Πανεπιστήμιο Θεσσαλίας, 2013-2014

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Θέμα:

<< Σχεδιασμός Και Υλοποίηση Αλγορίθμων Εξοικονόμησης Ενέργειας Για Κινητές Συσκευές >>

<< Design and Implementation of Energy Saving Algorithms for Mobile Devices >>

Σούτης Βασίλειος

UNIVERSITY OF THESSALY

Επιβλέπων Καθηγητής:

Κοράκης Αθανάσιος (Λέκτορας Καθηγητής)

Συν επιβλέπων Καθηγητής:

Λάλης Σπυρίδων (Αναπληρωτής Καθηγητής)

Βόλος, Σεπτέμβριος 2014

Ευχαριστίες,

Με την εκπόνηση της παρούσας Διπλωματικής εργασίας, φέρνω εις πέρας τις προπτυχιακές μου σπουδές στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας.

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον Λέκτορα του Τμήματος Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών κ. Κοράκη Αθανάσιο, για την ανάθεση της παρούσας Διπλωματικής Εργασίας και για την ευκαιρία συνεργασίας με την ομάδα του NITlab που μου έχει προσφέρει. Από καρδίας θα ήθελα να ευχαριστήσω ολόκληρη την ομάδα του NITlab για την καθοδήγηση που μου έδωσε καθόλη τη διάρκεια της εκπόνησης της Διπλωματικής μου εργασίας. Ιδιαιτέρως να ευχαριστήσω τον Διδάκτορα του Τμήματος Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών Κερανίδη Στράτο, τον Μεταπτυχιακό φοιτητή Πασσά Βιργίλιο και τον υποψήφιο Διδάκτορα Καζδαρίδη Γιάννη για την βοήθεια και τις υποδείξεις στην εκπόνηση της εργασίας, καθώς και την εμπιστοσύνη που έδειξαν στο πρόσωπό μου.

Ευχαριστώ θερμά την οικογένεια μου για την αμέριστη συμπαράσταση που μου παρείχε όλα αυτά τα χρόνια για την ολοκλήρωση των προπτυχιακών μου σπουδών. Ιδιαίτερα τους ευχαριστώ για την στήριξη και την εμπιστοσύνη που επέδειξαν στις επιλογές και στην κρίση μου.

Τέλος θα ήθελα να ευχαριστήσω όλους τους υπέροχους φίλους που απόκτησα εδώ στο Βόλο που πάντα βρίσκονταν δίπλα μου για να με στηρίξουν όχι μόνο στην εκπόνηση της Διπλωματικής μου εργασίας, αλλά καθ'όλη τη διάρκεια της φοίτησης μου στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών.

Αφιερωμένο στους γονείς μου, Νώντα και Ελένη.

Και στην αδερφή μου, Γεωργία.

CONTENTS

ΠΕΡΙΛΗΨΗ

Η πρόθεση πίσω από την παρούσα διπλωματική εργασία, είναι η ανάπτυξη ενός εργαλείου ικανού, να χαρακτηρίζει με online τρόπο την κατανάλωση ενέργειας των οθονών των κινητών συσκευών. Προϋπόθεση είναι η μέγιστη ακρίβεια αυτού του εργαλείου στις μετρήσεις, καθώς και η απουσία ιδιαίτερων τεχνικών γνώσεων κατά την χρήση του. Στόχος είναι ο μέσος χρήστης να μπορεί να χρησιμοποιεί αυτό το εργαλείο στην καθημερινότητά του και να μην ανατρέχει στο εργαστήριο για να μετρήσει την κατανάλωση της κινητής του συσκευής. Η υλοποίηση μας, θα απομονώνει τα ιδιαίτερα χαρακτηριστικά της οθόνης που κυριαρχούν στην κατανάλωση ενέργειας. Στη συνέχεια, θα επιστρέφει στον χρήστη μία αντιστοίχιση μετρήσεων ενέργειας με τα χαρακτηριστικά της οθόνης του. Τέλος, θα προτείνει λύσεις για την εξοικονόμηση ενέργειας της συσκευής, βασισμένες στα πορίσματα που προέκυψαν από τις συγκεκριμένες μετρήσεις.

ABSTRACT

The motivation behind this thesis is the development of a tool that is capable of the online power consumption characterization of mobile phone displays. This tool would be accurate and would not require technical knowledge or lab experiments, in order to measure the power consumption of the mobile devices. Our implementation will identify the display factors which affect the power consumption and will return to the user an energy consumption mapping. This mapping, could be used as feedback to users, in order to isolate the unique display parameters that dominate their mobile's energy consumption. In the end, our tool would offer specific solutions and hints, on how the user can reduce the consumption and maximize the energy efficiency.

1. Introduction

Mobile consumer-electronics devices, especially phones, are powered from batteries which are limited in size and therefore capacity. This implies that energy efficiency is very important and crucial to these mobile devices usability. At the same time, device capabilities are increasing rapidly. Modern high-end mobile phones combine the functionality of a pocket-sized communication device with PC-like performances. Users are now capable of voice communication, audio and video playback, web browsing, SMS and email communication, media downloads, gaming and more. This kind of rich functionality emphasizes the need for effective energy management.

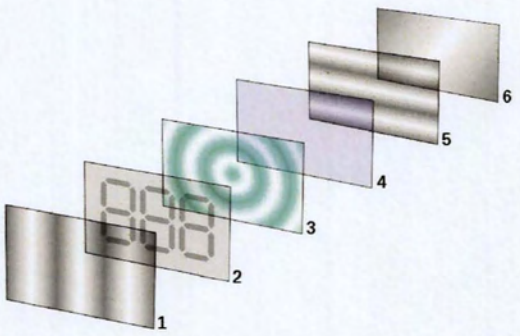
The effective energy management requires a good understanding of where and how the energy is used. As the type of the display technology is known to be among the largest power-consuming components on a modern mobile device, we begin this diploma thesis with a brief presentation of the most known display technologies. Their operation is presented without insisting on much detailed information in chapter 2. We continue with the third chapter: our initial energy measurements. The problems we faced lead us to the implementation of our own task killer, which we describe in chapter 4. In chapter 5, we analyzed the energy saving approaches we followed. The conclusions of each approach and the energy results of various coloring scenarios, are presented. In chapter 6, we proceeded with our main implementation: the online tool for power consumption characterization of mobile phone displays. An introduction to the Android environment is discussed, followed by the tool implementation details. At the end, final conclusions are made and various solutions and actions are offered.

2. Mobile Screen Displays

Screen of a mobile phone or a smartphone is one of the key features, users concern when buying a new mobile device. It's the main hardware that, user can interact with the embedded software and do various things. Generation to generation the mobile screen has been evolved and today humans reached the milestone of making flexible screens. There are so many types of screens out there with different technologies and pixel arrangements. Initially, at the earlier days we had dot matrix screens and later we managed to make them display colors and now time has taken us to the era of touchscreens where screens are capable of showing crisp images and colors.

2.1 LCD

Liquid-crystal displays (LCD) are flat panel displays that use the light modulating properties of liquid crystals. Liquid crystals do not emit light directly. Each pixel of an LCD typically consists of a layer of molecules aligned between two transparent electrodes and two polarizing filters. Without the liquid crystal between the polarizing filters, light passing through the first filter would be blocked by the second polarizer **[1].**



Reflective twisted nematic liquid crystal display.

1. ***Polarizing filter*** (vertical axis) to polarize light as it enters.
2. Glass substrate with ***electrodes*** (vertical ridges). The shapes of these electrodes will determine the shapes that will appear when the LCD is turned ON.
3. ***Twisted nematic liquid crystal***.
4. Glass substrate with ***electrodes*** (horizontal ridges) to line up with the horizontal filter.
5. Polarizing filter with horizontal axis to block/pass light.
6. Reflective surface to send light back to viewer or a ***light source***.

2.2 TFT – LCD

The TFT – LCD screens, which stands for ***Thin Film Transistor Liquid Crystal Display*** technology are the most common screen types and used in entry level to high end smartphones. Unlike the normal LCD panels, TFT LCD panels give better image quality and also support higher resolutions.

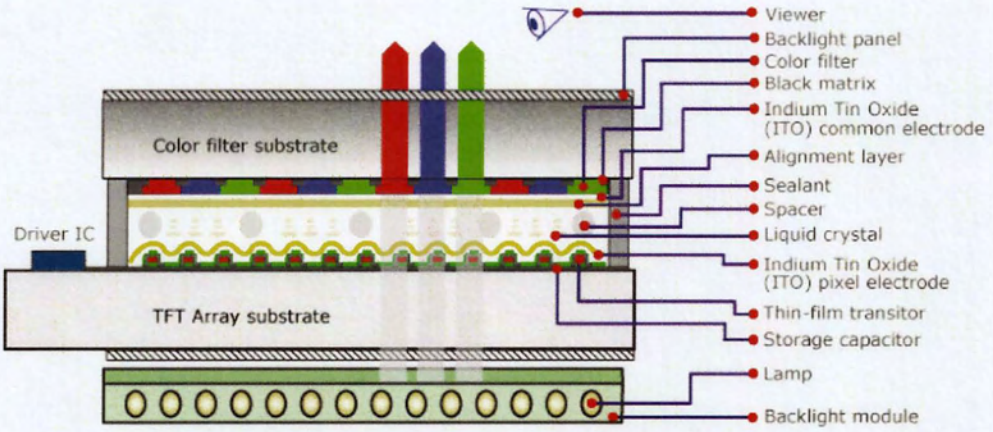
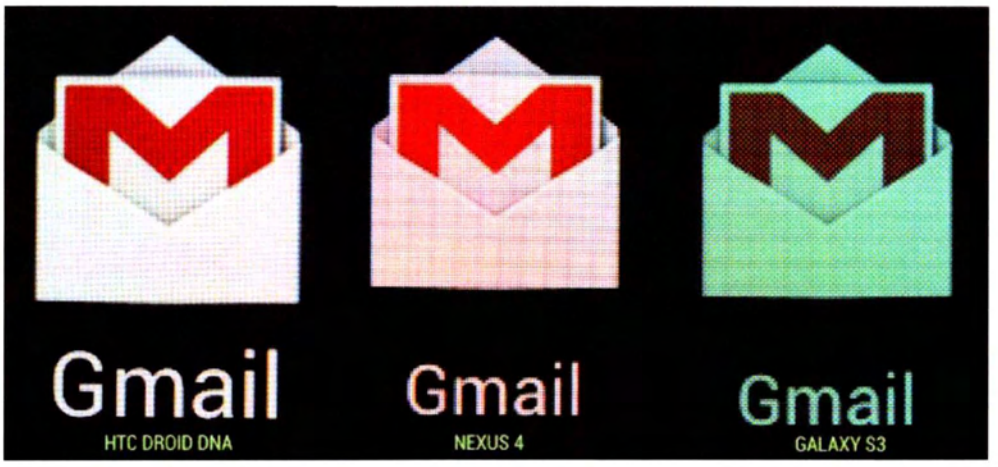


Diagram 1: Structure of TFT-LCD

As we can see in the structure above, liquid crystals are placed between two Indium Tin Oxide Pixel electrodes. The light is emitted from the backlight panel and goes through liquid crystals and bright up the pixels. However TFT-LCD screens consume a lot of energy as they require external lighting. Since TFT – LCD screens are cheaper to manufacture most smartphone manufacturers tend to use these panels. Viewing angles are not the best and one more disadvantage is the poor visibility in direct sunlight.

2.3 Super LCD

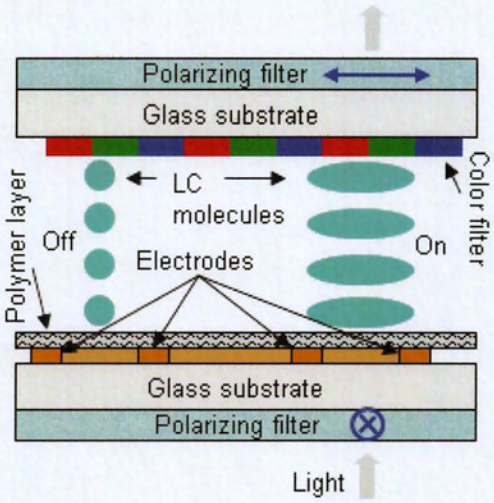
Super LCD screens are another type of TFT LCD screens. However they are specially improved for low power consumption without reducing image quality and they don't have an air gap between display element and outer glass like in ordinary TFT LCD panels. This makes the colors more accurate and reduce glare. In fact Super LCD panels give more color definition than AMOLED screens **[2]**.



We can see the performance of Super LCD 3 screen of HTC droid DNA with LG nexus 4 and Samsung Galaxy SIII. Note that Droid DNA has a 1080p screen while the other two have 720p.

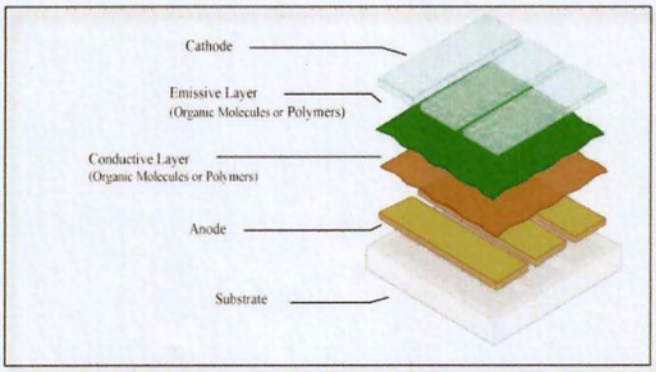
2.4 IPS – LCD

IPS LCD stands for ***In Plane Switching Liquid Crystal Display.*** This is another great improvement for LCD screens. The technology used here is arranging and switching Liquid Crystal molecules between the glass substrates in a plane, parallel to the glass layers. IPS panels do a great job when it comes to viewing angles **[3]**. Unlike TFT screens, IPS displays are capable of wide viewing angles and less power consumption.



