Future Computing and Informatics Journal

Volume 2
Issue 1 *2017, Volume 2, Issue 1*

Article 6

2017

Enersave API: Android-based power-saving framework for mobile devices

Y Beeharry
Department of Computer Science and Engineering, University of Mauritius, Réduit, Mauritius,
y.beeharry@uom.ac.mu

A.M Muharum
Department of Computer Science and Engineering, University of Mauritius, Réduit, Mauritius,
mohammad.muharum@umail.uom.ac.mu

V Hurbungs
Department of Computer Science and Engineering, University of Mauritius, Réduit, Mauritius,
v.hurbungs@uom.ac.mu

vershley joyejob
Department of Computer Science and Engineering, University of Mauritius, Réduit, Mauritius,
vershley.joyejob@umail.uom.ac.mu

Available online at www.sciencedirect.com

ScienceDirect

Future Computing and Informatics Journal 2 (2017) 48–64
http://www.journals.elsevier.com/future-computing-and-informatics-journal/

Enersave API: Android-based power-saving framework for mobile devices

A.M. Muharum, V.T. Joyejob, V. Hurbungs, Y. Beeharry*

Department of Computer Science and Engineering, University of Mauritius, Réduit, Mauritius

Received 9 September 2016; revised 15 May 2017; accepted 2 July 2017
Available online 29 July 2017

Abstract

Power consumption is a major factor to be taken into consideration when using mobile devices in the IoT field. Good Power management requires proper understanding of the way in which it is being consumed by the end-devices. This paper is a continuation of the work in Ref. [1] and proposes an energy saving API for the Android Operating System in order to help developers turn their applications into energy-aware ones. The main features heavily used for building smart applications, greatly impact battery life of Android devices and which have been taken into consideration are: Screen brightness, Colour scheme, CPU frequency, 2G/3G network, Maps, Low power localisation, Bluetooth and Wi-Fi. The assessment of the power-saving API has been performed on real Android devices and also compared to the most powerful power-saving applications – DU Battery Saver and Battery Saver 2016 – currently available on the Android market. Comparisons demonstrate that the Enersave API has a significant impact on power saving when incorporated in android applications. While DU Battery Saver and Battery Saver 2016 help saving 22.2% and 40.5% of the battery power respectively, the incorporation of the Enersave API in android applications can help save 84.6% of battery power.

Keywords: Mobile; Android; Energy consumption; API

1. Introduction

Mobile devices have become an integral part of people's lives [1,2]. The reason for them being given the name of smart devices is essentially due to the fact that they are used to perform several important tasks. Along with the unprecedented evolution of mobile devices, comes those applications which magnify their services but at the same time increase the average power consumption leading to shorter and less anticipated battery life [3]. Thousands of applications are developed on a daily basis with continuous improvements and novel features but very often, the power consumption factor is overlooked. It is therefore necessary to study the usage pattern for proposing better power saving solutions.

The current most popular operating systems are Google's Android, Microsoft's Windows Mobile and Apple's IOS. Fig. 1 shows an indication of the market share of these operating systems as at 2015 [4]. Clearly, Android has the highest market share of approximately 83%. Android is an open source mobile operating system developed by Google [5]. The Android marketplace continues to grow at an aggressive rate, and a great number of powerful applications are uploaded on a daily basis. According to Statista, there are more than 1.5 million applications on the Android Market [6].

Android has not been spared against the difficulties involving the battery life of its devices. Despite being equipped with various and powerful functionalities, the battery life of the mobile devices is very often neglected. Prolonged use of those features limits the battery life to only a few hours. The authors of Ref. [7] argued that in spite of many improvements in low-power hardware design and battery life, there is now growing

* Corresponding author.
E-mail addresses: mohammad.muharum@umail.uom.ac.mu (A.M. Muharum), vershley.joyejob@umail.uom.ac.mu (V.T. Joyejob), v.hurbungs@uom.ac.mu (V. Hurbungs), y.beeharry@uom.ac.mu (Y. Beeharry).

Peer review under responsibility of Faculty of Computers and Information Technology, Future University in Egypt.

http://dx.doi.org/10.1016/j.fcij.2017.07.001

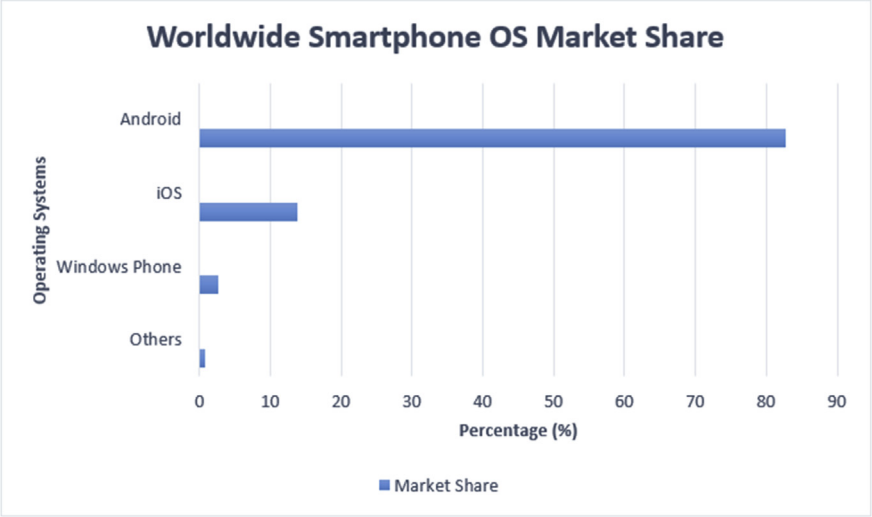

Fig. 1. Android OS market share [4].

awareness that a strategically viable approach to energy management must include higher levels of the system. The authors of Ref. [8] stated that energy should be considered as a first-class resource together with the traditional OS perspective of performance maximization. Despite of the works produced, most of them gather and model data using benchmark statistics without taking much consideration of the users' behaviour on the device. A study along this line has been conducted in Ref. [9]. The first attempt to building a personalized power model in this direction was performed in Ref. [10].

In order to cope with the power-hungry mobile phones, researchers are actively looking for ways to improve not only energy density but also longevity, self-discharge and operational costs. The two most widely used battery types are the Lithium-Ion (Li-ion) and Lithium-Polymer (Li-Po) batteries. The use of Li-ion battery system in mobile phones has taken an upward trend during the recent years. The fact that this battery provides high-energy density and is light in terms of weight makes it very suitable for mobile phones. The power density of the battery is 126 Wh/kg [11]. Renowned companies like Samsung, LG and Apple make use of Li-ion batteries to power their mobile phones. The average charging time is 2 h [11]. The power density of Li-Po batteries is 185 Wh/kg [12], which is more than Li-ion batteries. Apple's iPhone 5 which is fuelled by this type of battery takes approximately slightly more than 2 h to fully charge according to [13]. The battery of the Apple's iPhone 5 can last up to 10 h when making voice call over 3G network [14].

The power consumption may be broken into the following main subsystems which have been collected by measurements from the operating system derived from Refs. [15,16]. These components consume more power than other ones in mobile devices.

Power
used = display + CPU + audio + GPS + Bluetooth + Wi-Fi.

During normal operation of a mobile phone, it is the display and the GSM network which utilize the most power [17]. Table 1 shows the screen power consumption of the Samsung Galaxy S5 from Ref. [18]. Some of the common tasks using a smart mobile device together with the corresponding average energy usage excluding backlight is shown in Table 2 [18].

The research carried out in Ref. [15], showed that browsing the internet through Wi-Fi or 3G-network can consume up to about 1500 mW of power at brightness level of 36 while watching YouTube videos at brightness level of 102 can exhaust up to 1800 mW. These figures give an idea of how much power is required to carry out some tasks on smartphones. Additionally, researchers at CNET and AVG, made a list of android applications that are resource intensive and which can even bring the latest smartphones to their knees. The compiled list from Refs. [19,20] of the applications that CNET and AVG have found to negatively impact the battery life are: Facebook App, AllShareCast (Samsung), ChatON Voice and Video Chat (Samsung), Instagram, and Sportify Music. According to Refs. [21,22], the Facebook Android application by itself consumes about 58% of the battery when used in Wi-Fi mode for about half a day. This demonstrates that even with the well-known applications present on the market, developers very often do not put the extra effort into the energy-saving factor. Some power consumption facts about smart mobile devices are described next.

1.1. Screen brightness and Colour scheme

In Ref. [16], the authors determined that screen brightness is one of the battery depleting features. Therefore, the more the screen stays on, the more energy is being drained. The only way to minimize its power usage is to reduce the brightness

Table 1
Screen power consumption of Samsung Galaxy S5 [18].

	Average brightness/mW	Maximum brightness/mW
Backlight	820	1500

Table 2
Average energy usage [18].

Task	Power consumed/mW
57 s GSM phone call	1054
Sending a text message	302
Sending/Receiving email over mobile network	610
Sending/Receiving email over Wi-Fi	432

level as demonstrated in the paper. Applications which require the screen to stay on as long as the user is using the phone need to have efficient ways to deal with the long power drain. According to technicians at Howtogeek.com [23], black pixels do not produce any light compared to white pixels, and therefore use less power in AMOLED screen devices.

