．

a) wifi.scan では，5 秒に一度の頻度でアクセスポイントをスキャンするプログラムを作成し，測定した．
b) wifi.on では，アクセスポイントに接続し，データを通信していない状態で測定した．
c) wifi.active では,特定のサーバに毎秒 8 パケットずつデータを送信するプログラムを作成，測定した．

3.3 予備実験Ⅱの結果と考察

予備実験による消費電流係数の測定結果を表 3 に示す．表 3 と表 1 より，今回実測した消費電流係数の結果と，デフォルトのパワープロファイルの値に大きな違いがあることが分かる．また，表 3 の結果を用いた推定値の計算結果と，実測値の比較を図 5 に示す．動画 A の誤差率は 1.8%，動画 B の誤差率は 40%であった．これより，予備実験 I の結果よりも推定精度が上がっていることが分かる．また，動画 A と動画 B の比較を行うと，動画 A の推定精度は高いが， 動画 B の推定精度が悪い．これは，予備実験Ⅱの推定値の計算方法では，液晶の表示色を考慮していないため，実際に画面が移り変わるような動画 B では，推定精度が悪くなったと考察できる．これより，液晶画面の色が異なる場合では，推定精度が悪くなるため，消費電流係数に色に関するリソース状態を追加することが推定精度向上のために必要であると考えられる．

4. 改善手法の提案

4.1 提案手法の概要

2 つの予備実験より，スマートフォンのアプリケーション実行時の消費電流量推定には，リソース状態毎の消費電流量の実測と，液晶での色や輝度の考慮が必要であることが分かった．以下にこれらに対する，スマートフォンの消費電流量推定方式の改善手法を提案する．

表 3 Android の消費電流係数の測定結果(mA)

測定項目		消費電流係数
（リソース）	（状態）	（測定データ）
CPU	cpu.idle	1.02
	cpu.active1	32.44
	cpu.active2	52.47
	cpu.active3	96.87
	cpu.active4	111.87
液晶	screen.on	152.32
GPS	gps.on	19.68
Wi-Fi	wifi.scan	20.15
	wifi.on	10.24
	wifi.active	20.08

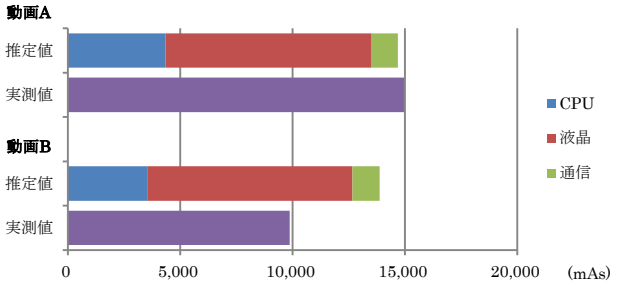

図 5 従来方式Ⅱを用いた推定値と実測値の比較

4.2 液晶の消費電流係数の導入

複数の色を表示した際の消費電流量の測定を行い，色を考慮した液晶の消費電流係数を求めた．測定方法は 3.2.2(2)と内部設定輝度以外の設定を同じとし，表示色と内部設定輝度を変化させて測定した．表示色は，2 色(White, Black)とし，輝度は両 OS の内部設定輝度の値を 5 通り(0.25, 0.34, 0.50, 0.76, 1.0)に変化させて測定した．測定結果を図 6 に示す．測定結果より，色と輝度を考慮した消費電流係数を以下の式(2)のように定義する．Px, Cx の値は図 6 の結果より，以下の表 4 の値となる．

$$I_{screen\_x} = P_x \bullet B + C_x \qquad (2)$$

I_{screen_x} : x 色を表示した際の液晶の消費電流係数
B : 内部設定輝度 (screenBrightness)
P_x : x 色を表示した際の傾き
C_x : x 色を表示した際の切片

