Πανεπιστήμιο Θεσσαλίας

Πολυτεχνική Σχολή

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Νιζάμης Γ. Αλέξανδρος

Μελέτη της χρήσης των αισθητήρων σε κινητά android

Study of android phone sensor use

Βόλος, 2014

Dedicated to Georgia, my family and my friends

Credits

First of all, I would like to thank my three supervisors P.Bozanis, P.Tsompanopoulou and A.Housti for the trust that they showed on me and they offered me such an interesting subject for my thesis.
Special thanks go to Athanasios Fevgas for his help in this Master thesis.
Finally thanks my family which supports me for many years.

Alexandros Nizamis
Volos, 2014

Contents

List of Figures

List of Tables

List of Algorithms

Περίληψη

Στις μέρες μας παρατηρείται τεράστια αύξηση στον αριθμό των έξυπνων κινητών τηλεφώνων και των ταμπλετών. Το μεγαλύτερο ποσοστό αυτών των συσκευών τρέχει λειτουργικό Android. Το Android σε συνδυασμό με το υλικό των συσκευών προσφέρει πλήθος αισθητήρων οι οποίοι παρέχουν στο χρήστη πλήθος ευκολιών και δυνατοτήτων μέσα από τη χρήση των διαφόρων εφαρμογών. Οι αισθητήρες μπορούν να προσφέρουν πληροφορίες σχετικές με την κίνηση της συσκευής στο χώρο ή πληροφορίες για τον ίδιο το χώρο στον οποίο βρίσκεται η συσκευή. Στη παρούσα μεταπτυχιακή διατριβή πραγματοποιείται ενδελεχής μελέτη των αισθητήρων αυτών καθώς και της χρήσης τους. Επίσης στα πλαίσια της εργασίας αυτής υλοποιείται ένας σύνθετος αισθητήρας για την ανίχνευση εσωτερικού ή εξωτερικού χώρου για συσκευές Android. Για την υλοποίηση του γίνεται χρήση των βασικών αισθητήρων που παρέχονται από τις συσκευές και το Android.

Abstract

Nowadays, there is a huge increase of the number of Smartphones and Tablets. The largest part of this number is devices which are running Android OS. The combination of Android OS with the hardware of the devices provides a variety of sensors. These sensors offer to user many features via the applications which the user possesses. Android sensors provide information about device's motion or information about the surrounding environment of the device. In this Master thesis made a study of Android phones' sensors use. Moreover, in this thesis developed a software/virtual sensor for Android devices. This sensor detects if the surrounding environment is an indoor or an outdoor environment. The components of the new sensor are common sensors which are provided by mainstream Android devices.

Chapter 1

Introduction

1.1 Thesis Contribution

The contribution of this Master thesis is an advanced study of Android phones' sensors and the implementation of an indoor/outdoor detector combining Android's standard sensors. Firstly, every one of the existing Android sensors is presented. Built-in and software/virtual sensors are studded. Also this thesis provides a methodology of reading sensors' values using the programming language Java and Android SDK. A methodology for combining sensors programmatically is described too. The final provision of this thesis is the implementation of a low energy cost indoor/outdoor detector that uses only basic sensors which are met in common mobile models.

1.2 Thesis Organization

- Chapter 2 consist an overview of Android OS and provides the general theoretical background.
- Chapter 3 consist a study of all Android phones' sensors. Also in this chapter is described a method for retrieving data from device's sensors.
- Chapter 4 consist a study of Android's virtual sensors.
- Chapter 5 consist the presentation of a research paper from University of Singapore. This is the paper on which is based the implementation of current thesis.
- Chapter 6 presents the implementation of indoor/outdoor detector.
- Chapter 7 presents the results of the experiments conducted and the conclusions

Chapter 2

Android OS Review

2.1 Introduction

Android is a mobile operating system currently developed by Google. It is based on the Linux Kernel. Android mainly is designed for touch screen devices such as smartphones and tablets. Google releases Android’s source code under open source licenses. Android’s open nature has encouraged a large part of mobile devices manufacturers to adopt it. Today Android represents over than 80% of total mobile devices’ OS market [1]. But Android’s success does not stop only in commercial field. It has a large community of developers and it is a very promising field of research in universities.