1.2. CPU frequency

As noted by developers at ajqi.com [24], "The higher the CPU clock, the more power is used." Extra frequency implies higher voltage power from the battery. On the other hand, low CPU frequency is synonymous to slow running of CPU intensive applications. But small applications that do not require much CPU usage can afford to run on reduced CPU frequency. As explained in Ref. [25], the CPU clock and voltage scaling is a major technique used to reduce power consumption. They described namely 2 types of CPU frequency profiles or governor types as shown in Table 3 [26].

1.3. Network

A fair Wi-Fi signal on average with an Android device utilizes less power when compared to a 3G or 2G network connection. Nevertheless this is not ideal in all situations, since according to [27], the power consumption of the mobile data and Wi-Fi would vary depending on the signal strength and there may be times where switching to a 3G or 2G network may be more efficient than accessing internet through Wi-Fi.

1.4. Maps

Google Map API provides different types of maps, and each one utilizes different amounts of power depending on the volume of data required. The maps are as described in Table 4. It has been shown in Ref. [28] that how the longer the data takes to download, the longer the wireless radio needs to stay in full power thereby draining more battery power. The frequency at which an application updates the location can also cause heavy battery consumption [28]. However, this can be lowered to achieve better battery efficiency in applications which do not require continuous location updates.

Table 3
CPU governors [26].

CPU governor	Details
Powersave	This governor sets the CPU at a minimum frequency.
Interactive/OnDemand	This governor dynamically sets the CPU frequency depending on the workload being executed by the processor.

Table 4
Map types [28].

Map type	Details
RoadMap	The roadmap type displays a simple roadmap view with the corresponding road names.
Satellite	The satellite type displays Google Earth satellite images.
Hybrid	The hybrid type is a combination of the roadmap and satellite types.
Terrain	The Terrain type displays a physical map from terrain information.

Another feature from the Maps module is the location frequency update which determines the interval at which the current location is updated. This frequency impacts the battery consumption heavily. A list pertaining to the details about how the different intervals can affect battery life is shown in Table 5 [28]. It can be inferred from Table 4 that less power is consumed with low update frequency at the expense of location accuracy.

1.5. Low power localization

There are 3 ways of obtaining the current location in an Android smartphone which are shown in Table 6. The most power consuming option is the GPS option followed by the network and finally the passive option which takes the least battery power. According to Ref. [29], GPS is the most accurate network provider despite having the most power usage. Applications which do not require high accuracy could switch to more efficient alternatives in view to save battery power.

1.6. Bluetooth

Very often mobile users tend to forget turning off the Bluetooth option after using it. It stays active even when not connected to any Bluetooth devices and continues to scan other Bluetooth enabled devices every few second until it is successfully connected to one. This process of continuous searching causes significant battery power depletion [30]. The different power modes of the Bluetooth module have been shown in Ref. [30]. It also demonstrates the power consumption of each of these Bluetooth modes in a regular smartphone. They demonstrate that an active Bluetooth takes a significant amount of battery power during different phases. Allowing continuous scanning for Bluetooth devices require considerable resources and can have a negative effect on the battery level if Bluetooth is not turned off.

1.7. Wi-Fi

According to Ref. [31], turning off the Wi-Fi radio when it is active and not connected is an efficient way for saving

Table 5
Frequency update intervals [28].

Time interval	Details
Every 5 s	This provides high location accuracy but require heavy battery power.
Every 1 min	This is the default setting, and provides a better balance between battery power and location accuracy.
Every 30 min	This provides better battery efficiency but relatively poor location accuracy.

battery life by preventing continuous scanning for access points by the wireless radio.

In this work, an API which is an application-level library is developed for battery-life optimisation of mobile devices running Android operating system. The power-saving API intends to help developers improving the battery life of any Android device running any application with the library integrated. The efficiency of the API has been tested by implementing it on a RSS-Feed-Reader and a Weather application. The system model is described in Section 2. Section 3 gives the results which have been observed from the tests performed. Section 4 concludes the work together with some future works.

2. System model

The API is designed to be non-intrusive with the main algorithms of the applications. The features monitored and regulated are shown in Fig. 2.

Table 6
Location providers [29].

Accuracy	Battery usage	Technology
20 ft	High	GPS: • Uses GPS chip on the device • Line of sight to the satellites • Need about 7 time units to get a fix • Takes a long time to get a fix • Does not work around tall buildings
200 ft	Medium–Low	Assisted GPS, Network: • Uses both GPS chip on device, and network to provide a fast initial fix • Very low power consumption • Very accurate • Works without any line of sight to the sky • Depends on carrier and phone supporting this (even if phone supports it, and network does not then this does not work)
5300 ft/1 mile	Low	Network, Wi-Fi: • Very fast lock, and does not require GPS chip on device to be active • Requires no extra power at all • Has very low accuracy; sometimes can have better accuracy in populated and well mapped areas that have a lot Wi-Fi Aps, and people who share their location with Google.

Fig. 2. Power-saving API diagram.

The approaches for battery saving techniques are explained in the following sub-sections.

2.1. Screen brightness & colour scheme

Screen brightness is a major battery issue in applications such as the RSS-Feed-Reader and Weather provider where the device's screen needs to stay on as long as the user is using the application. The API works in such a way that whenever the battery level falls below a 30%, the screen dims itself and the dark theme is automatically applied to the application. A special feature is implemented to determine whether it is daytime or nighttime by using the system clock and thus adjusting the screen brightness accordingly. It uses the idea from Ref. [23] where dark pixels use less power. The API applies a semi-transparent overlay on the screen to darken the screen pixels thus reducing the power required by the mobile phone. It also reduces the screen timeout to turn of the display earlier when the user is not using the phone. The colour scheme module changes the theme of the application from light to a dark, in order to further help consuming less power at pixel level. The screen brightness module dims the display by applying a semi-transparent overlay on the screen as shown in Fig. 3. The transparency of the overlay depends on the time of the day for a better user experience. The overlay values and time intervals can be modified by the developer to fit a particular brightness profile. The screen timeout is also reduced to 15 s to prevent the screen from staying on when not in use. An example for a brightness profile can be as shown in Table 7 [23].

2.2. CPU frequency

The weather application or the RSS-Feed-Reader do not require constant high CPU speed to function properly. Thus

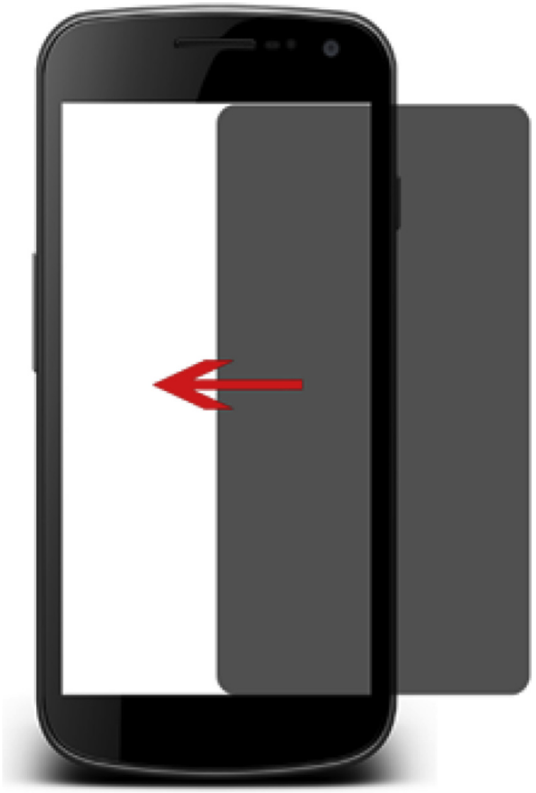

Fig. 3. Screen brightness.

the API implemented on both these applications automatically switches the CPU profiles of the Android device between the previously mentioned governors depending on the battery level of mobile phone. For instance, the API may apply the '*Powersave*' governor whenever the battery level drops below a particular level. This greatly helps extending the use of the mobile phone for a longer period with low battery level.

The power-saving API consists of a CPU frequency module which allows the modification of the device's CPU governor. It can switch between two governors, namely *onDemand* and *PowerSave*. The *onDemand* governor is the default CPU governor on most Android devices. This dynamically allocates CPU clock speed to the current workload. This governor is set when the application is closed. Whenever the battery levels falls below the 15%, the power-saving API switches the CPU governor *PowerSave* mode. This allows the CPU to run at a minimum frequency and thus reduces power consumption. When the application is closed, the default CPU governor is reset.

Table 7
Brightness profile [23].

Time intervals	Transparency level
6 am–12 pm	Low
12 pm–4 pm	Very Low
4 pm–7 pm	Medium
7 pm–9 pm	Relatively high
9 pm–6 am	High

2.3. Network

Wireless connectivity, Wi-Fi and mobile data can be made to use less battery based on their signal strengths. The power-saving API allows the RSS-Feed-Reader and the weather applications to switch between 2G, 3G and Wi-Fi connectivity. It also turns off any wireless connections during low-power or when not in use and turns it back on again when needed. This module also allows data to be downloaded automatically to the applications and cached locally whenever an internet connection is available. The API implements two features: Auto-Download and Network Switching as explained next.