2.5 OLED

OLED screens made another milestone in Screen technology. It stands for ***Organic Light Emitting Diode***, a totally different and impressive screen technology when compared to normal LCD screens **[4]**. This technology uses organic molecules to produce their electrons and holes. A simple OLED is made up of six different layers. On the top and bottom there are layers of protective **glass** or **plastic**. The top layer is called the **seal** and the bottom layer the **substrate.** In between those layers, there's a **negative terminal** (sometimes called the cathode) and a **positive terminal** (called the anode). Finally, in between the anode and cathode are two layers made from organic molecules called the **emissive layer** (where the light is produced, which is next to the cathode) and the **conductive layer** (next to the anode).



2.5.1 How an OLED emits light in 4 steps

1. To make an OLED light up, we simply attach a voltage (potential difference) across the anode and cathode.
2. As the electricity starts to flow, the cathode receives electrons from the power source and the anode loses them (or it "receives holes," if you prefer to look at it that way).
3. Now we have a situation where the added electrons are making the emissive layer negatively charged, while the conductive layer is becoming positively charged.
4. Positive holes are much more mobile than negative electrons so they jump across the boundary from the conductive layer to the emissive layer. When a hole (a lack of electron) meets an electron, the two things cancel out and release a brief burst of energy in the form of a particle of light—a **photon**, in other words. This process is called **recombination**, and because it's happening many times a second the OLED produces continuous light for as long as the current keeps flowing.

2.5.2 Advantages and Disadvantages of OLEDs

OLEDs are superior to LCDs in many ways. Their biggest advantage is that they are much thinner. Specifically around 0.2–0.3 mm or about 8 thousandths of an inch, compared to LCDs, which are typically at least 10 times thicker and consequently lighter and much more flexible. They are brighter and need ***no backlight***, so they consume much less energy than LCD displays. This translates into longer **battery** life in portable devices. LCDs are relatively slow to refresh. A problem when it comes to fast-moving pictures such as sports on TV or computer games, OLEDs respond up to 200 times faster. They produce truer colors and a true black, through a much bigger viewing angle **[5]**.

As for drawbacks, one widely cited problem is that OLED displays don't last as long: degradation of the organic molecules meant that early versions of OLEDs tended to wear out around four times faster than conventional LCDs or LED displays. Manufacturers have been working hard to address this and it's much less of a problem than it used to be. Another difficulty is that organic molecules in OLEDs are very sensitive to **water**. Though that shouldn't be a problem for domestic products such as TV sets and home computers, it might present more of a challenge in portable products such as cellphones.

Super AMOLED and ***Super AMOLED Plus*** are marketing terms for variants of standard OLED technology. These displays may have small variations such as a different pattern of red, green, and blue sub-pixels, but fundamentally they work the same as other AMOLED displays **[6][7][8]**.

3. Initial Measurements: Motivation & Background

Initially, our goal behind the Energy measurements was to confirm the different behavior in power consumption between LCDs and OLEDs displays. We expected that in LCDs, where the backlight is dominant, the power consumption to be only affected by the lighting intensity regardless the RGB color of the pixels. On the contrary, in an OLED display which does not require a light source as its pixels are emissive, we anticipated the color of the pixel to directly impact the power consumption. While OLED displays consume close to zero when presenting a black screen, they are much less efficient than LCDs in presenting certain colors, in particular white **[9]**.

Energy measurements were implemented on mobile devices with their display screen attached. Among all the components of the mobile phone, screen is considered to be the most power thirsty. Our objective was to make accurate conclusions about the energy behavior of these two types of screen technology with this specific set up: out of the box devices with their display attached.

A huge amount (> 50 %) of consumed energy is usually related to the operation of mobile phone displays!

Existing battery technologies cannot meet the increased energy demands (battery capacity limits double every 10 years)

3.1 Testing Devices

For our energy measurements we used two mobile devices. Each one supports the desired display technology:

1. Sony Xperia Miro
2. Samsung Galaxy Nexus I9250

Sony Xperia Miro

Display: **Type** TFT-LCD, 16M colors

Features:

OS: Android OS, v4.0 (Ice Cream Sandwich)

Chipset: Qualcomm MSM7225A

CPU: 800 MHz Cortex-A5

GPU: Adreno 200

Samsung Galaxy Nexus

Display: **Type** Super Amoled, 16M colors

Features:

OS: Android OS v4.0 (Ice Cream Sandwich)

Chipset: TI OMAP 4460

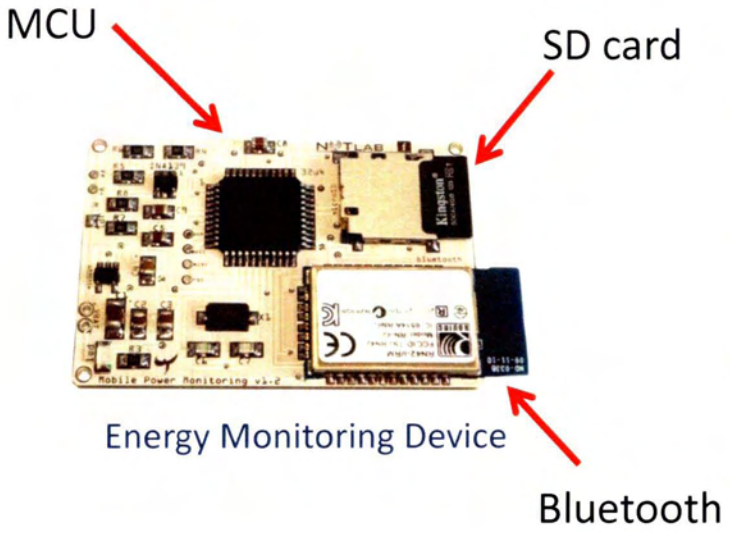
CPU: Dual-core 1.2 GHz Cortex-A9

GPU: PowerVR SGX540

The devices were rooted for the purpose of our experiments. Android, as an open-source platform was the OS we really needed to use.

3.2 Mobile Monitoring Solution (MMS)

For the purpose of Energy Measurements in Mobile Devices, NITlab has developed a Mobile Monitoring Solution called MMS board **[10]**. NITOS MMS board enables online power consumption of portable devices through the integration of a tiny custom-designed board, in order to provide energy efficiency evaluation under realistic mobile scenarios.



The MMS board is Arduino based. **Arduino** is a single-board microcontroller, intended to make building interactive objects or environments more accessible. Introduced in 2005, the Arduino designers sought to provide an inexpensive and easy way for hobbyists, students, and professionals to create devices that interact with their environment using **sensors** and **actuators**. Common examples for beginner hobbyists include simple **robots**, **thermostats** and motion detectors. It comes with a simple **integrated development environment** (IDE) that runs on regular personal computers and allows users to write programs for Arduino using **C or C++.**

"Arduino Uno" Revision 3

Type: Single-board microcontroller

Arduino Software IDE

A screenshot of the Arduino IDE showing the "Blink" program, a simple beginner program.

The MMS board responds to a specific API that is implemented in the Arduino integrated development environment (IDE) **[11]**. **It supports two basic commands**:

- **Sample:** samples=(time in sec),name=(name of file, max=7 chars)
  - Sample command to set the MMS board in Energy sensing mode, specification of the sensing time and of the file name.
  - The Energy measurement is stored in SD card of the board (see below) in Binary file.
- **Send:** send=(name of file, max=7 chars)
  - Send command that sends the Binary file from the SD card location to the specified device (see below).

The MMS board is consisted of **three basic components:**

- **Micro Controller (MCU):**
  - It uses an 8-bit Micro Controller (MCU), which runs at 8MHz.
  - This MCU is capable of high sampling rate of about 17 KHz

- Offers a 10-bit resolution.
- It is also distinguished for the low power consumption of approximately 20mA.

- **Bluetooth Module:**
  - Fully qualified Bluetooth 2.1/2.0/1.2/1.1 module
  - Bluetooth v2.0 + EDR support, a faster PSK modulation scheme capable of transmitting data 2 or 3 times faster than previous versions of Bluetooth.
  - Enhanced data rate: 2.1 Mbits /sec
  - Baud rate speed: 1200bps up to 921Kbps, non-standard baud rates can be programmed.
  - Serial communication with MCU: 2400 – 115200bps.
  - Bluetooth Services are used for the communication with the mobile device in which the MMS board is attached.

- **SD card:**
  - After the sensing period, a Binary file which contains the actual energy measurements is stored in the SD card location.
  - Bluetooth Services are also used for the file transfer from the SD card location to the desired location in the Mobile device.

Monitoring Device deployed on a smart-phone

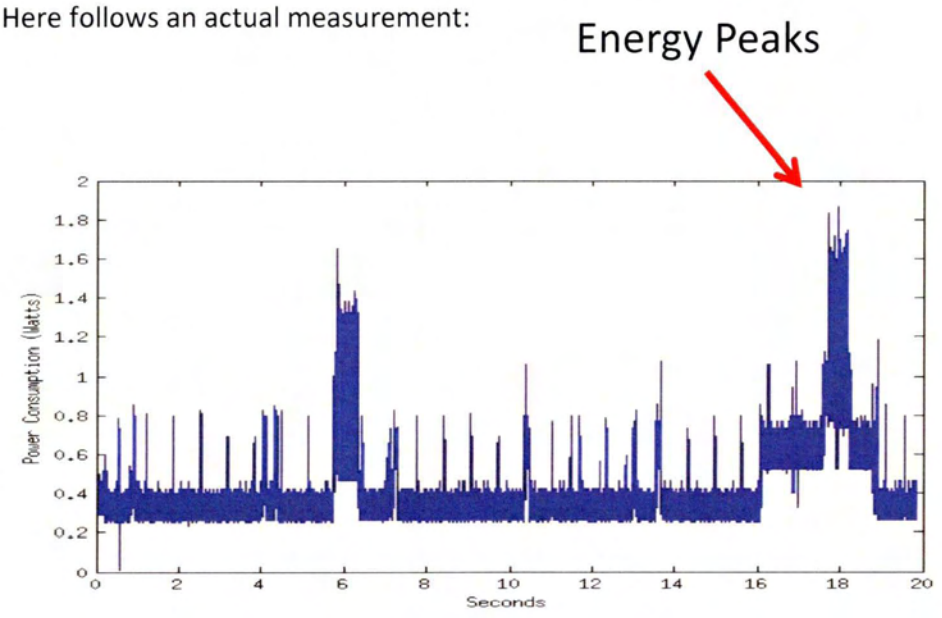
MMS board attached in the mobile device and ready for energy sensing.

3.3 First Energy Results

In our scenario, the display screens are attached on the mobile devices when we measure the energy consumption. As a result, we want to achieve the **maximum accuracy** in order to isolate the display power consumption from the OS services.

At our first attempts, we just measured the power consumption of the mobile devices in order to observe the Energy behavior of our testing devices.

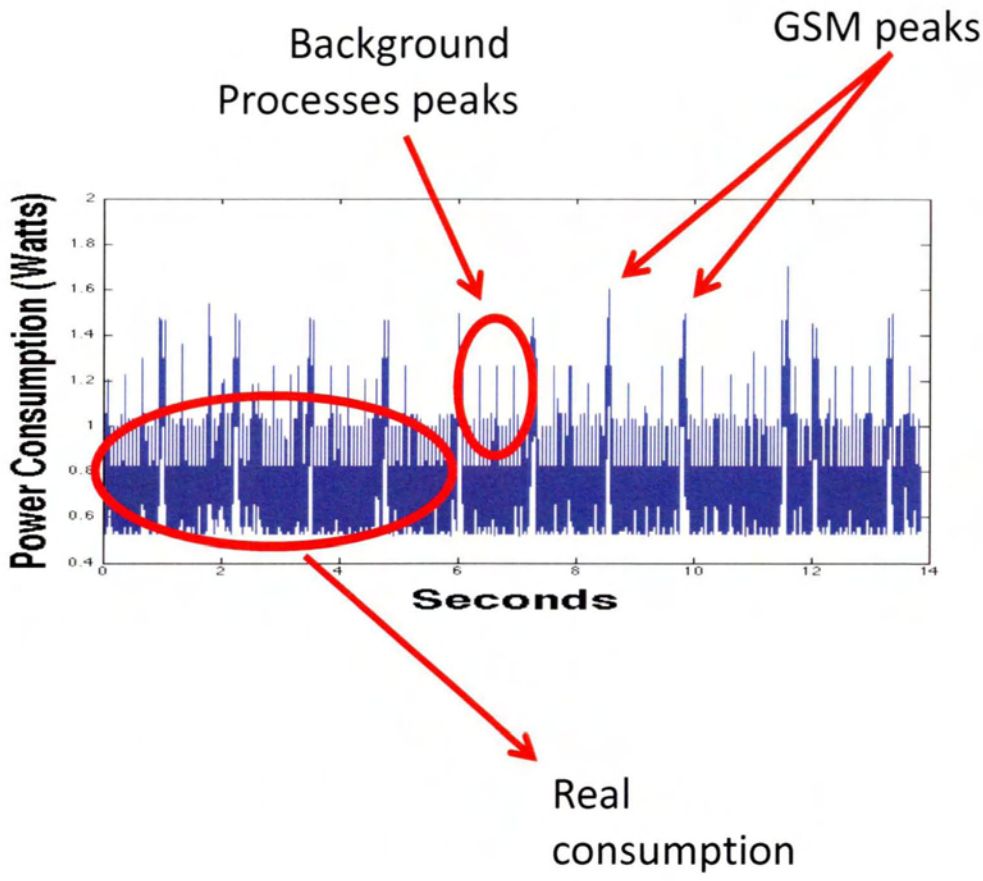
Here follows an actual measurement:



3.4 Energy Fluctuations Problem

From our first attempts we realized that the energy peaks we were experiencing were increasing the average energy consumption. They didn't lead us to safe conclusions about the Display Energy Consumption. Assumptions about the Energy behavior of the screens were inconclusive at that point. So, energy fluctuations related with other device components have to be avoided.