2.2 History [1, 2]

Android Inc was founded in October 2003 in Palo Alto, California by a small group of developers and designers. The members of this group were Andy Rubin, Rich Miner, Nick Sears and Chris White. Initially Android was developed for televisions, game consoles, digital cameras and electronics. Later Android shifted to Mobiles. Their target was to create smarter mobile devices based in their owner’s position and preferences as they said. But Android’s faith changed dramatically by the time Google acquired it. This happened on August 2005. After the acquisition many key employees stayed at the company such as Rubin, Miner and White. In 2007 Google announced the development of Android OS, a mobile device platform based on Linux kernel. One year later the first commercial mobile phone running Android OS was released. It was the HTC Dream. Dream was running Android 1.0 or Android Alpha. Android’s logo [3] was a green robot. A creation of California-based graphic designer named Irina Blok. It is a prime example of a playful logo which corresponds to Google’s profile. The green color stands for growth, freshness and prosperity. In the next versions the features was added from one version to another. Also have been released many subversions which fixed bugs or improved the performance in specific parts. Except the first two versions all the other versions have the name of a sweet.

Today Android counts 11 main versions and it is the most used mobile OS in the world. Figure 1.1 represents the Android's versions evolution.

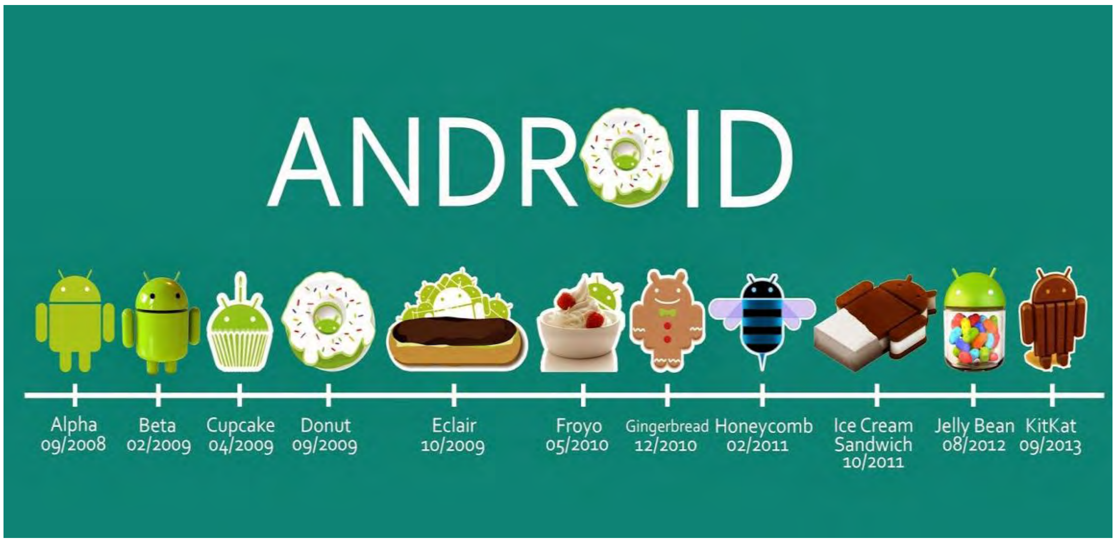


Figure 2.1: Android OS Evolution from [4]

Separately from smartphones and tablets today Android is used on other electronics, including laptops, Smart TVs and digital cameras. Lately it is used on smart glasses/Google Glasses, smart watches, portable media players and cars.

2.3 Architecture [2, 5]

The Android OS can be referred as a software stack of different layers. Each layer provides services to the layer just above it. From bottom to top the layers are Linux Kernel, Libraries, Runtime, Application Framework and Applications as they presented in Figure 2.2.

2.3.1 Linux Kernel

Linux Kernel is the basic layer of Android's software stack. It provides core services including process and memory management, security, network, hardware drivers and power management. These services are handled by a Linux 2.6 Kernel with some further Google's architectural changes. Generally kernel layer provides all mechanisms that control and communicate with the hardware. So it acts as an abstraction layer between the hardware and other software layers. The selection of

Linux kernel, one so popular and proven foundation, made the porting of Android to variety of hardware, a painless task.