これらより，本稿では，式(2)と表 4 より定義される，表示色と輝度を考慮した液晶の消費電流係数を用いた推定手法の提案を行う．

表 4 表示色毎の Px, Cx

	Android (mA)	
	P_x	C_x
White	140.8	50.71
Black	0.371	58.77

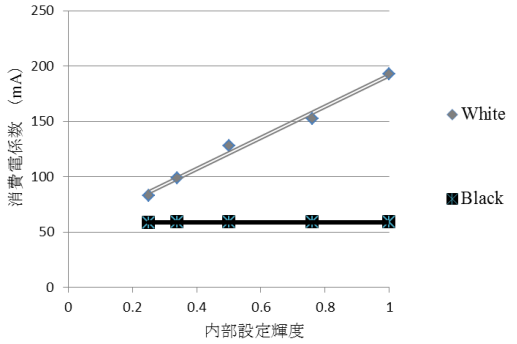

図 6 Android の様々な輝度や色毎の消費電流係数

5. 実験

4 章で提案した消費電流量の推定方式の有効性を示すため，動画再生時の消費電流量の推定と実測をそれぞれ行った．また，リソース状態取得頻度に関して考察するため，様々な取得頻度での評価も行った．さらに，推定手法が従来研究で対象である Android だけでなく，新たに注目されている OS である，Firefox OS についても適用できるかどうか調査し，両 OS の比較を行った．

5.1 提案手法の有効性を評価するための実験

5.1.1 測定方法

以下に示す 4 つの YouTube の動画について，動画再生時の消費電流量の推定値の計算と実測を行った．なお，測定条件は，3.1.1 節と同じとした．

動画①：Blank Screens の WhiteScreen(16:9)
(http://www.youtube.com/watch?v=Jr3tlqXH7is)
白画面を表示し続ける動画の 60 秒間(0:30~1:30)

動画②: PrudenceNumberNine の Ten Minutes of Black Screen
(http://www.youtube.com/watch?v=zwOGdnP-Kw0)
黒画面を表示し続ける動画の 60 秒間(0:30~1:30)

動画③: Google Developers の YouTube API Overview
(http://www.youtube.com/watch?v=ylLzyHk54Z0)
白基調の明るめな動画の 60 秒間(0:30~1:30)

動画④: firefoxchannel の Firefox OS The Jorney
(http://www.youtube.com/watch?v=LizAjaOXi4U)
暗めな動画の 60 秒間(0:30~1:30)

5.1.2 推定値の計算

推定値の計算は，液晶以外の部分は従来手法 II と同じ手法で計算した．液晶に関しては，4 節提案手法の式(2)と表 4 に従い，動画①，③では，White を表示した際の液晶の消費電流係数，動画②，④では，Black を表示した際の液晶の消費電流係数を用いて計算した．

5.1.3 測定結果

4 つの動画の推定値と実測値の比較結果を，図 7 に示す．動画①～④の誤差率はそれぞれ 1.8%，12%，4.0%，16.1%であり，従来方式 I 及び II よりも推定精度が高い結果となった．また，白画面のみを表示する動画①の推定値は，誤差率が 1.8%と最小であり，非常に精度が高い結果となった．

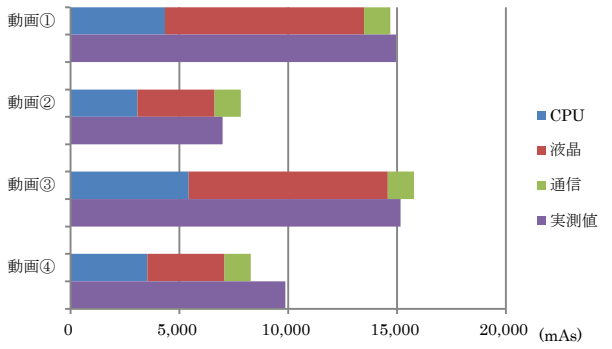

図 7 動画再生時の推定値と実測値の比較結果

5.2 リソース状態取得頻度に関する評価

実際に推定値を計測するアプリケーションを作成する場合，リソース状態を高頻度に収集することは，リソース状態収集機能自体のオーバヘッドを生じる要因となるため，適当な頻度でリソース状態を取得することが求められる．5.1 の実験で測定した結果のうち，最も誤差率が低かった動画①について，CPU クロック周波数の測定頻度を，毎秒，10 秒毎，毎分とした場合の推定値と実測値の比較を行った．その結果を図 8 に示す．測定結果より，毎秒，10 秒毎，毎分の誤差率はそれぞれ 1.8%, 2.9%, 14%となり，測定頻度が粗くなるにつれ誤差率が大きくなる傾向となった．