These energy fluctuations were mainly caused, by the communication interfaces like GSM, WiFi etc, the OS was running. The operating system of the mobile devices runs background processes in order to operate or awakes other processes for its needs. We should also take in regard that the device interface and GUI (Graphical User Interface) trigger services for better user experience.



4. Design of a Task Killer

We wanted to eliminate these Energy peaks. For this purpose we developed our own Task Killer. But why our own task killer, when the Play Store is full of Task Killers and Task Managers?

Task Killers found on stores **[12]**:

- Don't have root and Super User (SU) privileges.
- They can't really kill a process – only processes of their application package.
- They just move the processes of an application package from android stack.
- But most of them start right back up, further draining the CPU and cause energy peaks.

When an android application starts, it creates a package and starts its own services inside that package. A service or a process inside an application package cannot kill

or end a service/process from another application package. It can kill only services that are included in the same application package. That kind of action, killing a service from another package, requires Super User privileges. That's why our testing devices are rooted and this is the reason we created our own task killer.

4.1 Task Killer Implementation

The development of our own task killer was based in the following actions:

- Identification of the really basic processes that OS needs to operate.
- Activation of Airplane Mode to disable communication interfaces and GPS.
- Kill all other processes.

Task Killer implementation was an android application. With our rooted mobile devices and our Super User privileges we were able to discover all the running packages with their services and processes. From that point, we continued by eliminating packages in order to identify the absolute minimum number of processes the OS need to operate.

This approach led us to some constraints:

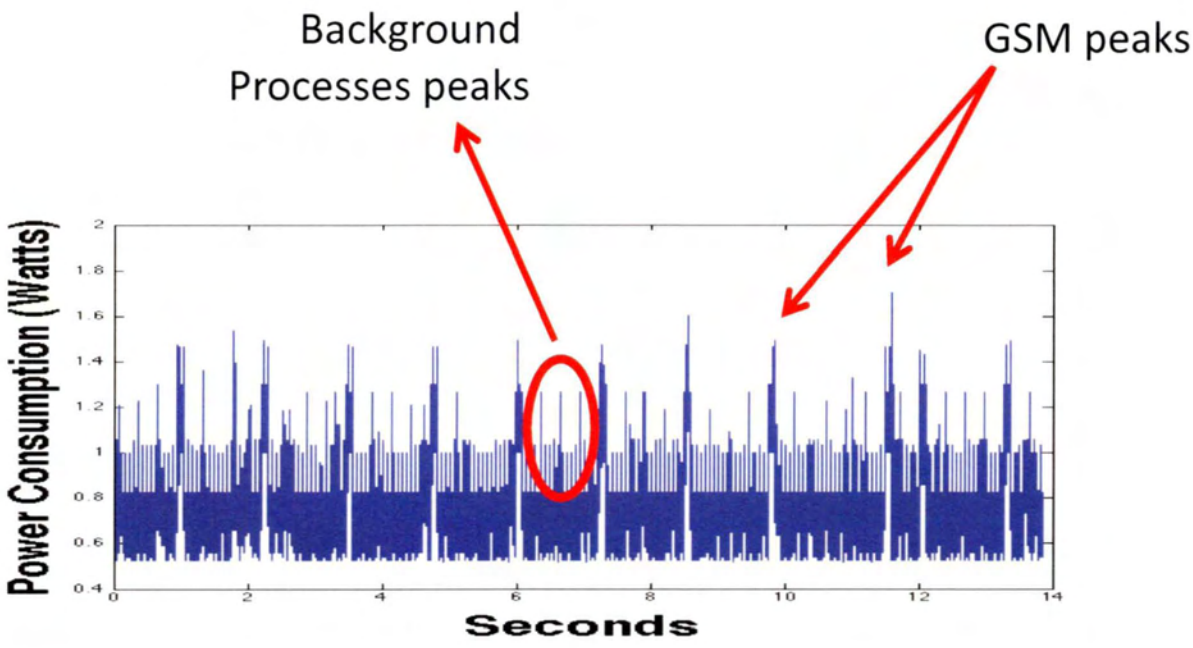
- Different basic processes for different devices.
- Different Task Killers for different devices.

As a result, we created two device specific task killers. One for the Sony Xperia Miro and another one for the Samsung Galaxy Nexus. The Sony mobile device had its own user interface on top of the android stack. That resulted in more energy peaks as some processes were refusing to end their "lives". On the Samsung mobile device, things were more straightforward. It had "clean" android stack and the energy peaks we experienced were more controlled and expected.

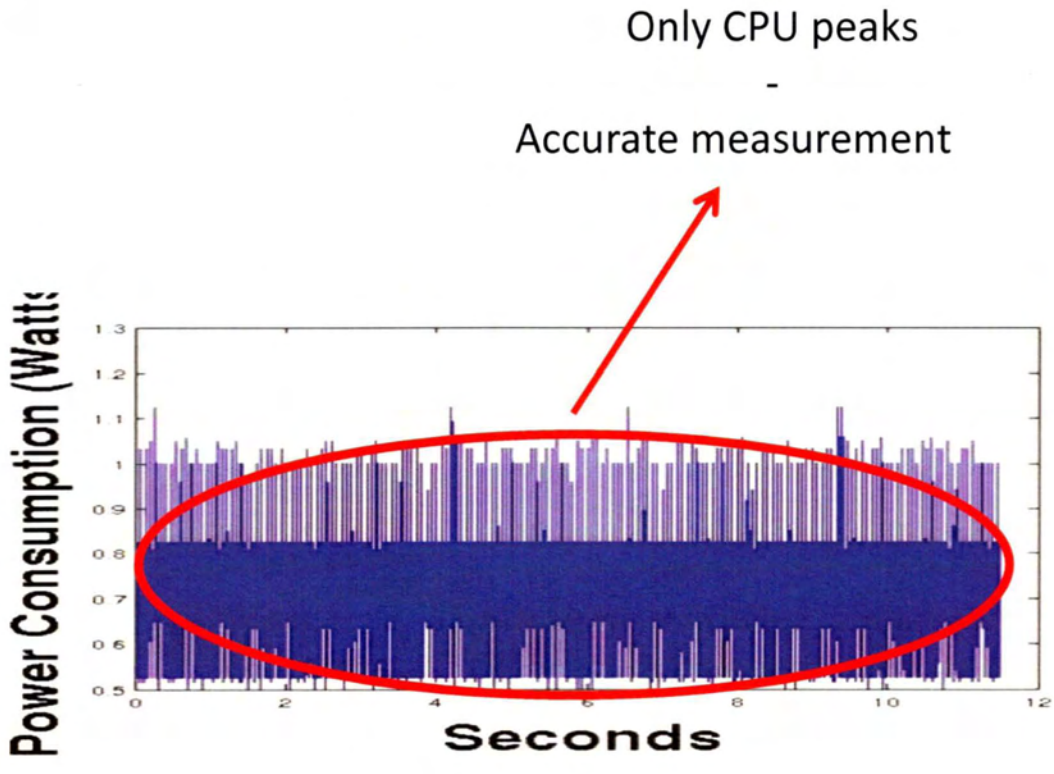
After the implementation of the Task Killer we had a tool which could guarantee the maximum accuracy of our energy measurements and we could continue from that point with our evaluation of the energy behavior of the mobile screens.

4.2 Task Killer Results

Before:



After:



5. Energy Saving Approaches

After the implementation of the Task Killer, we are at a point where we can **accurately** measure the energy consumption. So, we are going to evaluate two energy approaches based on the motivation we discussed in chapter 3:

- Adjusting brightness
- Adjusting color presentation

5.1 First Approach: Adjusting Brightness

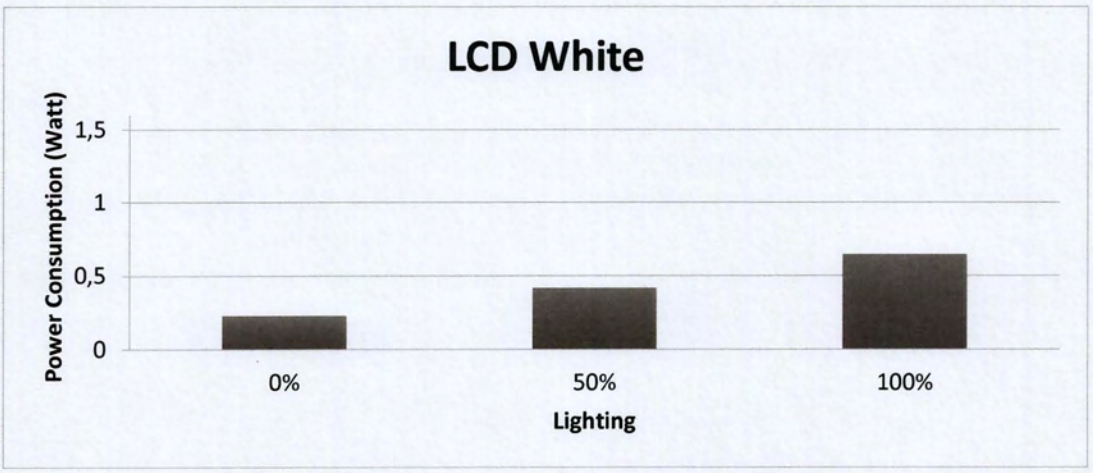
At first, we measured the white color energy consumption by painting the whole screen and in the maximum alpha value.

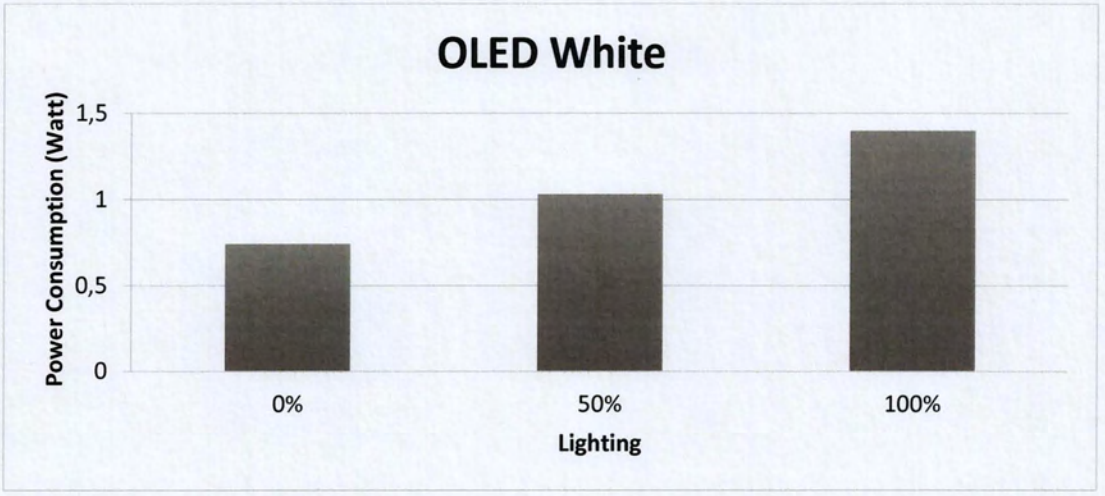
LCD

Sony Ericsson Miro

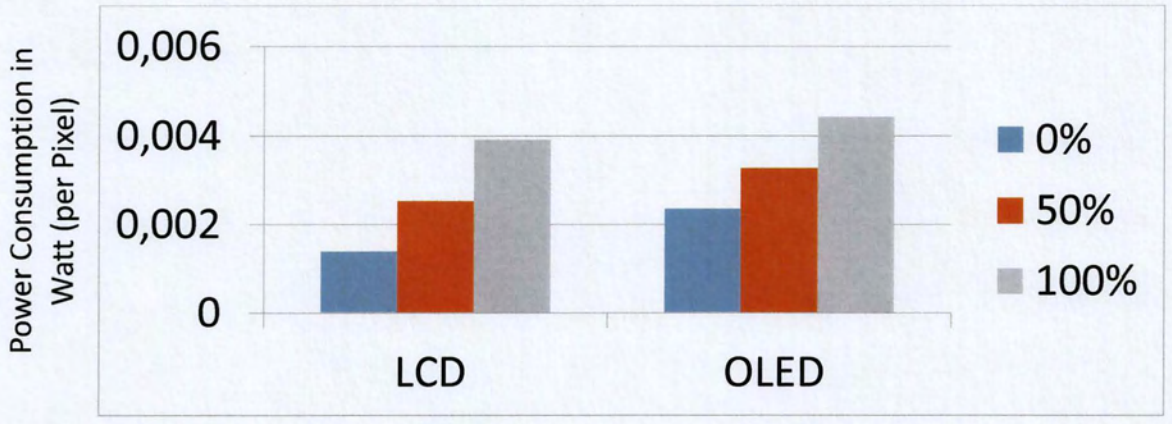
OLED

Samsung Nexus



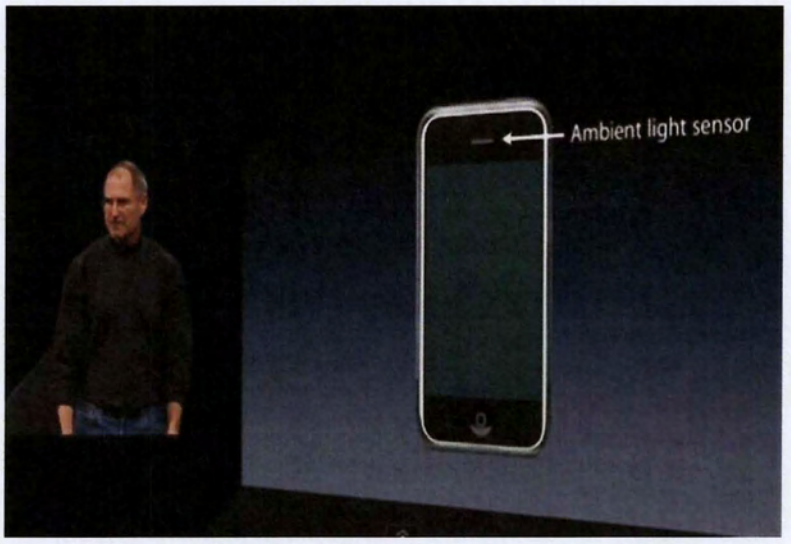


From the diagrams above we can assume that the **brightness affects the power consumption on both types of display screens.** The more bright, the more energy is consumed. But, as our devices have different dimensions and a different pixel density, we can't conclude which display technology is affected the most. Sony Xperia Miro has a 3.5 inches LCD display with a 165 ppi pixel density and the Samsung Galaxy Nexus has a 4.65 inches AMOLED display with a 316 ppi pixel density. **The following diagrams show the power consumption per pixel [13]:**