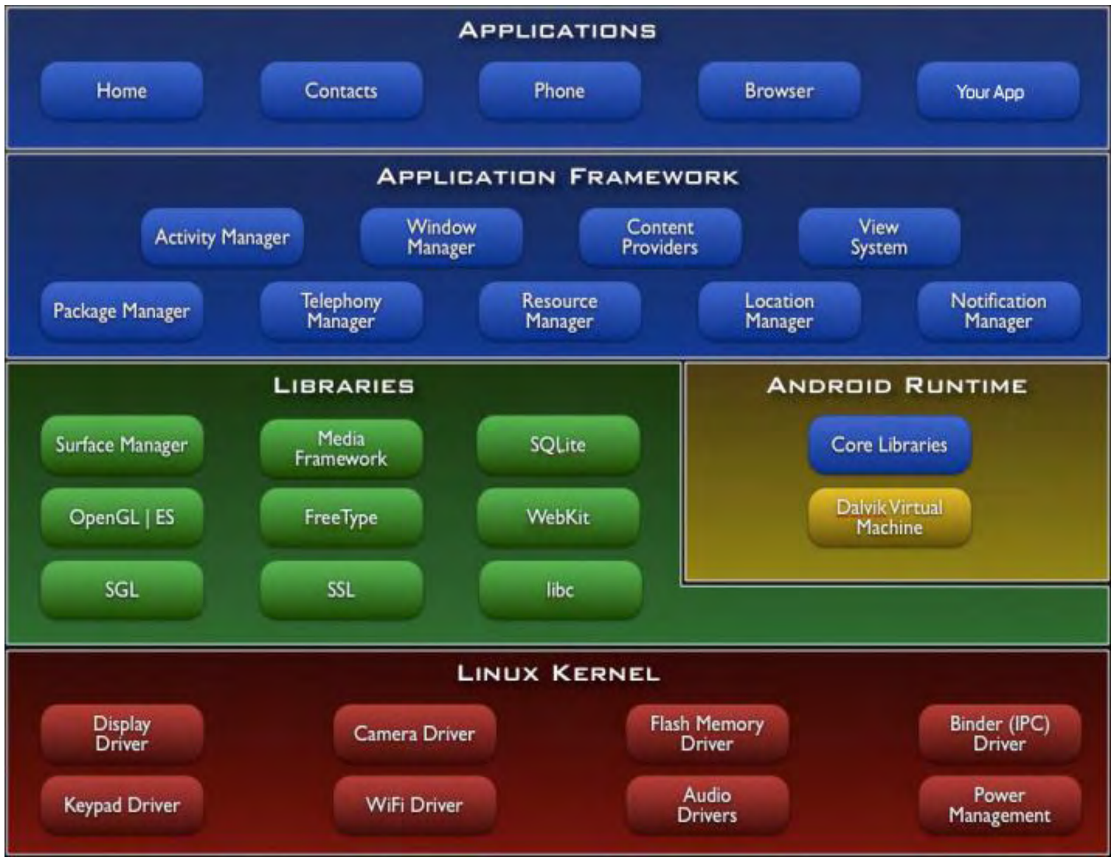


Figure 2.2: Android Architecture Diagram from [5]

2.3.2 Libraries

Android's native Libraries are running on top of Linux kernel. These Libraries are written in C or C++ programming languages and they are designed for a particular hardware. The most important native Libraries are Surface Manager, SQLite, OpenGL, Media Framework, WebKit and SSL. Surface manager is used for composing window manager with off-screen buffering. This buffering offers the transparency of windows and is an important part of the screen the user sees. SQLite is the database engine used in Android for database support. OpenGL is a graphic library for 2D and 3D graphics. Media framework provides media codecs for playback of audio and video media. WebKit and SSL are for integrated web browser engine used to display HTML content and internet security respectively.