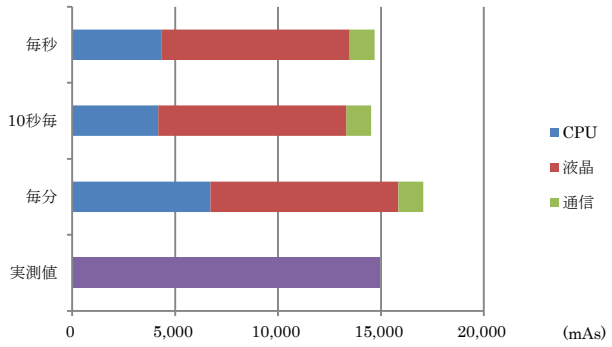

図 8 リソース状態取得頻度による推定結果の比較

5.3 Firefox OS での実験

5.3.1 消費電流係数の測定

① 測定方法

推定手法が Firefox OS でも適用できるかどうかを調査するために，まずその消費電流係数を実測した[6]．測定方法は 3.1.1 節と同じ測定回路とし，試験端末は Firefox OS (ver Boot2Gecko1.1.0.0hd-prerelease)をビルドした Nexus S を用いた．また，Android の消費電流係数の測定に使用したプログラムと同じ機能を持つ Firefox OS の API を調査[7][8]し，それらを用いて Firefox OS の消費電流係数を測定した．以下の表 5 は，それらの対応表である．

表 5 Firefox OS の測定用プログラムの機能と API

機能	API
内部設定輝度	Power Management API navigator.mozPower.screenBrightness
Wi-Fi ON/OFF	Setting API ‘wifi.enabled’
Wi-Fi 通信	TCP Socket API
GPS ON/OFF	SettingAPI ‘geolocation.enabled’ + Geolocation API
CPU クロック周波数	Android と同じ手法(adb 接続)

② 測定結果

以下の表 6 に, Firefox OS の消費電流係数の測定結果を示す. なお, 測定は 3 節と同様, 10 回測定の平均値とした. 表 6 より, CPU クロック周波数が増えると, 両 OS で消費電流量が増大することが分かった. また, Firefox OS と Android の結果を比較すると, CPU クロック周波数が高い状態で負荷をかけた場合の消費電流量は Android が 20mA 程度 Firefox OS より少ないが, CPU クロック周波数が低い状態や, GPS, Wi-Fi に関連する設定項目については, 消費電流量に殆ど差が無かった.

以上, Firefox OS と Android のリソース毎の消費電流係数は, 一部を除き大差が無い結果となった.

表 6 Firefox OS と Android の消費電流係数測定結果

測定項目		消費電流係数(mA)	
（リソース）	（状態）	**Firefox OS**	**Android**
CPU	cpu.idle	1.30	1.02
	cpu.active1	26.96	32.44
	cpu.active2	52.34	52.47
	cpu.active3	115.66	96.87
	cpu.active4	130.03	111.87
液晶	screen.on	150.84	152.32
GPS	gps.on	20.61	19.68
Wi-Fi	wifi.scan	23.67	20.15
	wifi.on	3.57	10.24
	wifi.active	24.35	20.08

5.3.2 Web ブラウザ各操作時の測定

両 OS の Web ブラウザ操作時の消費電流量の違いを考察するため, Web ブラウザでの 4 つの操作, すなわち, ①起動時, ②無操作時(blank ページ表示), ③ブラウジング操作時（検索・閲覧）, ④ページ遷移時（通信・レンダリング）におけるそれぞれの消費電流量を測定した. Android の Web ブラウザには, デフォルトのブラウザ（以下, Android default と表記）と Firefox アプリ[9](以下, Android firefox と表記)を使用した. Firefox OS の Web ブラウザには, デフォルトのブラウザ(firefox ブラウザ)を使用した. また, Web サイトは Firefox OS の設定に合わせ, PC サイト表示に統一して表示させ, 測定した. 測定はそれぞれ 10 回行った.

① 起動時の消費電流量

Web ブラウザのアプリケーション起動時にかかる消費電流量を測定した. アプリケーションのアイコンを押した際のクロック周波数を実測したところ, クロック周波数が 2 秒間最大になることから, ここではアイコンを押してから 2 秒間の消費電流量を起動時の消費電流量として測定した.

② 無操作時の消費電流量（blank ページ表示）

それぞれのブラウザで about:blank のページ（空白のページ）を表示させた際の消費電流量を測定した. ページを表示させてから起動にかかる消費電流量と区別するため, 10 秒後に測定を開始し, 30 秒間測定した.