In the LCD display by decreasing the brightness from 100% to 0% we achieved a 65% reduction in energy consumption. In the AMOLED display, for the same scenario we achieved a 48 % reduction in power consumption. We can now conclude, that **LCD displays are more affected by the brightness level than the OLED displays.**

The observation, that the brightness level affects the power consumption of all types of display screens, was made from the early days of the smartphones. For this reason, an ambient light sensor is incorporated in most Mobile Devices and is responsible for the continuous adaptation of screen lighting to preserve battery life.

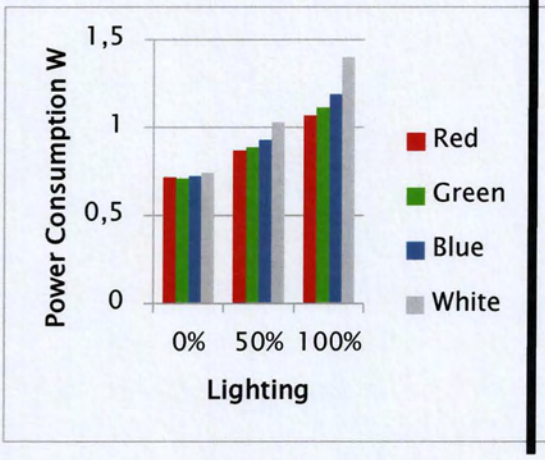


Ambient light sensor attached at the first breed of iphones.

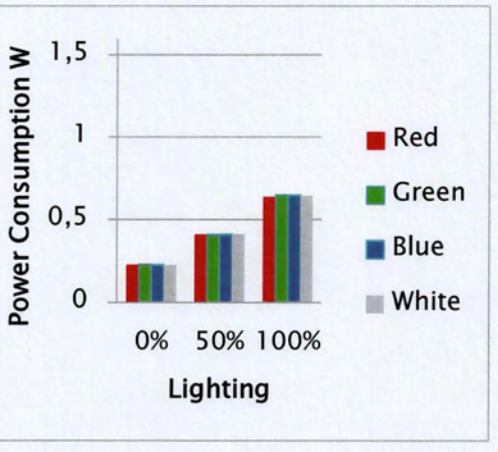
5.2 Second Approach: Adjusting Color

As OLEDs display screens seem to be affected less from the brightness level than the LCDs, we started to display different colors from white in the highest alpha value, in order to observe their energy consumption behavior.

OLED display:



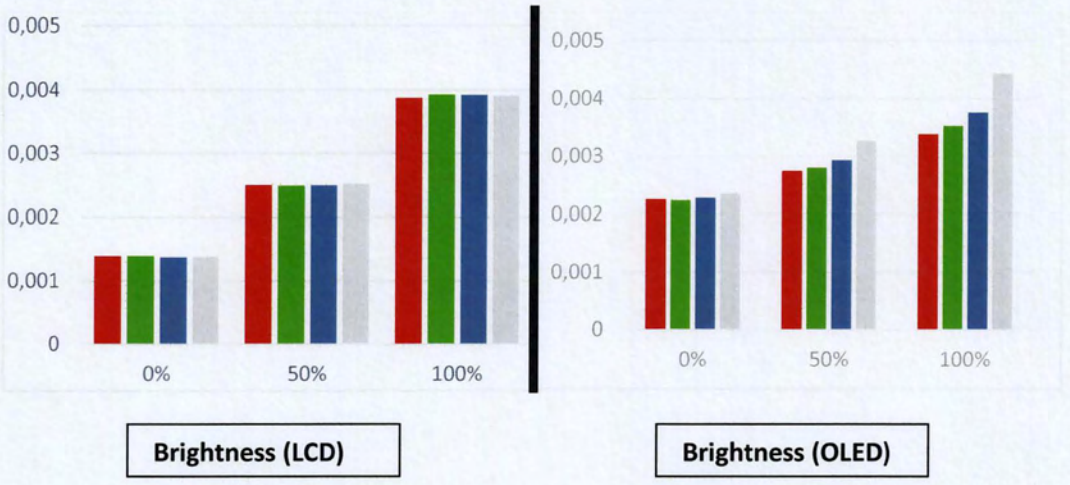
LCD display:



From the above diagrams, we can observe that the **OLED display consumption is also affected by the presenting color except the lighting level.** The OLED screen seem to have less power consumption in certain colors in the same brightness level. In contrast, in the LCD display the dominant factor of the energy consumption is the brightness level and the color presentation is not a major factor in power consumption. Colors such as red and green, appear to be more energy friendly in the

OLED display, than blue or white. A behavior that is not presented in the LCD display as the decisive energy factor is the backlight source.

In order to evaluate better this behavior, we calculate the **per pixel consumption** as the dimension and density of the screens were different:



The conclusions from per pixel consumptions are the same. LCDs energy behavior is depended on the lighting level and only a small portion of the energy consumption is affected by the presenting color. OLEDs energy behavior depends on the lighting level but also on the color of the presenting pixels.

5.3 Identifying the Most Efficient Color in OLED

As it is mentioned before, OLEDs displays seem to prefer certain colors and to behave more energy efficient at presenting them than others. As light intensity in OLED displays is not a decisive factor in energy consumption, we proceeded to the following measurements by setting lighting to 100% and adjusting the "alpha" value of each color. To achieve better correlation between our measurements and reality, we set a background and on top of that we started to switch between the various presented colors **[14]**.

Default google background that we used in our scenarios.

Black color behavior

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2519W	100	0.9022W	200	0.6723W
10	1.2040W	110	0.8705W	210	0.6557W
20	1.1684W	120	0.8425W	220	0.6476W
30	1.1250W	130	0.8163W	230	0.6358W
40	1.0916W	140	0.7893W	240	0.6290W
50	1.0590W	150	0.7629W	255	0.5965W
60	1.0246W	160	0.7371W		
70	0.9992W	170	0.7167W		
80	0.9704W	180	0.7012W		
90	0.9366W	190	0.6823W		

By presenting black pixels we achieved the minimum power consumption. While increasing the alpha value, energy consumption was decreasing. Beside this goal, we were also interested in user's better experience. As a result, we did not want to influence the usability and the functionality of the display screen. We observed that user's experience is not disturbed for alpha values up to 100. In fact, for this alpha

range up to 100, power consumption is less than 1 Watt. This is very encouraging as our testing device was equipped with a 4.65 inches display screen.

White color behavior

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2360W	110	1.2575W	220	1.3210W
10	1.2332W	120	1.2619W	230	1.3261W
20	1.2327W	130	1.2666W	240	1.3336W
30	1.2492W	140	1.2639W	255	1.3463 W
40	1.2488W	150	1.2814W		
50	1.2509W	160	1.2803W		
60	1.2513W	170	1.2874W		
70	1.2534W	180	1.2922W		
80	1.2522W	190	1.2989W		
90	1.2545W	200	1.3125W		
100	1.2574W	210	1.3141W		

The white presentation was the worst case in energy efficiency and also in user's functionality. It caused blur (θάμπωμα) to the screen and for the same alpha's values as in black pixels, the result was not pleasant. As expected, by increasing the intensity of the white color, the power consumption was also rising.

Red color behavior

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2449W	110	1.0421W	220	1.0191W
10	1.2160W	120	1.0336W	230	1.0260W
20	1.1894W	130	1.0270W	240	1.0305W
30	1.1665W	140	1.0252W	255	1.0603W
40	1.1466W	150	1.0234W		
50	1.1236W	160	1.0150W		
60	1.1059W	170	1.0134W		

70	1.0887W	180	1.0130W		
80	1.0717W	190	1.0118W		
90	1.0599W	200	1.0112W		
100	1.0514W	210	1.0143W		

Between red, green and blue display colors (RGB), the red one is presented as more energy friendly. It provides better user experience than white but worse than black. Its energy behavior is different than the previous two. By increasing the alpha value we reach a minimum but this minimum is not caused by the biggest alpha value. We noticed that for alpha 200 we achieved the smallest consumption of 1.0112W. Increasing this value causes a growth in consumption. This is the result of the background we are using. At the alpha value of 200, the combination of the background and the red color results in the smallest consumption. After this critical point, the consumption is stabilized at the red's color one, which is bigger.

Green color behavior

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2682W	110	1.0622W	220	1.0424W
10	1.2352W	120	1.0537W	230	1.0510W
20	1.2110W	130	1.0452W	240	1.0515W
30	1.1802W	140	1.0457W	255	1.08228W
40	1.1550W	150	1.0299W		
50	1.1357W	160	1.0254W		
60	1.1154W	170	1.0291W		
70	1.0985W	180	1.0327W		
80	1.0807W	190	1.0345W		
90	1.0789W	200	1.0351W		
100	1.0701W	210	1.0373W		

Similar behavior to red. It has a slightly bigger consumption than red but it's on display presentation is also slightly more pleasant. We measured the smallest consumption at the alpha value 160 and after this point the consumption is stabilized at the green's one.

Blue color behavior

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2457W	110	1.1118W	220	1.1321W
10	1.2292W	120	1.1059W	230	1.1403W
20	1.2063W	130	1.1080W	240	1.1501W
30	1.1891W	140	1.1094W	255	1.1650W
40	1.1701W	150	1.1197W		
50	1.1644W	160	1.1197W		
60	1.1509W	170	1.1144W		
70	1.1397W	180	1.1099W		
80	1.1239W	190	1.1152W		
90	1.1172W	200	1.1164W		
100	1.1130W	210	1.1247W		

Blue presentation consumes more than red and green. The behavior follows the red and green pattern. For alpha value of 120, it shows the smallest consumption of 1.1059 W. From users perspective it's bearable but not as black.

Yellow color behavior (Red=255, Green=255, Blue=0)

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2463W	110	1.1507W	220	1.2101W
10	1.2353W	120	1.1533W	230	1.2159W
20	1.2188W	130	1.1482W	240	1.2284W
30	1.2038W	140	1.1526W	255	1.2552W
40	1.1910W	150	1.1587W		
50	1.1835W	160	1.1617W		
60	1.1755W	170	1.1617W		
70	1.1658W	180	1.1740W		
80	1.1580W	190	1.1820W		
90	1.1595W	200	1.1912W		
100	1.1503W	210	1.1964W		

As it can be seen from above, we tried to evaluate the behavior of yellow as it is the combination of two basic colors of RGB with the smallest consumption, red and

green. Yellow's energy consumption is very bad and it is worse than blue's one. Understandably, it follows the same pattern as the previous ones.

Dim Gray color (Red=105, Green=105, Blue=105)

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2717W	110	1.0159W	220	0.8564W
10	1.2410W	120	0.9945W	230	0.8476W
20	1.2154W	130	0.9806W	240	0.8339W
30	1.1965W	140	0.9630W	255	0.8199W
40	1.1694W	150	0.9469W		
50	1.1448W	160	0.9344W		
60	1.1265W	170	0.9265W		
70	1.1091W	180	0.91331W		
80	1.0887W	190	0.8981W		
90	1.0695W	200	0.8791W		
100	1.0477W	210	0.8667W		

The basic indication from our measurements is that we should choose darker presentations. As a result, the next color we tested was Dim Gray which is close to black.

The conclusions from the above table are like the ones made from black measurements. The consumption is lower, than any other color except black. Also, its behavior is like blacks. By increasing the alpha values the consumption is dropping and it reaches its minimum at the biggest alpha value. It is also pleasant to the user.

Dark Gray (Red=64, Green=64, Blue=64)

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2377W	110	0.9569W	220	0.7537W
10	1.2005W	120	0.9346W	230	0.7365W

20	1.1674W	130	0.9087W	240	0.7312W
30	1.1382W	140	0.8887W	255	0.6994W
40	1.1131W	150	0.8662W		
50	1.0875W	160	0.8505W		
60	1.0653W	170	0.8316W		
70	1.0411W	180	0.8222W		
80	1.0293W	190	0.7996W		
90	1.0126W	200	0.7843W		
100	0.9852W	210	0.7679W		

From now on, we tend to use more black in our tested colors. It's really nice for the user and it has lower consumption than dark gray. As its behavior is concerned, it follows the pattern from black.