2.3.1. Auto-Download

The API implements a feature which allows any application using the framework to capture internet connectivity and start downloading data even if none is running. This data is locally cached and used when required by the application. In the case of the RSS-Feed-Reader, the API pre-fetches and caches data every 4 h when connected to the internet and for the weather application this is done every 6 h.

2.3.2. Network Switching

The other feature which the wireless connectivity module consists of is that the power-saving API automatically chooses the most efficient way to allow the application to connect to the internet. Whenever the battery level falls below 30%, and the Android application is connected to the internet, the API switches between Wi-Fi, 3G or 2G connectivity as shown in Table 8 [32].

2.4. Low power localization

The feature implemented in the API uses the 'Network Provider' to get the current location of the Android device. The Wi-Fi connection uses the efficient capabilities of the Low-Power-Localisation, to connect to known hotspots which are saved to a database. When the location of a known access-point is found in a radius of 2.5 km from the current location, the power-saving API is automatically turned on and connects to the access point.

2.5. Wi-Fi

The power-saving API implements a module whereby Wi-Fi is automatically turned off when not connected to any access point and turned back on when a known Wi-Fi access point is available by using the *Low-power Localization* module. It constantly learns where the hotspots are located. The

Table 8
Network Switching [32].

Current network	Condition	Switching network
Wi-Fi	<−70 dB (~35%)	3G/2G
3G	<−110 dB (~55%)	2G/Wi-Fi
2G	<−125 dB (~30%)	Wi-Fi

Wi-Fi module intelligently gives the user 5 min to connect to an access point before turning it off if it is still not connected.

2.6. Maps

The power-saving API provides efficient ways to display the maps. Reducing the map details decreases the amount of data needed and thus the time for the wireless radio to stay active. This method helps reducing power consumption. Another feature from the Maps module is the location frequency update which determines the interval at which the current location is updated. The two features implemented in the API are described next.

2.6.1. Low-bandwidth maps

This module provides developers with energy-efficient Map utilities. The API consists of a map that consumes/downloads less data as compared to normal maps. This is achieved by using RoadMap.

2.6.2. Custom update frequency of maps

This feature allows modification of the update frequency of applications using location updates. Applications like the weather application which require regular location updates may use such feature to save battery. This value is changed through the *Shared Preference* Screen interface as shown in Table 9.

2.7. Bluetooth

The power-saving API automatically detects whether the Bluetooth radio is active and not connected to any devices, and turns it off to prevent the battery drainage. The Bluetooth module turns off the Bluetooth radio whenever it is not in use. This occurs when the Bluetooth is initially active and has not been connected to any devices for a period of 5 min.

2.8. Graphical user interface

The power-saving API implements a *Shared Preference* menu which allows the user to change or disable some features set by the power-saving API. The proposed design of the *Shared Preference* menu is shown in Fig. 4.

3. Results

In this work, the power saving API has been tested using two routed Android smartphones equipped with Bluetooth, Wi-Fi, mobile data and GPS. Some of the features of the API work only for Android API 22 and above. The specifications of the mobile devices used are as shown in Table 10.

Table 9
Update frequency.

Frequency location update	Efficiency
30 min	Very efficient
10 min	Relatively efficient
3 min	Not efficient
1 min	Not efficient, drains battery faster

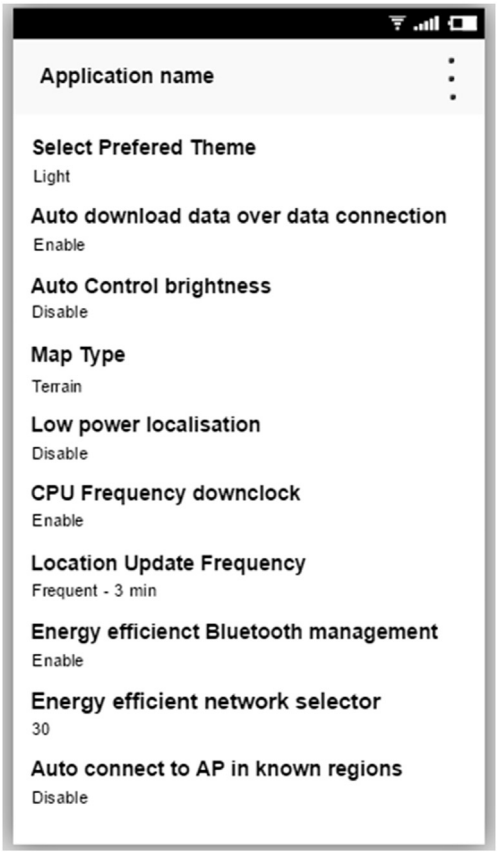

Fig. 4. Shared preference screen.

The efficiency measurement of the power-saving API and the monitoring of the battery usage have been performed using three power profiling Android applications: Trepn Profiler from Qualcomm Inc (for CPU and wireless power profiling), Power Tutor (for screen power) and Little Eye Labs (for network, wireless and CPU analysis). The Enersave modules in the API have been tested separately to measure and

Table 10
Mobile devices specifications.

	LG L70	Samsung A8
Processor	Dual-core 1.2 GHz	Quad-core 1.8 GHz & Quad-core 1.3 GHz
RAM	1 GB	2 GB
Screen	IPS LCD capacitive 4.5 inches	Super AMOLED 5.7 inches
Resolution	480 × 800 pixels	1080 × 1920 pixels
Wi-Fi	Wi-Fi 802.11 b/g/n	Wi-Fi 802.11 a/b/g/n/ac
Bluetooth	v 4.0	V 4.1
GPS	Yes, with A-GPS	Yes, with A-GPS
Network support	2G, 3G	2G, 3G, 4G
Battery	Li-Ion 2100 mAh	Li-Ion 3050 mAh
Operating system	Android OS, v 5.1.1 Lollipop	Android OS, v 5.1.1 Lollipop

compare the efficiency of the proposed Framework. The results obtained for each aspect is described in the following sub-sections.

3.1. Results for the screen brightness module

The Screen Brightness test is performed by setting the following features on the Android device to allow more accurate results:

1. Screen brightness is set to maximum.
2. Android mobile devices are charged to maximum and unplugged.
3. Colour Scheme is set as default, light.
4. All wireless radios (Wi-Fi, Network, Bluetooth, and GPS) are turned off.
5. Overlay transparency is at its lowest value when performing tests (6 am–12 pm).

The tests have been performed on the Weather application using the *Power Tutor* android application. Readings have been taken 3 times with both devices and averaged to produce better accuracy. The readings have been taken at an interval of 3 min for 15 min long. The graphical representation of how using the Enersave power-saving API consumes less **Screen Power** than the normal application without power-saving features is shown in Fig. 5.

Table 11 shows the average values when measuring the effectiveness of the Enersave power-saving API for the **Screen Brightness** module.

A slight difference is noticed when integrating the Enersave power-saving API with the Android application. This is due to dimming the pixels present in the AMOLED display of the phone. It is observed that after 15 min only **0.64%** battery power is saved. It can be inferred that the screen brightness module alone does not have a significant impact on the battery consumption.

Table 11
Screen Brightness – Average Power.

Average Screen Power without API (mW)	891.9
Average Screen Power with API (mW)	886.1
Average Screen Power saved (mW)	5.6

3.2. Results for the colour scheme module

The Colour Scheme test is performed by setting the following features on the Android mobile devices to allow more accurate readings:

1. Screen brightness is set to maximum and disable all other power-saving features.
2. Android mobile phones are charged to maximum and unplugged.
3. All wireless radios (Wi-Fi, Network, Bluetooth, and GPS) are turned off.

The tests have been performed on the Weather application using the *Power Tutor* android application. Readings have been taken 3 times with both devices and averaged to produce better accuracy. They were taken at an interval of 3 min for 15 min long. The graphical representation of how using the Enersave power-saving API consumes less power than the normal application without power-saving features is shown in Fig. 6.

Table 12 shows the average values when measuring the effectiveness of the Enersave power-saving API for the **Colour Scheme** module.

A better value for the screen energy saved is noticed using the *Dark Theme* for the **Colour Scheme** module. This is because, compared to the **Screen Brightness** module which only dims the pixels on the screen, the **Colour Scheme** module turns them off in places of black colour. A save of **1.41%** battery power is compiled for this feature.