③ ブラウジング操作時の消費電流量（検索・閲覧）

それぞれのブラウザで測定毎にキャッシュや閲覧履歴を削除し, blank ページからキーワードの検索を開始した瞬間から測定を始め, 目的のサイトへアクセスし, スクロールして閲覧するまでの消費電流量を実測した. ここでは, 以下の 2 つの Web サイトに対して同じ条件（時間と操作）で測定した.

ブラウジング 1: ページ容量の小さいサイト. 電気通信大学の英語版ホームページ (http://www.uec.ac.jp/eng/)

ブラウジング 2 : 画像などが多くページ容量の大きいサイト. 価格.com のホームページ (http://kakaku.com/)

図 9 は, Firefox OS で「uec」と検索し, 英語版の電気通信大学のホームページを閲覧した際(ブラウジング 1)の表示画面の例を示す.

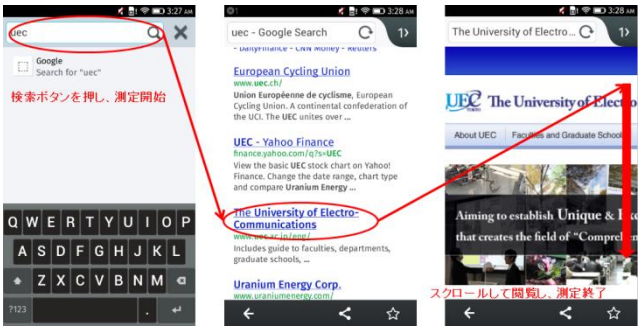

図 9 ブラウジング操作の例

④ ページ遷移時の消費電流量（通信とレンダリング）

それぞれのブラウザで測定毎にキャッシュや閲覧履歴を削除し, Web サイト表示画面から測定を開始して, キャッシュされていない別ページへの遷移を 30 秒間繰り返した際の消費電流量を実測した. 同じ条件で以下の 3 つの Web サイトに対して測定を行った.

ページ遷移 1 : ページ容量の小さいサイト. 電気通信大学の英語版ホームページ(http://www.uec.ac.jp/eng/)

ページ遷移 2 : 画像などが多くページ容量の大きいサイト. 価格.com のホームページ(http://kakaku.com/)

ページ遷移 3：テキストと画像が多くページ容量の大きいサイト. Yahoo! ニュースのホームページ (http://news.yahoo.co.jp/)

測定結果を表 7 に示す．なお，表内のデータの単位は，mA であり，10 回測定の平均値である．

表 7 より，Web アクセスでは，全ての場合において Firefox OS の消費電流量が Android の消費電流量より少ないことが分かる．特に，ブラウジング操作時の差が大きい．また，起動では，Firefox OS と Android のデフォルトのブラウザであまり大きな差は無かった．全体的に結果を見ると，どのブラウザでも，無操作時（blank ページ表示）の消費電流量が最も少ない．また，起動以外では，Android のデフォルトのブラウザと，Firefox アプリの間で大差は無い．

表 7　Web ブラウザ各操作時の毎秒の消費電流量 (mA)

	Firefox OS	Android default	Android firefox
起動	237.90	238.16	278.64
blank page	151.19	181.62	191.78
ブラウジング 1	225.14	323.29	310.32
ブラウジング 2	257.35	332.50	344.69
ページ遷移 1	301.62	320.15	328.08
ページ遷移 2	304.34	324.19	311.76
ページ遷移 3	306.26	330.48	334.46

5.3.3 推定値と実測値の比較

Web ブラウザ操作時のブラウジング 1 とページ遷移 1 について，以下のようにして推定値を計算する．なお，両 OS の Web ブラウザは，デフォルトのブラウザとした．

消費電流量の推定値の計算に用いるリソースは，ブラウザ上の操作であるため，CPU，Wi-Fi，液晶の 3 つとする．CPU と Wi-Fi に関しては，それぞれの比較対象の操作について，それぞれのリソース状態の滞在時間の測定を行った．なお，液晶については，ブラウザ画面のレイアウトなども考慮し，それぞれのブラウザで about：blank のページを表示した際の毎秒の消費電流量を推定値の計算に使用した．