2.3.3 Android Runtime

Android Runtime is what makes an Android phone rather than a Linux mobile device. It is the engine that powers the applications. Android Runtime consists of Dalvik Virtual Machine and Core Java Libraries. Dalvik Virtual Machine is a register-based virtual machine used to run applications in Android devices. It is optimized for low processing power and low memory environments. It provides high efficiency, security, memory management and thread support. Android applications are written at Java language. But Dalvik is not a Java virtual machine. So something must provide the functionalities are available in traditional core Java libraries. This is what Core Java Libraries of Android Runtime does.

2.3.4 Application Framework

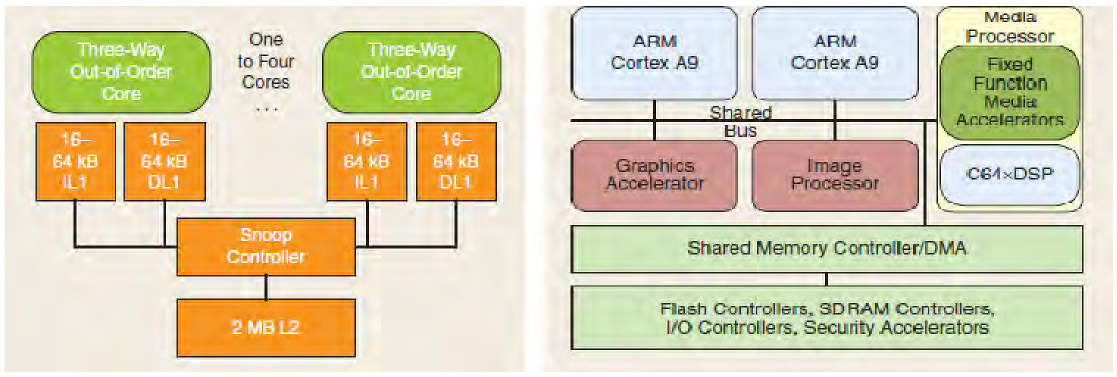
Application Framework is the last layer before applications and interacts with them. It is provides the classes used to develop Android applications. This framework offers mechanisms for resources management. It contains important blocks such as Activity Manager, Content Providers, Resource Manager, Location Manager and Telephony Manager. The first manages the activity life cycle of an application. Content Providers offer data sharing mechanisms between applications. Resource Manager manages the various types of resources of applications. Location Manager uses GPS or cell tower for location management. Telephony Manager manages all voice calls. It provides access to information about telephony services on the Android device.

2.3.5 Applications

Applications are the top layer in Android Architecture. All applications are built on this layer. Applications layer runs within Android Runtime which is presented above. Also it uses the services of Android Framework. Most devices initially come with pre-installed applications in this layer. Usually these applications are web browser, contact manager, dialer and SMS messenger.

2.4 Android Hardware Requirements

The main platform for Android is the 32bit ARMv7 architecture. Others are the Android-86 project which introduces x86 architecture and lately an Intel processor architecture. In Figure 2.3, are presented two representative processor units for Android devices. These are ARM Cortex A9 processor and TI-OMAP 4430 system on-chip.



Three-Way Out-of-Order Core
One to Four Cores
16–64 kB IL1
16–64 kB DL1
Snoop Controller
2 MB L2
ARM Cortex A9
Shared Bus
Media Processor
Fixed Function Media Accelerators
C64×DSP
Graphics Accelerator
Image Processor
Shared Memory Controller/DMA
Flash Controllers, SDRAM Controllers, I/O Controllers, Security Accelerators

(a) **ARM Cortex A9** **(b) TI OMAP 4430**

Figure 2.3: Processor Units of Android Devices from [6]

ARM provides general purpose mobile embedded cores. They offer high general-purpose performance. ARM's processors targeted to run an OS and desktop style applications. Unfortunately data dominated applications will not execute efficient from this kind of processors. The design is targeted for Smartphones with handless control and their traditional desktop style applications. Arm's processors support high performance in low power. This feature makes them extremely desirable for Android devices. On the other hand TI OMAP is not a regular processor but it is a general-purpose system-on-chip (SoC) targeted to Smartphones and mobile-internet-devices. OMAP devices include a general-purpose ARM processor core similar with one was described above plus a number of specialized co-processors. These Co-processors are GPU, an image processor and an audio/visual codec processor. They offer specific processing power with very low energy consumption.