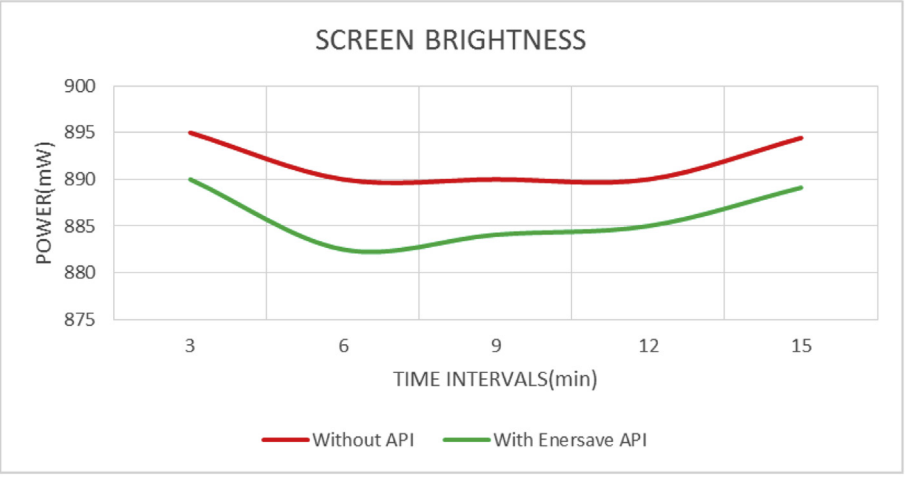

Fig. 5. Graph – Screen Brightness.

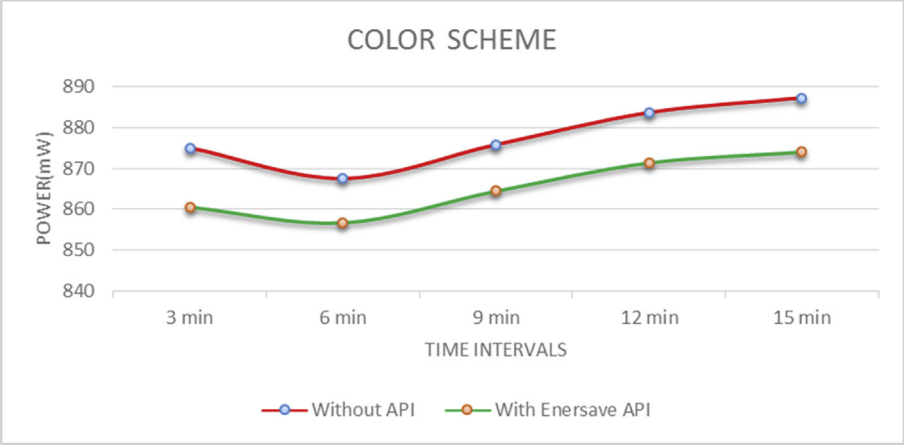

Fig. 6. Graph – Colour Scheme.

Table 12
Colour Scheme – Average Power.

Average Screen Power without API (mW)	877.8
Average Screen Power with API (mW)	865.4
Average Screen Power saved (mW)	12.4

3.3. Results for the screen timeout module

The Screen Timeout test is performed by setting the following features on the Android mobile devices to allow more accurate readings:

1. Screen brightness is set to maximum.
2. Android mobile phones are charged to maximum and unplugged.
3. All wireless radios (Wi-Fi, Network, Bluetooth, and GPS) are turned off.
4. All default settings are set to the mobile device (No other power-saving features enabled).

Fig. 7 shows a cumulative graph of the amount of power which is used by the screen at different **screen timeout values**. It can be observed that the larger the **screen timeout value** the more power is consumed by the screen. Setting the **Screen timeout** to 15 or 30 s is a safe way of preserving battery life.

3.4. Results for the Bluetooth module

The Bluetooth test is performed by setting the following features on the Android mobile devices to allow more accurate readings:

1. Android mobile phones are charged to maximum and unplugged.

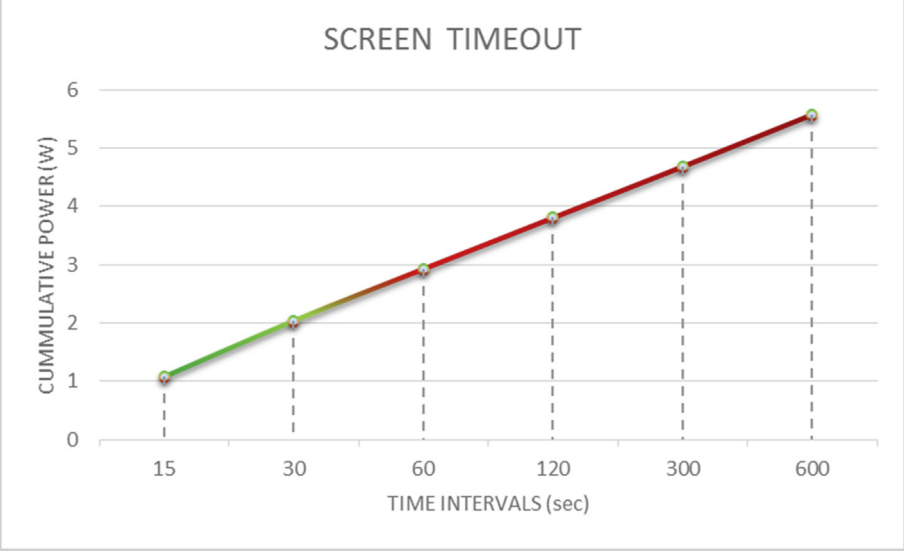

Fig. 7. Graph – Colour Scheme.

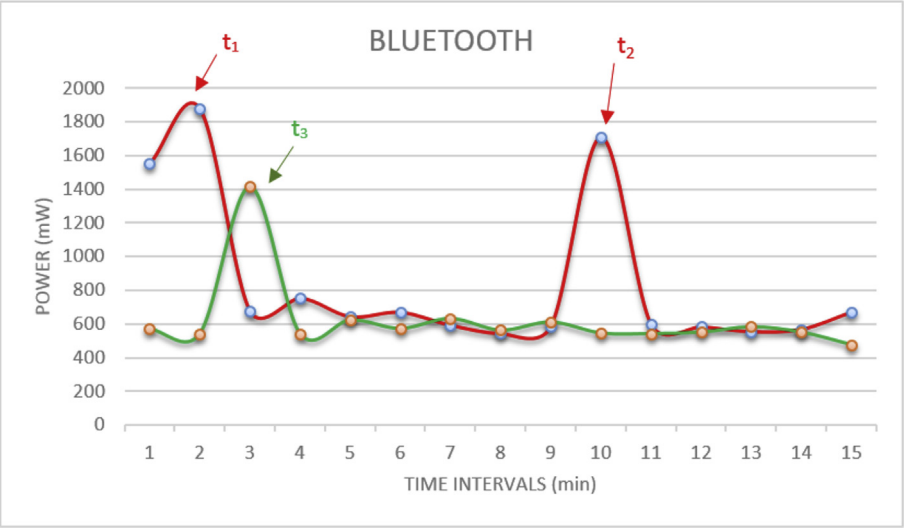

Fig. 8. Graph – Bluetooth.

Table 13
Bluetooth – Average Power.

Average Screen Power without API (mW)	840.4
Average Screen Power with API (mW)	622.5
Average Screen Power saved (mW)	217.9

2. All wireless radios (Wi-Fi, Network, Bluetooth, and GPS) are turned off.

Measurements have been taken from the RSS-Feed-Reader application using the *Little Eye Labs* power analysis application. Readings have been taken 3 times with both devices and averaged to produce better accuracy. The readings are taken at an interval of 3 min for 15 min long. Fig. 8 shows the power profile with the Bluetooth test.

At time $\mathbf{t_1}$ and $\mathbf{t_2}$, the Bluetooth radio performs searching to connect to a Bluetooth device. This process continues until the device is successfully connected or Bluetooth turned off. At time $\mathbf{t_3}$, the Enersave power-saving API detects that the Bluetooth radio is active and not connected and thus turns it off. Table 13 shows the average values when measuring the effectiveness of the Enersave power-saving API for the Bluetooth module.

A significant power save of **25.92%** is noticed with this feature. Preventing further searching of Bluetooth devices when the Bluetooth radio is enabled is an efficient way of saving battery power.

3.5. Results for the CPU frequency module

The CPU test is performed by setting the following features on the Android mobile device to allow more accurate readings:

1. Android mobile phones are charged to maximum and unplugged.
2. All wireless radios (Wi-Fi, Network, Bluetooth, and GPS) are turned off.

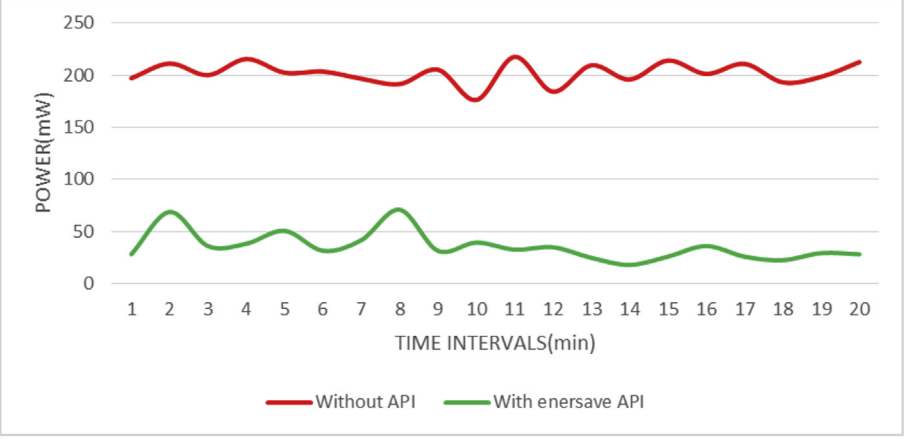

Fig. 9. Graph – CPU.