ブラウジング 1 では，CPU クロック周波数と Wi-Fi の通信時間は変動していたが，ページ遷移 1 では，CPU クロック周波数が常に最大値（1,000 MHz）となり，Wi-Fi は常に通信状態であった．図 10 と図 11 に，推定値と実測値の比較結果を示す．ブラウジング 1 では，Firefox OS の推定値の誤差率が 10%であり，一方で Android の誤差率は 8%であった．また，ページ遷移 1 では，Firefox OS の誤差率は 1%未満と非常に低く，Android でも誤差率は 3.5%であった．

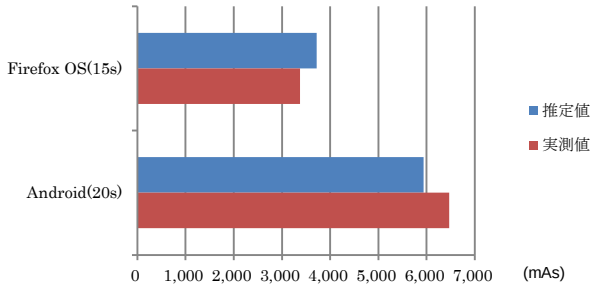

図 10　ブラウジング 1 の推定値と実測値の比較結果

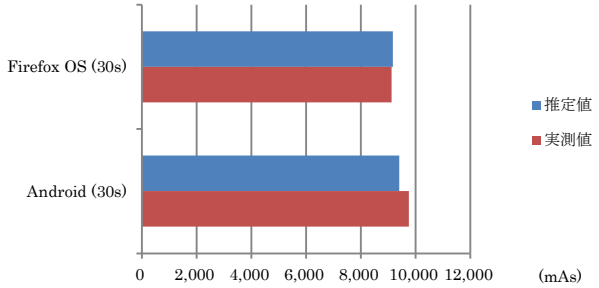

図 11 ページ遷移 1 の推定値と実測値の比較結果

6. 考察

6.1 推定手法の有効性に関する考察

5.1 節，図 7 の結果より，提案する推定手法の有効性を示すことができたと考えられる．また，5.3 節の結果より，推定手法は Firefox OS においても同等に適用できることが確認できた．さらに，それぞれの実験結果などから以下のことが考察できる．

(1) パワープロファイルに関して

予備実験 I より，デフォルトのパワープロファイルの値を用いて推定値の計算をした場合，推定精度が非常に悪くなり，実際にリソース状態毎の消費電流量を実測することが必要であると分かった．

(2) リソース状態取得頻度に関して

5.2 節の図 8 の結果より，リソース状態取得頻度は，高頻度であればあるほど，推定精度が上がることが分かる．ただし，毎秒の誤差率が 1.8%, 10 秒毎の誤差率が 2.9%程度であり，推定精度はそれほど大きく変わらない結果から，ユーザやアプリケーション開発者が求める推定精度によってリソース取得頻度を変更する必要がある．最適なリソース状態取得頻度に関しては，今後更なる調査が必要である．

(3) 表示色の測定結果に関して

動画再生時の結果が図 7 の結果のようになったのは，動画③や動画④は，実際は白画面または黒画面のみを表示する動画ではないので，動画③では推定値の方が大きくなり，動画④では推定値の方が小さくなったと考えられる．また動画②では，推定値の計算時，上部ステータスバー表示時に測定した黒画面の消費電流係数としていたが，実際の動

画再生時は，上部ステータスバーが非表示であったため，誤差が生じたと考えられる．なお，全画面黒表示にして消費電流係数を再測定し，推定値を再計算した結果，誤差率が 12%から 8%程度まで減少した．従って，高精度な推定値を必要とする場合は，液晶表示画面の消費電流係数を厳密に実際のアプリケーションで表示される画面と同じ条件で測定する必要がある．

(4) 表示色のリソース状態取得方法に関して

液晶の表示色に関するリソース状態取得方法については，今後の課題である．外部からのカメラを用いた撮影や，内部からのスクリーンショットによる撮影から，画像解析によって表示色を取得する方法や，Web サイトであれば HTML のタグから背景色を取得する方法などが例として挙げられるが，実際どれほどの精度でこれらを取得できるのかどうかについては，今後調査が必要である．