Dark Red (Red=139, Green=0, Blue=0)

Consumption is close to Dark Gray. Their behavior is close.

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2447W	110	0.9556W	220	0.7699W
10	1.2169W	120	0.9312W	230	0.7623W
20	1.1781W	130	0.9156W	240	0.7511W
30	1.1468W	140	0.8893W	255	0.7422W
40	1.1154W	150	0.8684W		
50	1.0959W	160	0.8475W		
60	1.0649W	170	0.8339W		
70	1.0533W	180	0.8174W		
80	1.0276W	190	0.8039W		
90	1.0013W	200	0.7922W		
100	0.9800W	210	0.7825W		

Dark Green

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2505W	110	0.9394W	220	0.7413W
10	1.2130W	120	0.9147W	230	0.7303W
20	1.1767W	130	0.8921W	240	0.7174W
30	1.1470W	140	0.8680W	255	0.7105W
40	1.1119W	150	0.8489W		
50	1.0860W	160	0.8337W		
60	1.0593W	170	0.8175W		
70	1.0303W	180	0.7984W		
80	1.0055W	190	0.7842W		
90	0.9852W	200	0.7680W		
100	0.9609W	210	0.7556W		

The consumption was really low and close to the one of black. The user will not find any inconvenience on his daily experience. It is actually more user friendly than dark red shades.

More Dark Red color (Red=100, Green=0, Blue=0)

In this measurement we used the respective values from the Dark Green. Its consumption is lower than Dark Green but it is not so user friendly.

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2378W	110	0.9188W	220	0.7089W
10	1.1997W	120	0.8943W	230	0.7006W
20	1.1670W	130	0.8714W	240	0.6886W
30	1.1332W	140	0.8447W	255	0.6713W
40	1.1013W	150	0.8196W		
50	1.0753W	160	0.7982W		
60	1.0457W	170	0.7803W		
70	1.0186W	180	0.7620W		
80	0.9962W	190	0.7476W		
90	0.9714W	200	0.7325W		
100	0.9452W	210	0.7204W		

Dark Slate Gray color (Red=47, Green=79, Blue=79)

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2283W	110	0.9635W	220	0.7587W
10	1.1951W	120	0.9419W	230	0.7447W
20	1.1674W	130	0.9187W	240	0.7292W
30	1.1388W	140	0.9008W	255	0.7096W
40	1.1104W	150	0.8805W		
50	1.0935W	160	0.8614W		
60	1.0660W	170	0.8434W		
70	1.0453W	180	0.8266W		
80	1.0322W	190	0.8091W		
90	1.0089W	200	0.7929W		
100	0.9873W	210	0.7796W		

It is more user friendly than the previous one. On the other hand, as the three basic colors of the RGB contribute, it's more energy thirsty.

Midnight Blue color (Red=25, Green=25, Blue=112)

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2448W	110	0.9738W	220	0.7727W
10	1.2175W	120	0.9533W	230	0.7596W
20	1.1873W	130	0.9319W	240	0.7483W
30	1.1574W	140	0.9065W	255	0.7295W
40	1.1337W	150	0.8873W		
50	1.1048W	160	0.8640W		
60	1.0802W	170	08450W		
70	1.0574W	180	0.8267W		
80	1.0310W	190	0.8113W		
90	1.0153W	200	0.7924W		
100	0.9957W	210	0.7803W		

On top of the white background that we use, it has more power consumption. It is pleasant but not as the black.

Purple color (Red=128, Green=0, Blue=128)

It looks like the red but the consumption is high.

Alpha	Watt	Alpha	Watt	Alpha	Watt
0	1.2223W	110	1.0125W	220	0.8384W
10	1.2001W	120	0.9917W	230	0.8292W
20	1.1727W	130	0.9797W	240	0.8127W
30	1.1488W	140	0.9626W	255	0.8035W
40	1.1259W	150	0.9510W		
50	1.1133W	160	0.9239W		
60	1.0901W	170	0.9110W		
70	1.0740W	180	0.8901W		
80	1.0535W	190	0.8735W		
90	1.0431W	200	0.8595W		
100	1.0294W	210	0.8496W		

Synopsis of the energy behavior of the various colors in OLED:

White is the most energy thirsty color. In contrast black, is the most energy efficient. Red and green are also efficient but not as black. They are also good in user experience such as the black. The final conclusion from all these scenarios, is that when we are using an OLED display it's more energy efficient when we are presenting darker colors. The darker the display screen, the more efficient is.

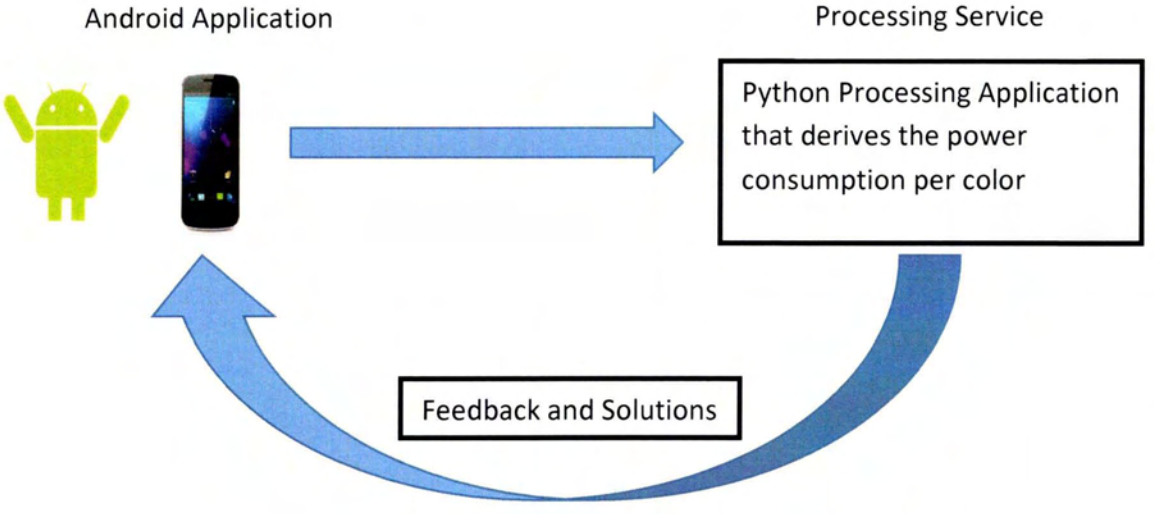
6. Online Tool for Power Characterization of the Display

Our goal is the implementation of a tool, capable of the online power characterization of the display screen of the mobile device. We want the user to be able to identify, in a short time, the display parameters that affect the energy consumption of his mobile device. In the end, the user will be able to seek a solution for the specific parameters of his display screen and will reduce, in some degree, the power consumption of his device.

6.1 Implementation

Our approach for the implementation of the online tool consists of three basic parts:

- Android Application
- Processing Service
- Feedback and Solutions

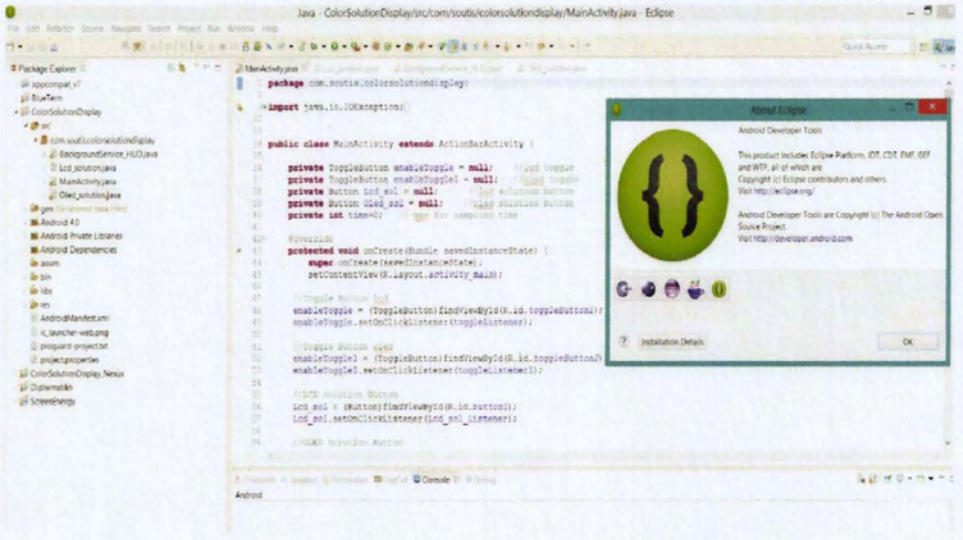


6.2 Introduction to Eclipse

We developed our online tool on the Android platform as it is open source, it is written in Java, which we are familiar with and is mobile device independent. The only requirement is, our device to support Android OS 3.0 platform (Honeycomb) and beyond. Android OS 3.0 stands for API Level 11. The environment where we developed our app was **Eclipse IDE with Android Development Tools (ADT) [15].**

Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications. By means of various plug-ins, Eclipse may also be used to develop applications in other languages: Ada, ABAP, C, C++, COBOL, Fortran, **Haskell**, JavaScript, Lasso,Natural, Perl, PHP, Prolog, Python, **R**, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and **Erlang**. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

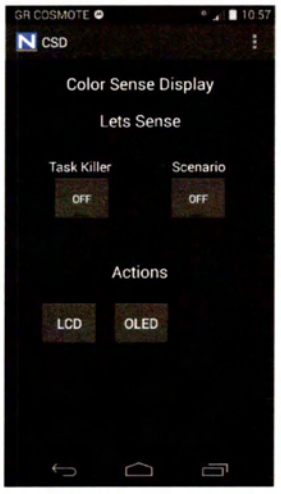
ADT extends the capabilities of Eclipse to let us quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug our applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute our application. Developing in Eclipse with ADT is highly recommended and is the fastest way to get started. With the guided project setup it provides, as well as tools integration, custom XML editors, and debug output pane, ADT gives us an incredible boost in developing Android applications.



Eclipse IDE with ADT plugin

6.3 Android Application

Android Application



Main actions:
1. Task Killer
2. Trigger MMS board sensing
3. Varying Color Test
4. Bluetooth transfer of the file from MMS board to Mobile Device

The Android App starts a Main Activity java class (MainActivity.class) which represents the main screen of the application **[16]**. This is what we see in the above picture. From this screen, the user is able to trigger various background services in order to online characterize the power consumption of his device. These various background actions will be discussed below.

Actions of the android app:

1. Task Killer:

As we have mentioned in chapter 3.5, the energy fluctuations problem which was caused mainly by the background services, was solved by the implementation of our own task killer. For that reason, a toggle button (initial state OFF) triggers the background service of the task manager. The development of the Task Killer was discussed on chapter 4.1. After the end of the Task Killer process, toggle button returns to its initial state (OFF) and the user is informed that the background service is stopped. After that, the user can continue to the next actions as his mobile device is stabilized to the minimum number of operational processes and the energy fluctuations are now avoided.

2. MMS board sensing:

Before starting changing the displayed colors, we must trigger the MMS board to start sensing the energy consumption. The MMS board, as we have mentioned in

chapter 3.3, is attached to the mobile device. By pressing the toggle button "Scenario", our mobile device starts a background service which initially enables the Bluetooth service and connects to the MMS board. After the successful Bluetooth connection, our application sends a specific command which respond to the implemented API in the MCU of the MMS board:

- **Sample command**: as it is mentioned in chapter 3.3, our app, by sending this command, sets the MMS board in sensing mode. The energy consumption of the mobile device is measured and stored in a binary format file, into the SD Card location.

Sampling duration of the MMS board is configured to 25 seconds. The reason behind this choice will be discussed in the following section of our application.

3. Varying Color Test:

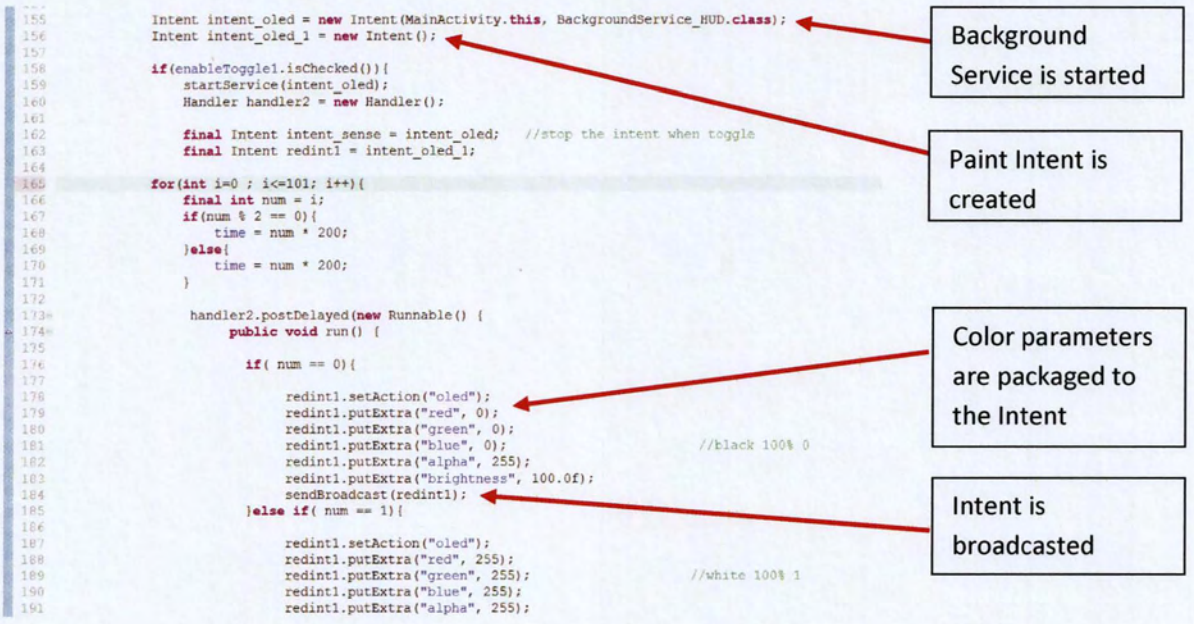
The same background service which triggers the MMS board sensing, is responsible for the varying color test. After the successful start of the sensing period, we are ready for the coloring sequence. The scenario we implemented, consisted of a 100 different color sequence. We wanted in a really short time to change 100 different colors and measure their power consumption.