Table 14
CPU – Average Power.

Average Screen Power without API (mW)	201.99
Average Screen Power with API (mW)	35.79
Average Screen Power saved (mW)	166.20

Table 15
Low-Power Localization – Average Power.

Average Screen Power without API (mW)	175.0
Average Screen Power with API (mW)	66.1
Average Screen Power saved (mW)	108.9

3. All default settings are set (Brightness, colour scheme, CPU governor).

Measurements have been taken from the Weather application using the *Little Eye* power analysis application. Readings have been taken 3 times with both devices and averaged to produce better accuracy. Readings are taken for 20 s. Fig. 9 shows the power profile with the CPU test.

Table 14 shows the average values when measuring the effectiveness of the Enersave power-saving API for the CPU module.

A substantial power save of **82.28%** is noticed using this feature. Lowering the CPU frequency is a great way for small applications to prolong battery life.

3.6. Results for the Low Power Localization frequency module

The Low-Power Localization test is performed by setting the following features on the Android mobile device to allow more accurate readings:

1. Android mobile phones are charged to maximum and unplugged.
2. All default settings are set (Brightness, colour scheme, CPU governor).

Measurements have been taken from the Weather application using the *Little Eye* power analysis application. Readings have been taken 3 times with both devices and averaged to

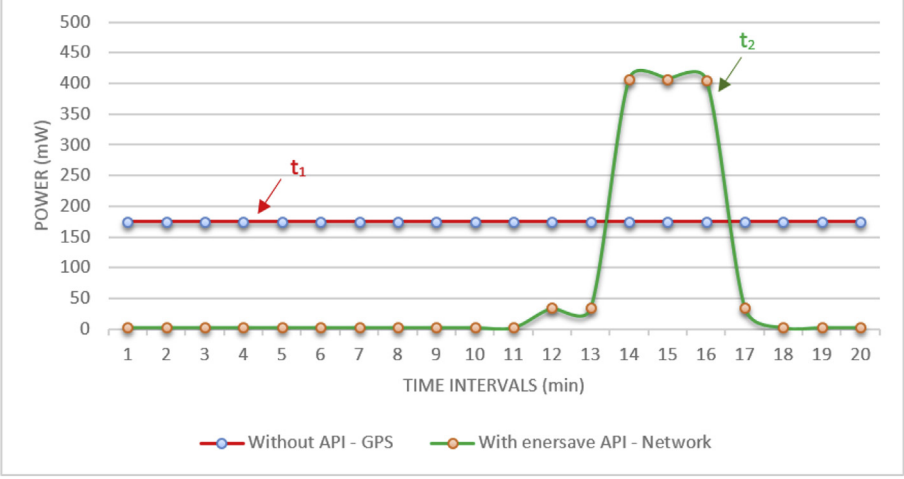

Fig. 10. Graph – Low-Power Localisation.

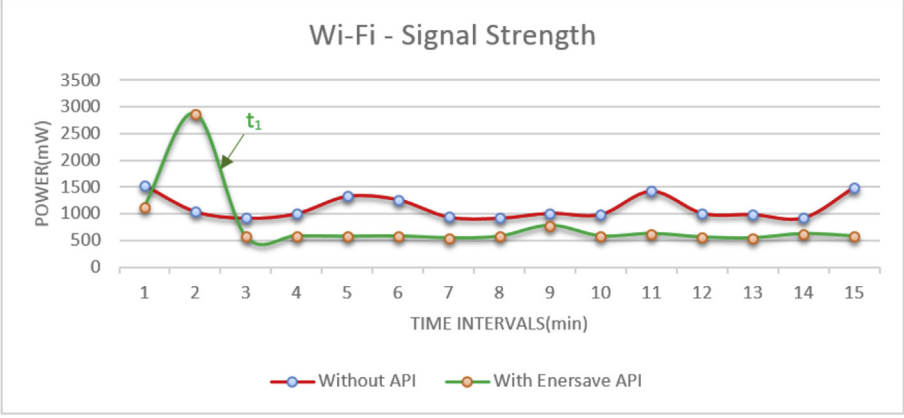

Fig. 11. Graph – Auto Turn off Low Signal.

Table 16
Auto turn off Wi-Fi – Average Power.

Average Screen Power without API (mW)	1120.8
Average Screen Power with API (mW)	780.2
Average Screen Power saved (mW)	340.6

Table 17
Low Power Localisation Wi-Fi – Average Power.

Average Screen Power without API (mW)	971.6
Average Screen Power with API (mW)	557.8
Average Screen Power saved (mW)	413.8

produce better accuracy. Readings are taken for 20 s. Fig. 10 shows the power profile with the Low-Power Localisation test.

At time $\mathbf{t_1}$, to retrieve the current location, the *GPS* is used. It uses a constant amount of power to be able to obtain the required coordinates. At time $\mathbf{t_2}$, to retrieve the current location, the *Network Provider* is used. It only uses a high burst of power to retrieve the current coordinates of the user. Table 15 shows the average values when measuring the effectiveness of the Enersave power-saving API for the CPU module.

A considerable power save of **62.2%** is noticed using this feature. For applications which do not require accurate location, the **Low Power Localization** is a battery efficient way of saving battery power.

3.7. Results for the Wi-Fi module

The Wi-Fi test is performed by setting the following features on the Android mobile device to allow more accurate readings:

1. Android mobile phones are charged to maximum and unplugged.
2. All default settings are set (Brightness, colour scheme, CPU governor).

The tests are performed in two parts: Auto Turn off during Low Wi-Fi Signal and Low Power Localisation Wi-Fi/Auto Connect/Disconnect to Access points. The two tests are shown in the following sub-sections.

3.7.1. Auto Turn off during low Wi-Fi Signal

Fig. 11 shows the power profile with the Auto Turn off during low Wi-Fi Signal test.

At time $\mathbf{t_1}$, the Enersave power-saving API detects that the Wi-Fi signal is low and automatically turns off the Wi-Fi radio. Without the API, more power is required to stay connected to the access point during low signal strength. Table 16 shows the average values when measuring the effectiveness of the Enersave power-saving API for the Auto Turn off Wi-Fi module.

A significant power save of **30.4%** is noticed using this feature. Turning off the Wi-Fi radio during low signal strength can be of great benefit for the battery life.

3.7.2. Low Power Localization Wi-Fi/Auto Connect/Disconnect to access points

Fig. 12 shows the power profile with the Low Power Localization Wi-Fi/Auto Connect/Disconnect to Access Points test.

At time $\mathbf{t_1}$, the Enersave power-saving API detects that the mobile device is in a known hotspot location and turns on the Wi-Fi radio to automatically connect to the access point. At time $\mathbf{t_2}$, the API automatically turns off the Wi-Fi when the mobile devices move away from the location of the access point. Without the API, the mobile device keeps using the mobile data to access the Internet even if the user can connect to a known Wi-Fi access point. Table 17 shows the average values when measuring the effectiveness of the Enersave power-saving API for the Auto Turn off Wi-Fi module.

A power save **42.6%** is observed when using this feature. Using Wi-Fi rather than the mobile data when a known hotspot is available can be efficient.

3.8. Results for the Map module

The Map test is performed by setting the following features on the Android mobile device to allow more accurate readings:

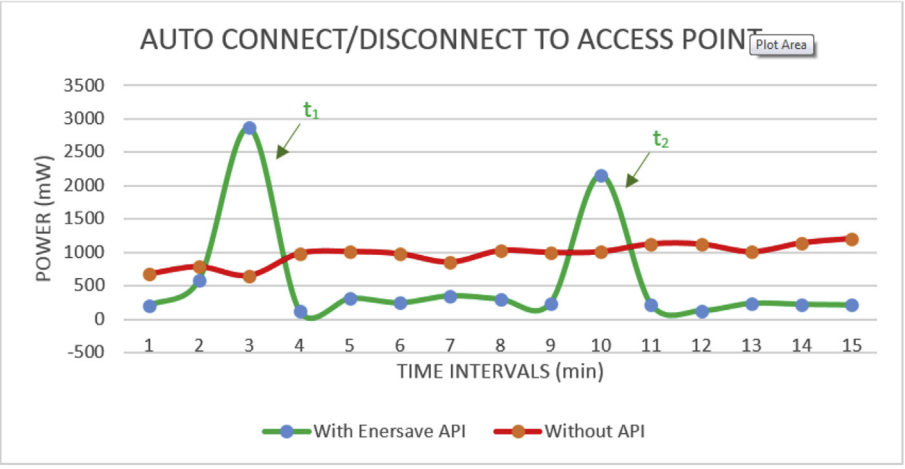

Fig. 12. Graph – Low Power Localisation Wi-Fi.