(5) 他リソースに関して

今回は，リソースとして，CPU，液晶，Wi-Fi, GPS の 4 つを対象としたが，スマートフォンには他にも Audio や GPU，Bluetooth，加速度センサ，ジャイロセンサなどといったリソースが存在する．これらのリソースを含めた消費電流モデルの作成や，消費電流量への影響調査については，今後の課題とする．

6.2 Android と Firefox OS に関する考察

(1) 消費電流係数の測定結果について

リソース毎の消費電流係数の測定結果が両 OS で一部を除き大きな差がなかった原因として，Firefox OS のデバイスドライバ群が，Android と同じものを利用している[10]ためと考えられる．しかし，CPU に負荷をかけた際には両 OS で差が出た．この原因には，Android はネイティブアプリ向けの実行環境（Dalvic VM）を持つのに対して，Firefox OS にはそれが無いことや，また，Firefox OS が未だ開発段階の OS であり，CPU が高負荷な状態での処理時間が多くなっているのではない等が考えられる．今後の Firefox OS のバージョンアップで改善されることが期待される．

(2) Web ブラウザ各操作時の測定結果について

Web ブラウザでの各操作時の測定結果が，Android より Firefox OS の方が少ない消費電流量であった原因として，Android ではブラウザや Firefox アプリを動作させた上で WebKit などの Web 系ランタイムを動かす必要があり，複数の実行環境を必要とするのに対して，Firefox OS では実行環境が Web 系ランタイムの Gecko のみであり，Web アプリを実行する環境が Android よりもシンプルであるためと考えられる． 今回測定した各操作に関して，特にブラウジング操作時の差が顕著であったが，この原因が，a)通信やレンダリングによるものか，b)ブラウザアプリ起動時にかかる負荷によるものか，あるいは，c)スクロールなどを含む画面制御によるものか等について，今後調査が必要である．

7. おわりに

本稿では，電池テスタなどの専用の測定器を使わずに，ソフトウェアベースでスマートフォンの消費電流量を推定する手法を提案，評価した．本手法は，液晶の表示色を考慮していることを特徴とする．実験で使用した YouTube の動画の場合，液晶の表示色を考慮しない従来手法では，最大 40%の誤差率が確認されたのに対し，提案手法では，最大 16.1%以下に抑えられ，特に白・黒のみの動画の場合，誤差率を 8%以下に抑えられることを確認し，提案手法の有効性を示した．また，白に近い動画，黒に近い動画がそれぞれ白，黒の消費電流係数を用いてマッピングした推定値に近かったことから，一般の動画の場合には，白・黒の消費電流係数を用いて推定値を算出できると推測できる．さらに，推定手法は Android だけでなく，Firefox OS にも適用できることが分かった．

今後は，推定精度をさらに向上させる方法の検討と実証を行っていく予定である．白・黒だけでなく 3 原色の消費電流係数の測定と推定精度の評価，GPU，Audio などといった他リソース毎の消費電流係数の測定と推定精度の評価などが課題として挙げられる．また，Firefox OS と Android において推定値に誤差が生じた要因に関しても今後更なる調査を進める予定である．

参考文献

[1] スマートフォンに関するアンケート調査，マイボイスコム（株）,2013, http://www.news2u.net/releases/108837/items/1/

[2] Power Profiles for Android | Android Developers, Google, 2013, http://source.android.com/devices/tech/power.html

[3] Lide Zhang, et al.,“Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones,” CODES/ISSS '10 Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, pp.105-114 (2010)

[4] YouTube, Google Inc. ,2014, https://play.google.com/store/apps/details?hl=ja&id=com.google.android.youtube

[5] Android Debug Bridge | Android Developers, Google, 2013, http://developer.android.com/tools/help/adb.html

[6] 井原卓也 田坂和之 大岸智彦 小花貞夫, “スマートフォンの Firefox OS と Android の消費電流量に関する一考察”, 情報処理学会第 76 回全国大会,4W-3, 2014

[7] Firefox OS - Mozilla | MDN, Mozilla, 2013, https://developer.mozilla.org/en-US/Firefox_OS

[8] WebAPI – MozillaWiki, Mozilla, 2013, https://wiki.mozilla.org/WebAPI

[9] Firefox モバイルブラウザ, Mozilla, 2013, https://play.google.com/store/apps/details?hl=ja&id=org.mozilla.firefox

[10] 本間雅史 他, Firefox OS アプリ開発ガイド,リックテレコム (2013)