6.3.1 Implementation of coloring sequence

The main activity starts the background service. At this time the application consists of two processes. We seriously considered the number of processes the application would use, because we did not want our implementation to use too many resources and services of the mobile device. We did not want to create a tool that targets the energy consumption minimization, but on the other hand the tool itself to be energy thirsty. The more efficient way we found was**, having the main activity responsible to notify in time intervals, with broadcast intents, the background service to paint the screen [17]**. In this way we only use two services.

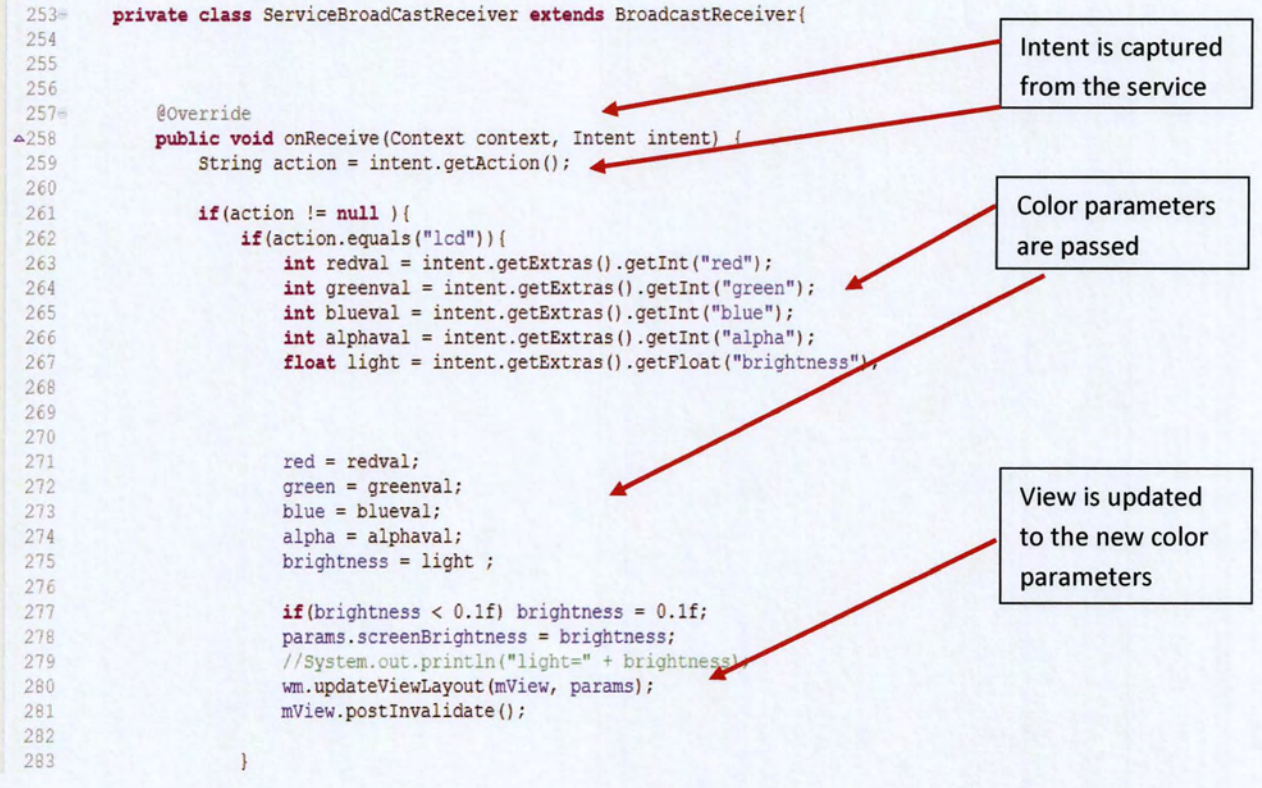
We statically set the color sequence that would be broadcasted to the service and then we had to **decide the time** each color would be painted on the mobile screen. We had to identify the smallest time we would need in order to measure the energy consumption with the maximum accuracy. The MMS board offers a high sampling rate of **17 KHz** which is very promising. This sampling rate means 1700 energy measurements in a time interval of 100 ms. Having this in consideration, a **first approach was an interval of 100 ms for each color** which seemed a really small duration but also capable of giving us efficient measurements for each color.

The following screenshot shows how the main activity **starts the background service**, which is responsible for the MMS board triggering and the paint of the screen. It also shows how the paint **intents are broadcasted**.




```
155         Intent intent_oled = new Intent(MainActivity.this, BackgroundService_HUD.class);
156         Intent intent_oled_1 = new Intent();
157
158         if(enableToggle1.isChecked()){
159             startService(intent_oled);
160             Handler handler2 = new Handler();
161
162             final Intent intent_sense = intent_oled;   //stop the intent when toggle
163             final Intent redint1 = intent_oled_1;
164
165         for(int i=0 ; i<=101; i++){
166             final int num = i;
167             if(num % 2 == 0){
168                 time = num * 200;
169             }else{
170                 time = num * 200;
171             }
172
173              handler2.postDelayed(new Runnable() {
174                  public void run() {
175
176                      if( num == 0){
177
178                          redint1.setAction("oled");
179                          redint1.putExtra("red", 0);
180                          redint1.putExtra("green", 0);
181                          redint1.putExtra("blue", 0);                    //black 100% 0
182                          redint1.putExtra("alpha", 255);
183                          redint1.putExtra("brightness", 100.0f);
184                          sendBroadcast(redint1);
185                      }else if( num == 1){
186
187                          redint1.setAction("oled");
188                          redint1.putExtra("red", 255);
189                          redint1.putExtra("green", 255);                 //white 100% 1
190                          redint1.putExtra("blue", 255);
191                          redint1.putExtra("alpha", 255);
```

The following screenshot shows how the broadcasted intent is received and how the color parameters are passed from one service to another.




```
250     * @author Vasilis
251     *
252     */
253    private class ServiceBroadCastReceiver extends BroadcastReceiver{
254
255
256
257        @Override
258        public void onReceive(Context context, Intent intent) {
259            String action = intent.getAction();
260
261            if(action != null ){
262                if(action.equals("lcd")){
263                    int redval = intent.getExtras().getInt("red");
264                    int greenval = intent.getExtras().getInt("green");
265                    int blueval = intent.getExtras().getInt("blue");
266                    int alphaval = intent.getExtras().getInt("alpha");
267                    float light = intent.getExtras().getFloat("brightness");
268
269
270
271                    red = redval;
272                    green = greenval;
273                    blue = blueval;
274                    alpha = alphaval;
275                    brightness = light ;
276
277                    if(brightness < 0.1f) brightness = 0.1f;
278                    params.screenBrightness = brightness;
279                    //System.out.println("light=" + brightness);
280                    wm.updateViewLayout(mView, params);
281                    mView.postInvalidate();
282
283                }
```

In the coloring scenario we implemented, we statically set the brightness level to 100%. We chose that level, because our intension was to exaggerate the influence of the specific characteristics of each type of screen on the power consumption.

From all the experience we gained from the energy measurements, we expected that the analysis of the LCD consumption would not be affected from the presenting color. In contrast, on the analysis of the OLED consumption we should be able to distinguish energy "steps", caused by the different presenting colors.

6.3.2 File transfer from MMS board to Mobile Device

At this point the binary file, that contains the energy measurements from our coloring sequence, is stored at the MMS board. In order to analyze it, we have to implement the process of transfer, from the MMS to our mobile device.

The operation of the file transferring via Bluetooth was challenging **[18]**. The process of Bluetooth connection between the devices was more straightforward. After a buffer stream was opened between the MMS board and the mobile device, we had to identify the **start** of the incoming file. For that reason, we used flags. A specific sequence of special characters marks the start of the incoming bytes. That technique was not the solution to the next challenge we faced: the identification of the **end** of the file. If we used a sequence of characters, any characters, to mark the end of the file, we could be unlucky and this sequence could really constitute an actual energy measurement. In this scenario, we are closing the incoming file and the MMS board is still sending energy measurements. For that reason, we chose the solution of **timestamps:** inactive incoming stream for explicit amount of time indicates the end of the file.

The next decision we had to make was the location of the incoming file on the mobile side. As the mobile devices don't support MatLab or Python services for the analysis of the file, we chose the Dropbox path for the internal storage. Having synchronized the Dropbox between the mobile device and our PC, the binary file would be automatically uploaded to our PC and then could be analyzed.

The process of the file transferring is completed, but we faced a new problem: a simple transfer of 180 Kb required 3min 30sec, which led to poor performance. In order to solve this problem, we had to identify the bottleneck. It is located on the MMS board or at the application side? Eventually it was located on both sides.

Bottleneck in app:

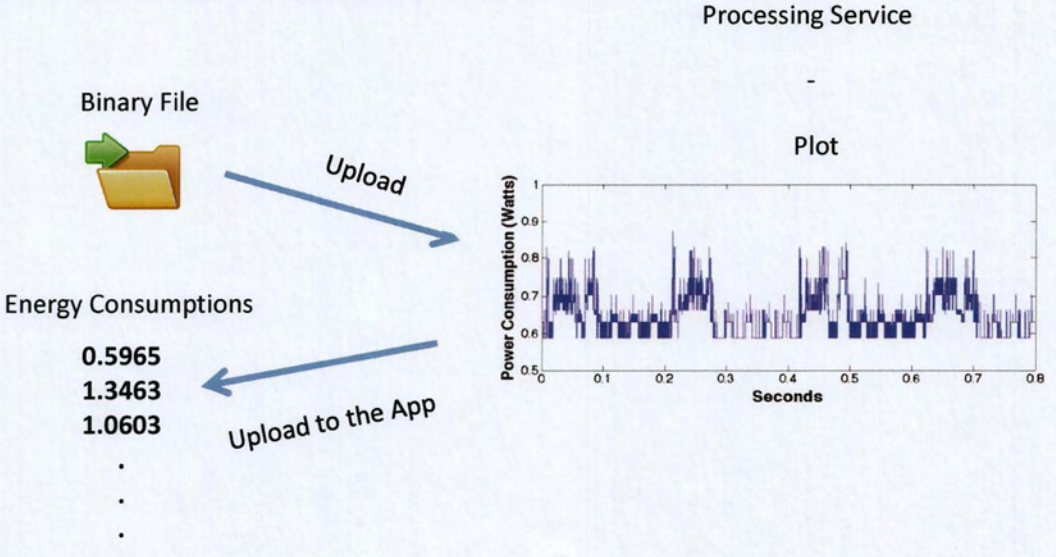
We had to carefully choose the buffer size. The buffer is created on the mobile device when a Bluetooth connection is established between the devices. A big buffer would lead to unnecessarily use of device resources and on the other hand a small buffer would result to poor utilization of Bluetooth transfer speed. After a lot of experiments and tries, we conclude that the appropriate buffer size is 1024 bytes.

Bottleneck in MMS board:

The bottleneck on the MMS side was located in the serial communication between the MCU and the Bluetooth module. We had to experiment with the serial speed and to carefully adjust it. Eventually we achieve a serial communication speed of 115200bps from the initial 9600bps. This speed translates to 1min and 30sec transfer of 180 Kb. That was a reduction of approximately 2 min.

6.4 Processing Service

Mobile devices are not yet equipped with MatLab or Python capabilities **[19]**. In addition, this kind of processes are really demanding on both CPU resources and on battery supply. For this reason, we set the Dropbox to automatically upload the binary file to a PC in which we run a python script. The python script is responsible for the analysis of the energy measurements **[20]**.



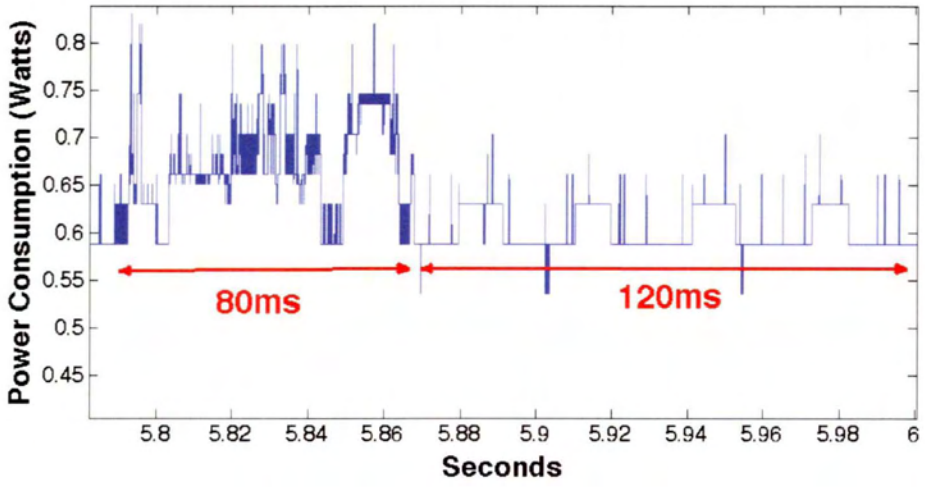
The intention of the python script is the upload to the mobile device, inside the Dropbox file, of a txt file. This txt file is a vector which derives each color to an energy consumption. Hence, the user of the online tools is now capable to characterize his mobile screen and to assign an energy consumption to each color of the tested scenario. The user is able to do this assignment, as the color scenario is static and the color sequence is considered known.