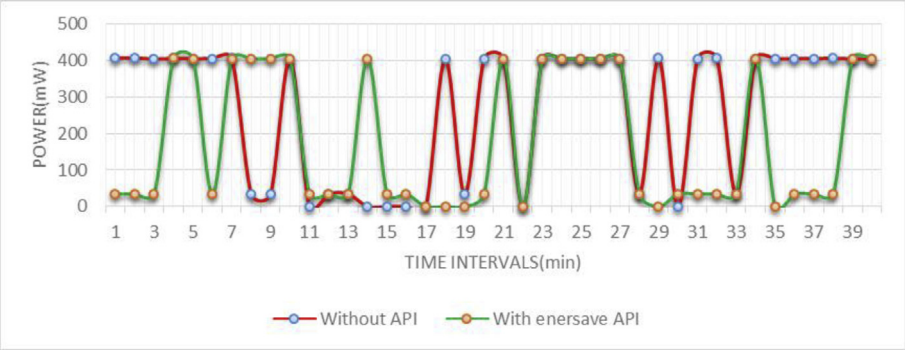

Fig. 13. Graph – Low Bandwidth Map – power consumption.

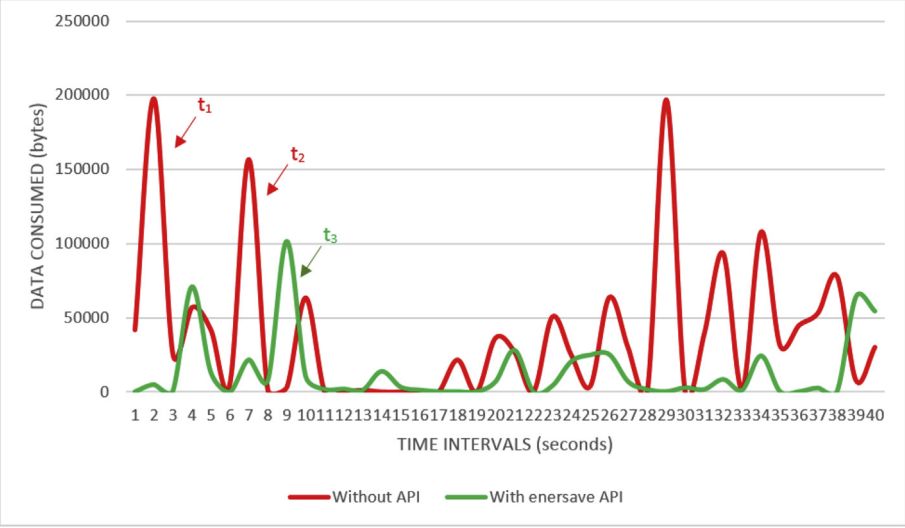

Fig. 14. Graph – Low Bandwidth Map – Data Consumption.

1. Android mobile phones have been charged to maximum and unplugged.
2. All default settings are set (Brightness, colour scheme, CPU governor).
3. The '*Network Provider*' is used to get location.

The tests are performed in two parts: Low Bandwidth Map and One-time location update. Measurements have been taken from the Weather application using the *Little Eye* and the Trepn power analysis application. The two tests are described in the following two subsections.

3.8.1. Low Bandwidth Map

Figs. 13 and 14 show the graphs for *Power (mW)* and *Data Consumed (bytes)* with the **Low Bandwidth Map** feature.

Table 18 shows the average values when measuring the effectiveness of the Enersave power-saving API for the **Low Bandwidth Map** module.

A power save of **34.2%** is observed using this feature. Low detailed maps are found to be very battery efficient.

At time $\mathbf{t_1}$ and $\mathbf{t_2}$, the high peaks indicate large data which is being downloaded for the maps whereas at time $\mathbf{t_3}$, the peaks are lower indicating smaller data which is being downloaded.

3.8.2. One-time location update

Fig. 15 shows a graphical representation of how using the Enersave power-saving API consumes less power than the normal application without power-saving features.

At time $\mathbf{t_1}$, $\mathbf{t_2}$ and $\mathbf{t_3}$, the application continuously updates the location of the mobile device. With the Enersave API, the

Table 18
Low Bandwidth Map – Average Power.

Average Screen Power without API (mW)	269.7
Average Screen Power with API (mW)	177.6
Average Screen Power saved (mW)	92.1

Table 19
One-time update – Average Power.

Average Screen Power without API (mW)	844.7
Average Screen Power with API (mW)	622.7
Average Screen Power saved (mW)	222.0

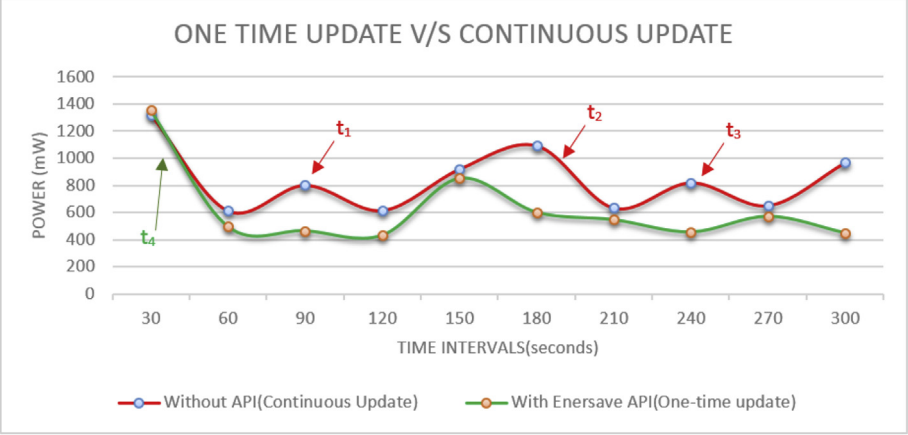

Fig. 15. Graph – One time Location Update.

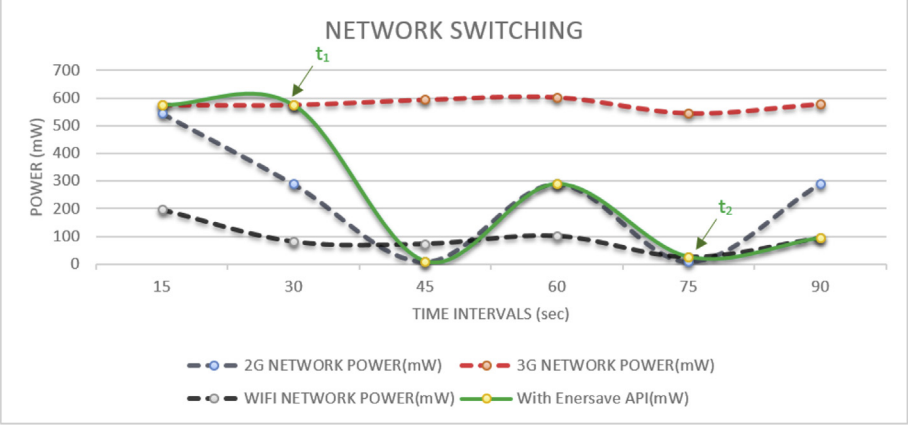

Fig. 16. Graph – Network Switching.

location update is done only once at time $\mathbf{t_4}$. Table 19 shows the average values when measuring the effectiveness of the Enersave power-saving API for the **One-time update** module.

A power save **26.3%** is observed when using this feature. Applications which do not require continuous update of the current location, similar to the Weather application, can use the **One-time update** feature for battery usage efficiency.

3.9. Results for the network module

The Network test is performed by setting the following features on the Android mobile device to allow more accurate readings:

1. Android mobile phones are charged to maximum and unplugged.
2. All default settings are set (Brightness, colour scheme, CPU governor).
3. The test is conducted in a low signal network area.

The tests are performed in two parts: Network Switching and Auto-Download. A description of the two aspects is given next.

3.9.1. Network Switching

Measurements are taken from the Weather application using the *Little Eye* power analysis application. Fig. 16 shows a graph when the mobile devices are connected to the internet initially using 3G network and at low signal the API switches the connection to 2G network. At very weak network signal, the API switches to Wi-Fi to connect to the internet.

At time $\mathbf{t_1}$, the Enersave API automatically switched from the 3G network to the 2G network to connect to the internet. At time $\mathbf{t_2}$, the API switches from the mobile data to Wi-Fi to connect to the internet. Table 20 shows the average values when measuring the effectiveness of the Enersave power-saving API for the **Network-Switching** module.

A power save **54.73%** is observed when using the API to switch to the most efficient wireless connection rather than continuously using the 3G network in a low signal area.

Table 20
Network Switching – average power.