But how the python script identifies the different color presentation? How it knows where and when to measure the energy consumption?

When we implemented the procedure of the broadcasted intents, we initially, as mentioned in chapter 6.2.1, set the time interval for each color to 100 ms. On the plotted energy measurements of this scenario, we noticed some energy peaks. These peaks were the result of the intents, which the main activity was sending to the background service to notify for the color change. These peaks were observed in a pattern, as they were broadcasted in 100 ms intervals. The high sampling rate of the MMS board, offered us the ability to identify these peaks, to measure their duration and to isolate them from the process of the energy consumption. These energy fluctuations affected our measurements but we used them as flags to identify the presented colors. Their exclusion of the energy measurements, could not affect us anymore and the maximum accuracy was achieved.

On the other hand, we were forced to increase the time interval for each color. The energy peak of the broadcasted intent had a time duration which we measured and added to the time interval. We measured a duration of 80 ms for each of the energy peaks. For that reason, and to be assured that the energy we measure is the correct one for each color, **we eventually determined the time interval to 200 ms.**

Here is a screenshot of the plot of an actual energy measurement:



As a result:

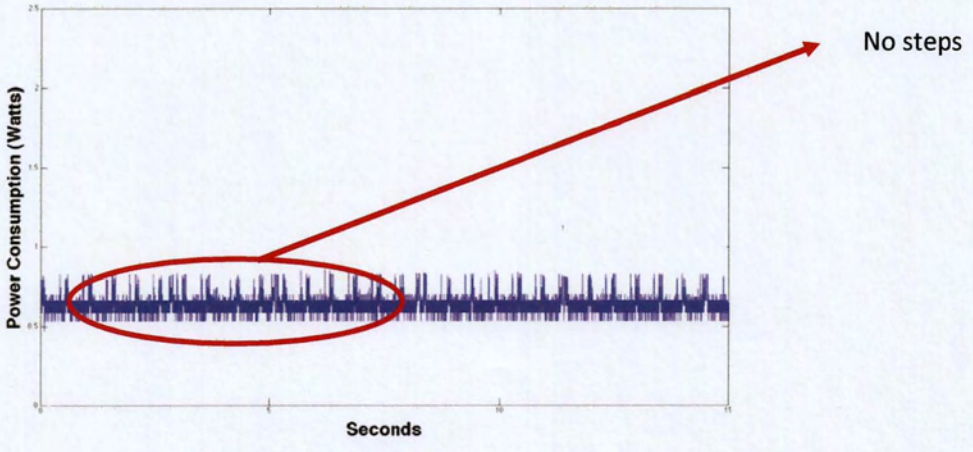
80 ms is the duration of the peak and the python script has 120 ms to measure the energy consumption per color. It constructs a txt file with the consumptions and sends this file to the Dropbox file on the mobile device. The user, as he knows the color sequence, is able to assign each color to its power consumption.

6.5 Feedback and Solutions

Our online tool for the power consumption characterization of mobile screens returns a really useful conclusion. This conclusion can be achieved dynamically, for all the mobile devices and the only required addition is the MMS board attachment to the battery of the mobile device. This conclusion of the power consumption per color can be used as feedback. This conclusion is a color mapping that Android Developers and Designers can take in serious regard when they are developing their Android applications. In this way, they can use colors that are more energy efficient. Another aspect of the use of this feedback, is the addition of a permanent color mapping inside the android stack. The GPU, for example, would be capable of presenting more energy efficient colors and as a result to contribute to the energy reduction.

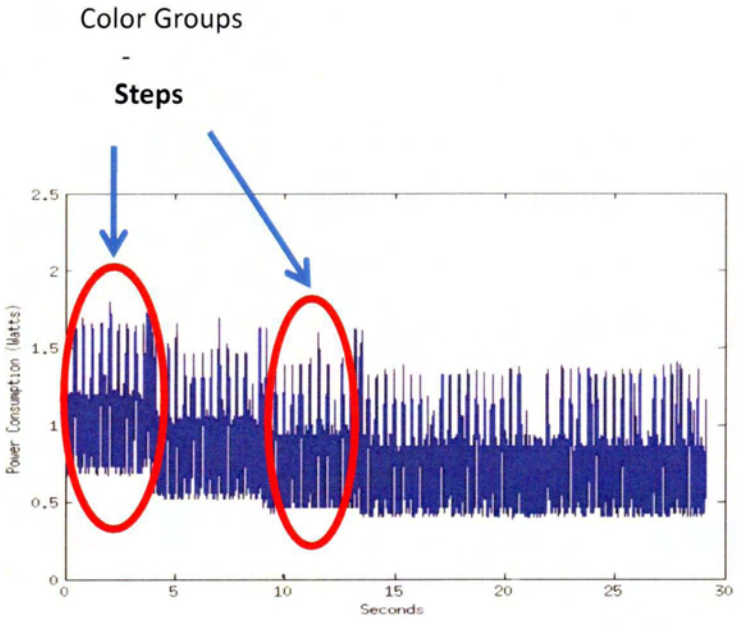
6.6 Real Measurements on the Coloring Scenario

Here follows the plot from the coloring scenario we implemented on the LCD display screen:



In the above plot, we can't identify energy steps because in LCDs displays the decisive energy factor is the backlight source. In the implemented coloring scenario, we set the lighting source to 100% and this parameter dominates the energy consumption. The presenting pixel color seems unable to influence the measurement in a degree that we can observe.

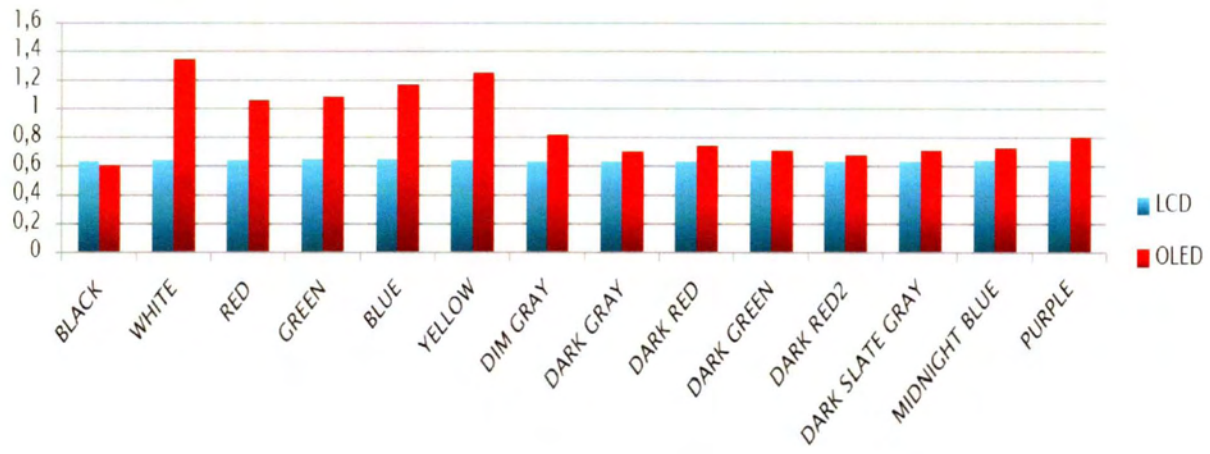
Here follows the plot from the coloring scenario we implemented on the OLED display screen:



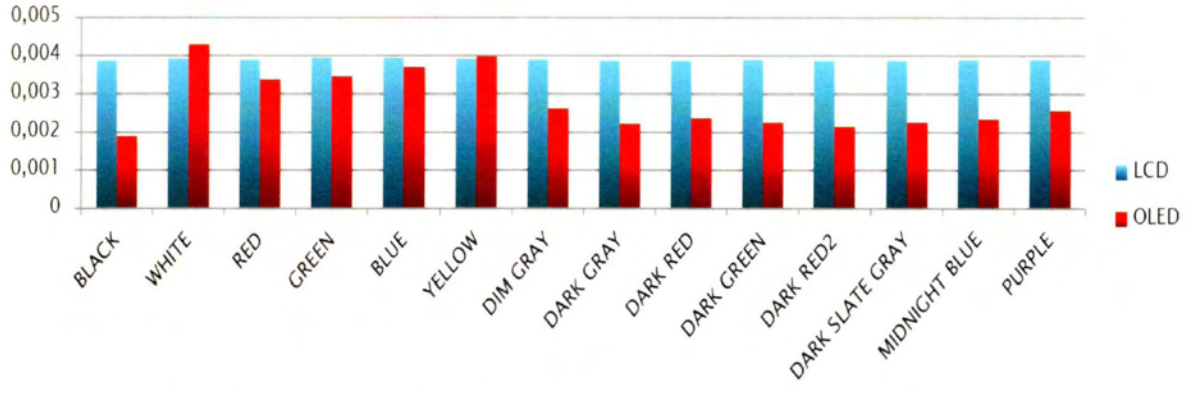
In this plot, we are able to identify energy steps which are caused from various color groups. Pixel presentation affects the energy consumption and that leads to these steps. Darker presentations tend to be more energy efficient.

7. Conclusions

From the returned txt file the user is capable of producing a color mapping in which he can assign energy consumptions to colors. Here is an example of a color mapping:



And if we want the per pixel consumption of our testing devices:



Each OLED display screen is unique and has its own energy behavior characteristics, but there is an observation that stands for all the OLEDs displays. White is the most energy thirsty color and in contrast Black is the most energy efficient. When the pixels emit white presentations, they consume the biggest amounts of electricity but when they emit black, the consumption is really low. LCDs power behavior follows a different pattern: the decisive consuming factor is the backlight source.

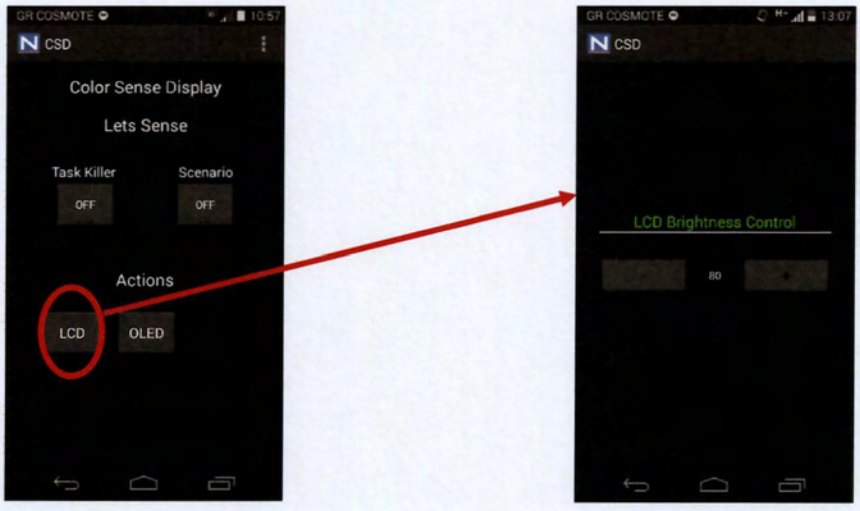
In the above mapping, black presentations consume 58 % less energy than the white ones.

7.1 Application Actions

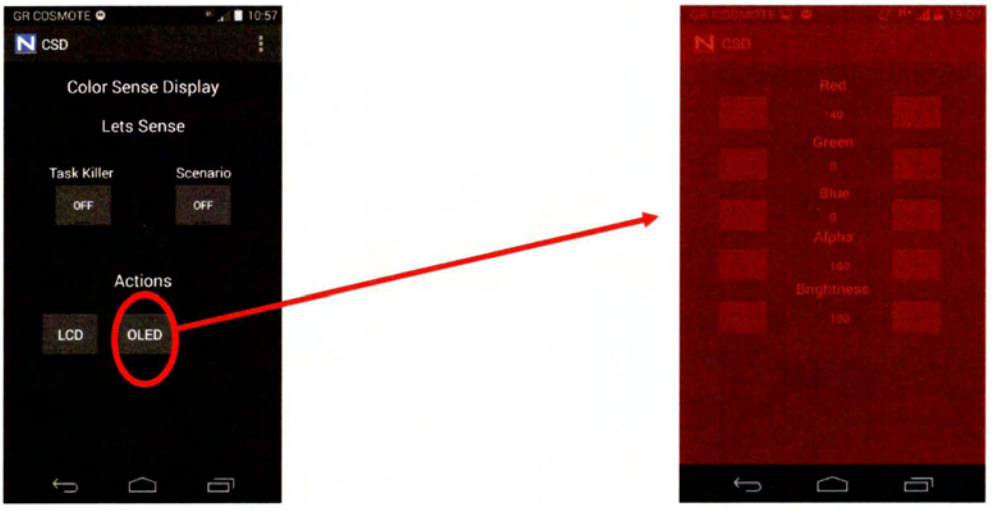
After the assignment of power consumptions to colors, the user can know understand if his mobile screen is LCD or OLED. If he observes steps in consumption, his screen is OLED and on the other hand, if he doesn't notice energy fluctuations, his mobile screen is LCD. Our Application offers two recommended solutions after the identification of the screen technology.