Average 3G Network Power without API (mW)	578.3
Average 2G Network Power without API (mW)	239.2
Average Wi-Fi Network Power without API (mW)	96.2
Average Network Power with API (mW)	261.8

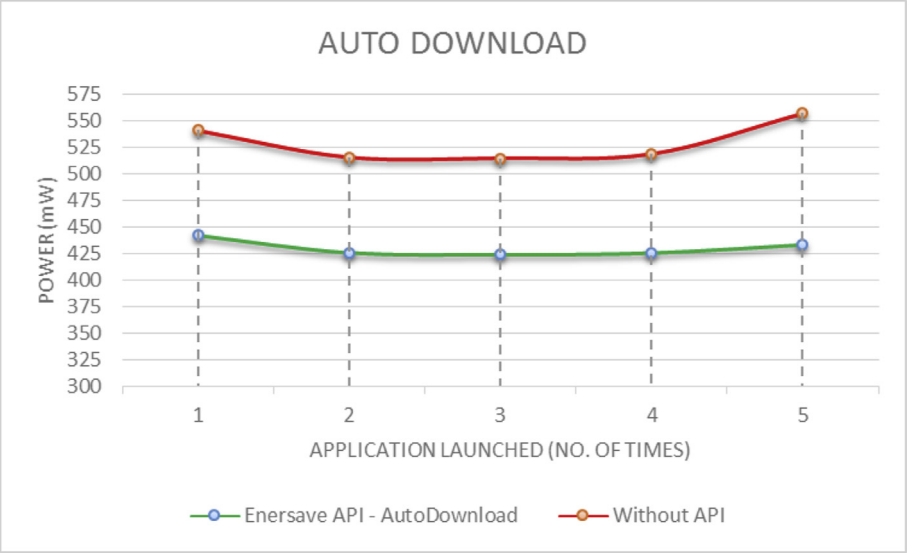

Fig. 17. Graph – Auto-Download.

Table 21
Network Auto Download – Average Power.

Average Power without API (mW)	529.06
Average Power with API (mW)	430.04
Average Power saved (mW)	99.02

Table 22
Test Plan – Enersave Testing.

With Enersave API	Without Enersave
Brightness 50%	Brightness 50%
Colour Scheme	No Colour Scheme
Overlay	No Overlay
CPU Frequency down-clock	No CPU Frequency down-clock
Low Bandwidth Map	High/Normal Bandwidth Map
Low-Power Localisation (Network)	High/Normal Power Localisation (GPS)

3.9.2. Auto-Download

Measurements are taken from the RSS-Feed-Reader using the *Little Eye* power analysis application. Fig. 17 shows a graph when the data from the RSS feed is updated each time the application is launched and another one where the API automatically auto download and cache data when internet connection is available.

Table 21 shows the average values when measuring the effectiveness of the Enersave power-saving API for the **Network–AutoDownload** module.

A power save **18.72%** is observed when using this feature. Preventing data to be downloaded every time the application is launched can have great impact on battery life for applications similar to the RSS-Feed-Reader or the weather applications.

3.10. Evaluation of the complete system

In this section, tests performed on the complete Enersave API system for the measurement and comparison of efficiency of the framework are described. The Enersave API is tested using the Weather Application and the Little Eye Labs Analysis tools with the parameters shown in Table 22. The results obtained with and without using the Enersave API are shown in Figs. 18 and 19 respectively.

The comparisons for the tests with and without the Enersave API are shown in Table 23.

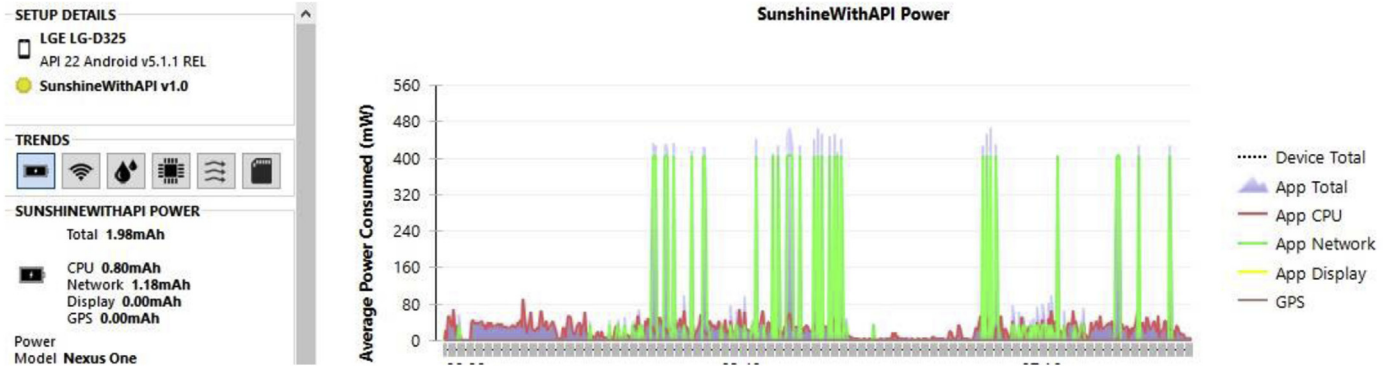

Fig. 18. Result – with Enersave API.

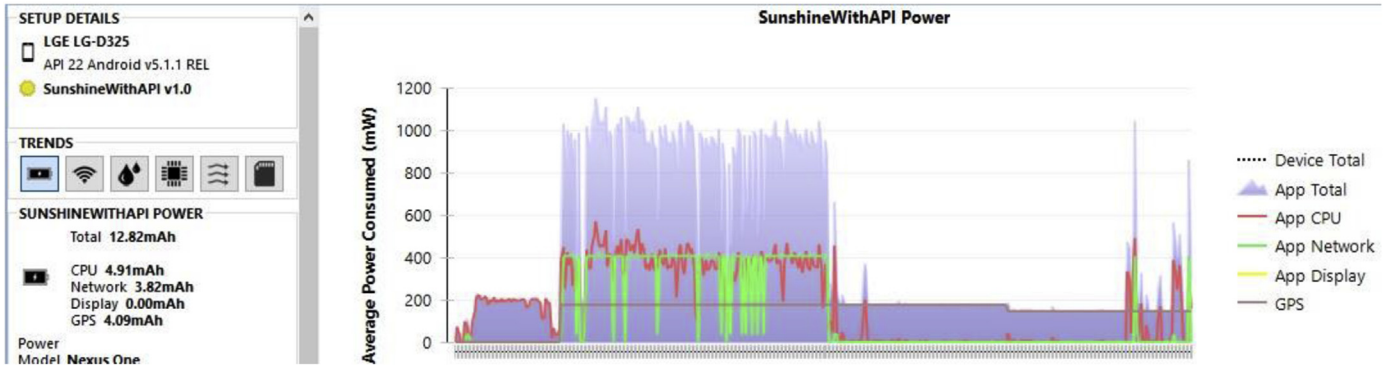

Fig. 19. Result – Without Enersave API.

Table 23
Test Plan – Enersave Testing.

	With Enersave API/mAh	Without Enersave/mAh
CPU	0.80	4.91
Network	1.18	3.82
GPS	0.00	4.09
Total	**1.98**	**12.82**

The results above indicate a significant save of **84.56%** in terms of power consumption is obtained when using the Enersave API.

In order to assess the efficiency of the proposed framework, similar tests have been performed with existing systems to compare with the Enersave API. The results obtained with Battery Saver 2016 are shown in Fig. 20, and the results obtained with DU Booster are shown in Fig. 21. Table 24 summarizes all the tests performed. The reason for making the comparisons between these two existing energy serving applications (DU Booster and Battery Saver 2016) for Android devices in this way is to perform a fair comparison and that we have to the best of our knowledge not come across any work which has considered all of the features: (Screen brightness, Colour scheme, CPU frequency, 2G/3G network, Maps, Low power localisation, Bluetooth and Wi-Fi) together in a single application.

The comparison table shows that the Enersave API is much better at saving battery life than the regular power saving Android applications.

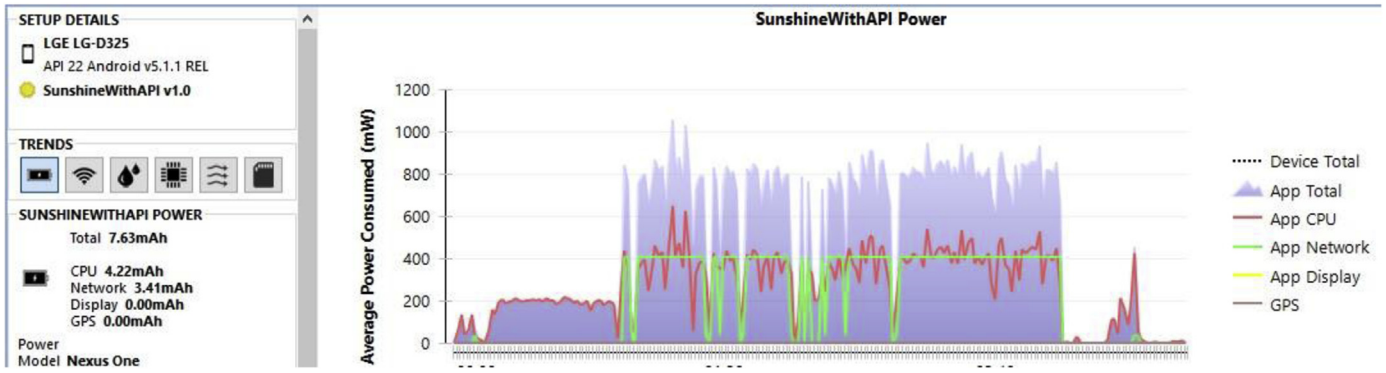

Fig. 20. Result – Battery Saver 2016.

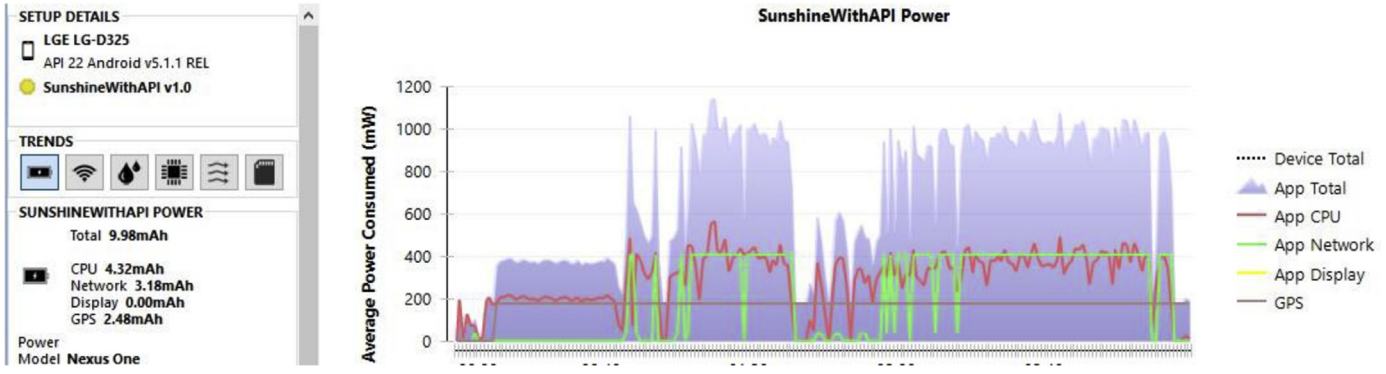

Fig. 21. Result – DU Booster.

Table 24
Results – comparison.

	With Enersave API/mAh	Battery Saver 2016/mAh	DU Booster/mAh	Without Enersave/mAh
CPU	0.80	4.22	4.32	4.91
Network	1.18	3.41	3.18	3.82
GPS	0.00	0.00	2.48	4.09
Total	1.98	7.63	9.98	12.82
% Save	84.56	40.48	22.15	0.00

4. Conclusion

The work presented in this paper proposes a power-saving framework (Enersave API) for Android based mobile devices. Enersave is an API which can be integrated effortlessly to any Android application by the developer. The API allows for undesired features to be disabled through the *Shared Preference Screen* in order to save on battery power consumption. Tests performed demonstrate that the proposed framework helps saving 84.56% of battery power consumption as compared to 22.15% and 40.48% when using DU Booster and Battery saver 2016 respectively which are current energy saving applications available for Android devices.

Several future works can be envisaged from the proposed framework. A direct extension would be to have battery power consumption profiles sent over to a cloud platform where pattern analysis could be performed and signals could be sent to the mobile devices so that the settings are automatically adjusted to the most optimum and battery power saving settings. Another more challenging future work would be to develop a cross-platform API which would be used in conjunction with a cloud platform so as to aid towards low power consumption of mobile sensor nodes being used in the Internet of Things context.

Acknowledgement

The authors would like to acknowledge the University of Mauritius for the necessary facilities provided towards the completion of this work.

References

[1] Hurbungs V, Beeharry Y, Calkee AK, Ahotar G. An energy efficient android application. ADBU J Eng Technol 2016;5(2016):1–10.
[2] Lundquist AR, Lefebvre EJ, Garramone SJ. Smartphones: fulfilling the need for immediacy in everyday life, but at what cost? Int J Human Soc Sci 2014;4(2):80–9.
[3] Korhonen K. Predicting mobile device battery life. Espoo, Finland: Alto University, School of Engineering; 2011.
[4] I. -. A. t. Future. IDC: smartphone OS market share. 2015 [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp [accessed 11.09.15.].
[5] Source.android.com. Source.android.com. 2015 [Online]. Available: http://source.android.com [accessed 01.09.15.].
[6] Statista. Android – Statistics & facts | Statista. 2015 [Online]. Available: http://www.statista.com/topics/876/android/ [accessed 06.09.15.].
[7] Raadschelders J, Jansen T. Energy sources for the future dismounted soldier, the total integration of the energy consumption within the soldier system. J Power Sources 2001;96(1):106–66.
[8] Zeng H, Lebeck C, Vahdat A. ECOSystem. ACM SIGOPS operating systems Review, vol. 31 (5); 2002. p. 123.
[9] Narseo V, Hui P, Crowcrott J, Rice A. Exhausting battery statistics: understanding the energy demands on mobile handsets. In: Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds, New York; 2010.
[10] Kang J, Seo S, Hong J. Personalised battery lifetime prediction for mobile devices based on usage patterns. J Comput Sci Eng 2011;5(4):338–45.
[11] Batteryuniversity.com. Charging lithium-ion batteries. 2015 [Online]. Available: http://batteryuniversity.com/learn/article/charging_lithium_ion_batteries [accessed 20.09.16.].
[12] Jason T, Mohamed M. How long does it take to fully charge the batteries? – Apple. Apple.com. 2010 [Online]. Available: http://www.apple.com/shop/question/answers/product/MC500LL/A/how-long-does-it-take-tofully-charge-the-batteries/QPYA7AUXXXTCYPYPU [accessed 29.09.15.].
[13] Lin M, Gong M, Lu B, Wu Y, Wang D, Guan M, et al. "An ultrafast rechargeable aluminium-ion battery. Nature 2015;520(7547):324–8.
[14] Gsmarena. Apple iPhone 5s – full phone specifications. Apple 2015 [Online]. Available: http://www.gsmarena.com/apple_iphone_5s-5685.php [accessed 28.09.15.].
[15] Zhang L. Power, performance modelling and optimisation for mobile system and applications. Michigan: University of Michigan; 2013.
[16] Caroll A, Heiser G. An analysis of power consumption in a smartphone. In: Proceedings of the 2010 USENIX conference on USENIX, USA; 2010.
[17] Network QD. Mobile apps and power consumption – basics, part 1. 2015 [Online]. Available: https://developer.qualcomm.com/blog/mobile-apps-and-powerconsumption-basics-part-1 [accessed 30.09.15.].
[18] Hruska J. OLED finally triumphant: the Galaxy S5 has the best smartphone display on the market | ExtremeTech. 2015 [Online]. Available: http://www.extremetech.com/electronics/179464-oled-finally-triumphant-the-galaxy-s5-has-thebest-smartphone-display-on-the-market [accessed 30.09.15.].
[19] CNET. What's killing your battery? Android's top 10 performance-scapping apps. 2015 [Online]. Available: http://www.cnet.com/news/whats-killing-your-battery-androids-top-10- [accessed 07.10.15.].
[20] AVG. Are these the world's greediest apps?. 2015 [Online]. Available: http://now.avg.com/are-these-the-worlds-greediest-apps [accessed 07.10.15.].
[21] Malik MY. Power consumption analysis of a modern smartphone. Cornell University Library; 2013.
[22] Forums.androidcentral.com. Facebook battery DRAIN – android Forums at AndroidCentral.com. 2014 [Online]. Available: http://forums.androidcentral.com/t-lg-g2/358723-facebook-battery-drain.html [accessed 10.10.15.].
[23] Howtogeek.com. HTG explains: does plain black wallpaper save battery on mobile devices?. 2015 [Online]. Available: http://www.howtogeek.com/131823/htg-explains-does-blackwallpaper-save-battery-on-your-mobile-devices/ [accessed 12.10.15.].
[24] Google. AJQI. 2015 [Online]. Available: http://ajqi.com/ [accessed 30.10.15.].
[25] Hwang H, Suh H. "Personal behavior-based dynamic governor switching an android system. Adv Sci Technol Lett 2014;66:4–7.
[26] Icrontic. Android CPU governors and you! (SetCPU, system Tuner, TegraK). 2014 [Online]. Available: http://icrontic.com/discussion/95140/android-cpu-governors-and-you-setcpusystem-tuner-tegrak [accessed 14.01.16.].
[27] Ding N, Wagner D, Chen X, Hu Y, Rice A. Characterising and modelling the impact of wireless signal strength on smartphone battery drain. ACM SIGMETRICS Perform Eval Rev 2013;41(1):29.

[28] G. Developers. Map types. 2015 [Online]. Available: https://developers.google.com/maps/documentation/javascript/maptypes?hl=en#BasicMapTypes [accessed 14.01.16.].

[29] Nazmul I. Android location providers – gps, network, passive – tutorial. 2010 [Online]. Available: http://developerlife.com/tutorials/?p=1375 [accessed 13.01.16.].

[30] Lindh J, Lee C. Measuring Bluetooth smart power consumption. 2015 [Online]. Available at: http://www.ti.com/lit/an/swra478a/swra478a.pdf [accessed 20.10.15.].

[31] Technology II. Simple ways to boost your smartphone's battery life. 2011 [Online]. Available: http://www.intel.com/content/dam/doc/bestpractices/technology-tips-boost_battery_life_handhelds.pdf [accessed 03.02.16.].

[32] Hyun J, Won Y, Nahm DS-C, Hong JW-K. Measuring auto switch between Wi-Fi and mobile data networks in an urban area. 12th International Conference on Network and Service Management (CNSM), Montreal, QC, Canada. 2016.