If the screen is LCD, the user should adjust the brightness level in order to reduce the energy consumption.

LCD action:



OLED action:



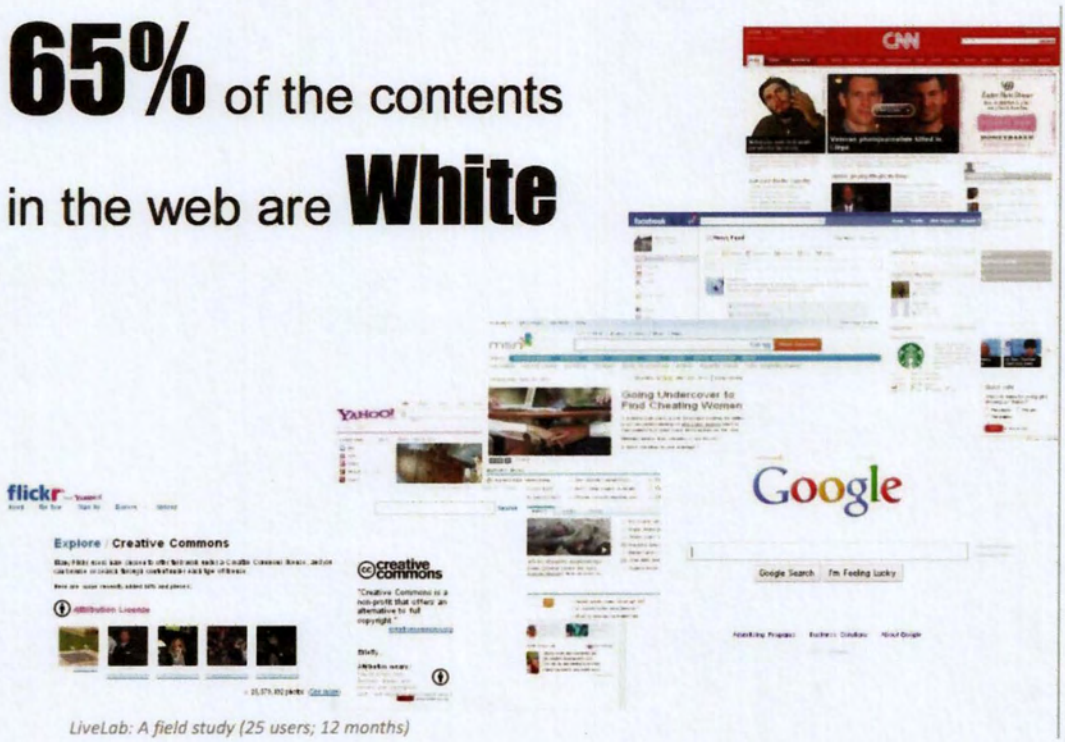
In the above figure, we can see a red color display which was manually set.

By pressing the "OLED" button a new activity (screen) is started. From this java class, the user is able to update and customize the color layout of his mobile screen. He can adjust the red/green/blue values. He can modify the transparency of the color, known as "alpha" value. Finally he can also set the brightness level. In this way, the user can manually set darker colors for his screen and eventually achieve an energy reduction.

The color adjusting process which the user is able to configure, is implemented by the same background service we discussed in chapter 6.3.1. The color parameters are passed with the broadcasted intent from the main acitivity to the background service.

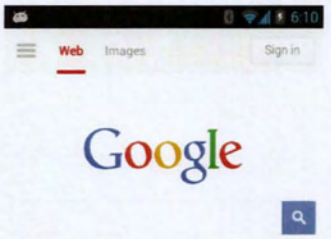
7.2 How to transform colors

We have noticed that the web content is not presented in energy efficient colors. The most visited web sites, such as facebook, google, twitter etc and the most commonly used applications such as skype, viber etc are using white as their main color theme. From our experiments, we have concluded that white is the worst color in terms of energy consumption **[21]**.

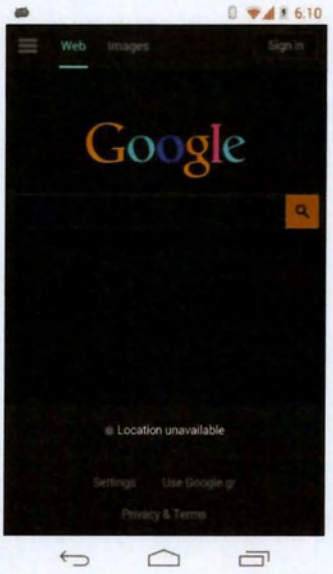


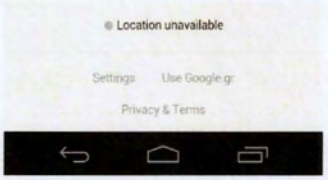
LiveLab: A field study (25 users; 12 months)

Our first and simple thought was a mechanism, capable of transforming white pixels to black ones. We thought that the idea of inverting colors would be a really good approach that would offer us an easy solution to our problem. As a result, we inverted some screenshots of the most commonly used apps and web sites and we measured their energy consumption on both types of mobile screens. In that way, we can observe the energy behavior of the inverted scenario.



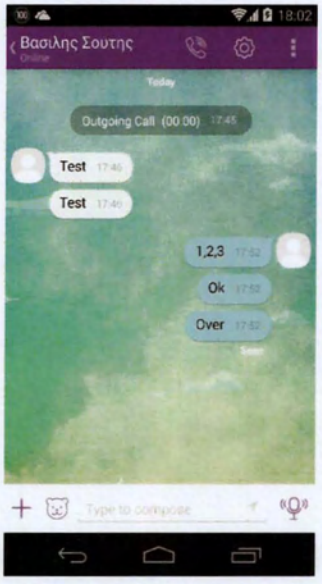
NORMAL
LCD:
0.6407 W
OLED:
1.3508 W



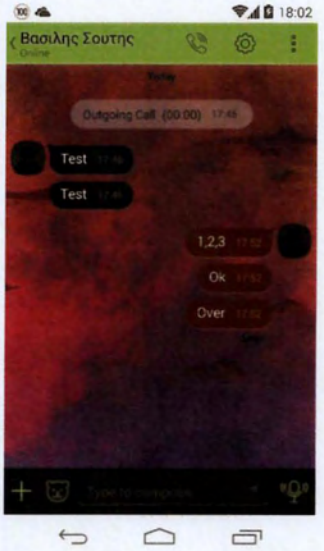


INVERTED
LCD:
0.6111 W
OLED:
0.6012 W

As expected, on the LCD display the consumption is not much affected. We noticed a small reduction but this could be down to other reasons rather than the change in color. On the OLED display the energy reduction is more than 50%. In fact, from 1.3508W it drops to 0.6012W, which is lower than the LCD!! This is really impressive, as the OLED display is 4.65 inches compared to 3.5 inches of the LCD one. The optical result is also great at this inverted scenario.



NORMAL
LCD:
0.6397 W
OLED:
1.1094 W



INVERTED
LCD:
0.6294 W
OLED:
0.9812 W

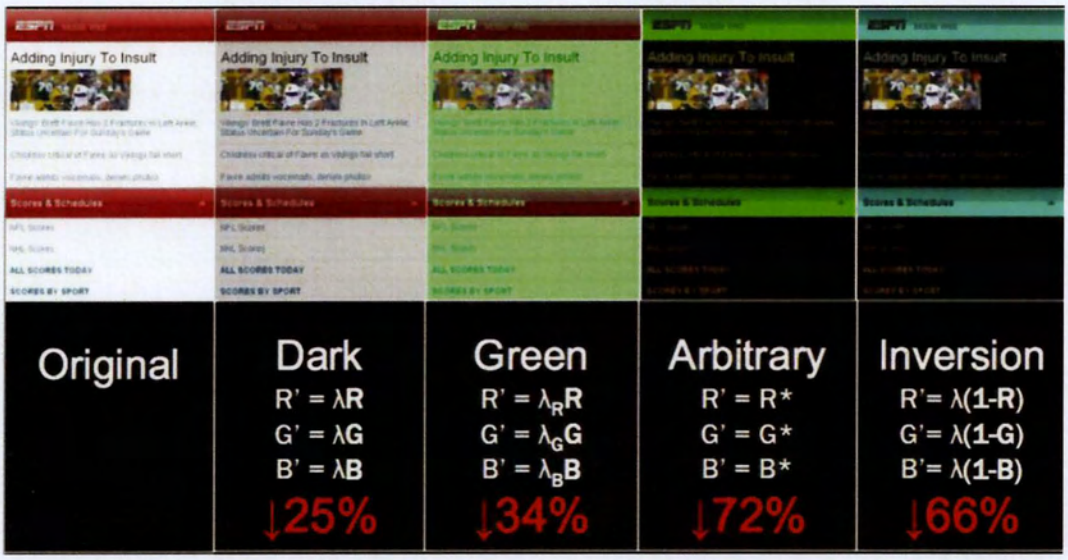
The next inverted screenshot we tested, was the one from Viber. Our initial screenshot was not consisting of so much white as the previous one and the inverted

picture was not that good from optical perspective. In the LCD scenario the energy reduction was negligible and also in the OLED was small. Much smaller than the previous screenshot. So we can admit, that this is a scenario where the inverted coloring is not working in the way we wanted. The energy reduction was small and we faced difficulties in reading the presented text. On the other hand, we identified an energy efficient pattern, as the consumption was lower and so, if one wants to save even some small portions of battery life, he can use the inverted coloring solution.

7.3 Other Coloring Solutions and Ongoing Work

A very interesting coloring solution is the Chameleon project, a color – adaptive Web browser for Mobile OLED displays (Mian Dong and L. Zhong, ACM Mobisys 2011, Best Paper Award). In this project, they statically measured the energy behavior of a specific device, they produced the color mapping and they developed a browser that changes the presenting themes in order to reduce the power consumption **[22]**.

Various color schemes and their energy behavior.



The disadvantage of this approach was, that they measured the energy behavior of a specific mobile device. Our online tool is capable of the online power consumption

characterization of any mobile device. The only requirement is the addition of the MMS board. Ideally, our online tool would be used as an input to this project and we could have a tool that dynamically produces the color mapping of any device and adapts the display presentation while browsing.

On our ongoing work for the inverted solution, we are facing a challenge: up to this point, we are painting the whole screen but our intention of the inverted scenario needs the per pixel coloring technique. This is something we are looking on how we are going to achieve it **[23]**.

References

[1]. Lcd Technology explained : http://www.practical-home-theater-guide.com/lcd-display.html

[2]. Super AMOLED vs Super LCD : http://www.techradar.com/news/phone-and-communications/mobile-phones/super-amoled-vs-super-lcd-the-big-screens-compared-1226721

[3]. IPS technology explained : http://www.pctechguide.com/flat-panel-displays/ips-in-plane-switching-lcd-monitors

[4]. Oled Technology explained: http://www.oled-info.com/oled-technology

[5]. LG OLED vs IPS LCD power consumption slide: http://www.oled-info.com/lg-oled-vs-ips-lcd-power-consumption-slide

[6]. General specs and technology comparison: http://en.wikipedia.org/wiki/Comparison_of_CRT,_LCD,_Plasma,_and_OLED

[7]. Power Management for OLED Displays: http://conferences.sigcomm.org/sigcomm/2012/paper/mobigames/p25.pdf

[8]. Super AMOLED vs Super LCD : http://www.techradar.com/news/phone-and-communications/mobile-phones/super-amoled-vs-super-lcd-the-big-screens-compared-1226721

[9]. Motivation & Background : http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6513473

[10]. S. Keranidis, G. Kazdaridis, V. Passas, G. Igoumenos, T. Korakis, I. Koutsopoulos and L. Tassiulas, "NITOS Mobile Monitoring Solution: Realistic Energy Consumption Profiling of Mobile Devices", presented in ACM e-Energy 2014, Oxford, UK, 11-13 June 2014

[11]. Arduino Environment: http://arduino.cc/en/guide/Environment

[12]. Task Killer Explained : http://lifehacker.com/5650894/android-task-killers-explained-what-they-do-and-why-you-shouldnt-use-them

[13]. Pixel Density Explained : http://community.giffgaff.com/t5/Blog/Pixel-Density-Pixels-Per-Inch-PPI-Explained/ba-p/9252950

[14]. Power Modeling of Graphical User Interfaces on OLED Displays : http://www.ruf.rice.edu/~mobile/publications/dong09dac.pdf

[15]. Eclipse IDE: https://www.eclipse.org/

[16]. Android Development : http://developer.android.com/training/index.html

[17]. Refreshing Coloring Parameters : http://stackoverflow.com/questions/14872354/refresh-all-view-in-android

[18]. Bluetooth Transfer in Android : http://www.egr.msu.edu/classes/ece480/capstone/spring14/group01/docs/appnote/Wirsing-SendingAndReceivingDataViaBluetoothWithAnAndroidDevice.pdf

[19]. Matlab Processing Service : http://www.mathworks.com/products/signal/

[20]. Python Daemons : http://stackoverflow.com/questions/4705564/python-script-as-linux-service-daemon

[21]. Transforming Colors : http://developer.android.com/reference/android/graphics/ColorMatrixColorFilter.html

[22]. Chameleon: A Color-Adaptive Web Browser for Mobile OLED Displays: http://www.ruf.rice.edu/~mobile/publications/dong11mobisys_chameleon.pdf

[23]. Per Pixel Coloring : https://source.android.com/devices/graphics.html