Except the requirement for ARM family's processor, Android requires an amount of RAM, a flash memory and a GPU if it is not part of processor unit. Original minimums were 32MB of RAM and 32MB of flash memory. These requirements have upgraded by the years. The last Android version requires a minimum 340MB RAM while recommended minimum amount of RAM is 512MB. In graphics sector Android requires a GPU that supports OpenGL ES 2.0 and lately OpenGL ES 3.0. To get all Android's features a device must be equipped with many optional hardware components like camera, GPS and several sensors.

2.5 Features

Android is a mobile operating system which provides a large list of features. In generally it offers a touch corresponded environment based on direct manipulation.

From users view, Android provides communication services, computing services, internet services, data storage services, sensor services, location services, camera services and a big variety of multimedia services. Android's full features can be listed as:

- Messaging
- Web browser
- Voice based features
- Multi-touch
- Multitasking
- Multiply language support
- Accessibility
- Handset layouts
- Video Calling
- Connectivity
- Tethering
- Location Services
- Media Support
- Sensing
- Internal or External Storage
- Easy applications downloading via Google Play
- Over 1 million applications

From developers view, Android provides an open source operating system and a full equipped Software Development Kit, the Android SDK. The SDK includes a set of development tools including a debugger, software libraries, a handset emulator, documentation, sample codes and tutorials.

Chapter 3

Android Phones’ Hardware Sensors

3.1 Introduction

Android phones provide many features. A very important feature is their built-in sensors [7, 8]. The addition of these sensors that can report information about the environment made the phones more useful and powerful for developers and users. Android’s sensors are capable of providing raw data with high precision and accuracy. There are three broad categories of sensors are supported in Android platform:

1. **Motion Sensors [9]:** These sensors measure rotational forces and acceleration forces along three axes.
2. **Environmental Sensors [10]:** These sensors measure various environmental parameters such as humidity, air pressure and temperature.
3. **Position Sensors [11]:** These sensors offer measurements of the physical position of a device.

The available hardware sensors on the Android devices are:

- Accelerometer
- Gyroscope
- Magnetic field sensor
- Light sensor
- Proximity sensor
- Humidity sensor
- Pressure sensor
- Temperature sensor

Usually most new Android phones are equipped with the first five of these seven sensors. The last two are met only in high-end and expensive models.

3.2 Sensors Overview

In general a sensor [12] is a device that detects and responses to some type of environment's changes. Environment's changes are the input of a sensor. The output is generally a signal that is converted to a human-readable display. Also an analog sensor signal needs to be converted to a digital signal which is suitable for processing. A sensor must provide correct and accurate results. A sensor is characterized as effective when:

- It is sensitive only to the measured property
- It is insensitive to all the others properties
- It does not influence the measured property

Sensitivity of a sensor is defined as the change in output per the change in value of measured property. But sensors have some deviations. First of them is the sensitivity error. This error occurs to the fact that sensitivity in practice may differ from the value specified. Second deviation is about sensors' offset. If a measured value is zero then the sensor's result is not zero but it is its offset value. A third factor which reduces sensor's effectiveness is the limitation of output range. That means that the output has a minimum and a maximum value which the measured property can exceed. Another deviation is caused when the environment changes extremely fast in time. This is a dynamic error. The last factor that causes deviation is the noise of the signal.

3.3 Android's Hardware Sensors

3.3.1 Accelerometer

Accelerometer [9] is a motion sensor. This sensor measures the acceleration force that is applied to a device. The applied force is measured on all three physical axes(x, y and z). Moreover, the gravity force is being included. The unit of measurement is m/s^2.

In more details, the accelerometer's coordinate system can be described as follow:

When a device is held in its default orientation:

- The X axis is horizontal and points to the right
- The Y axis is vertical an points up
- The Z axis points toward the outside of the screen face

In Figure 3.1 is presented the 3-axes coordinate system of Android phones' sensors:

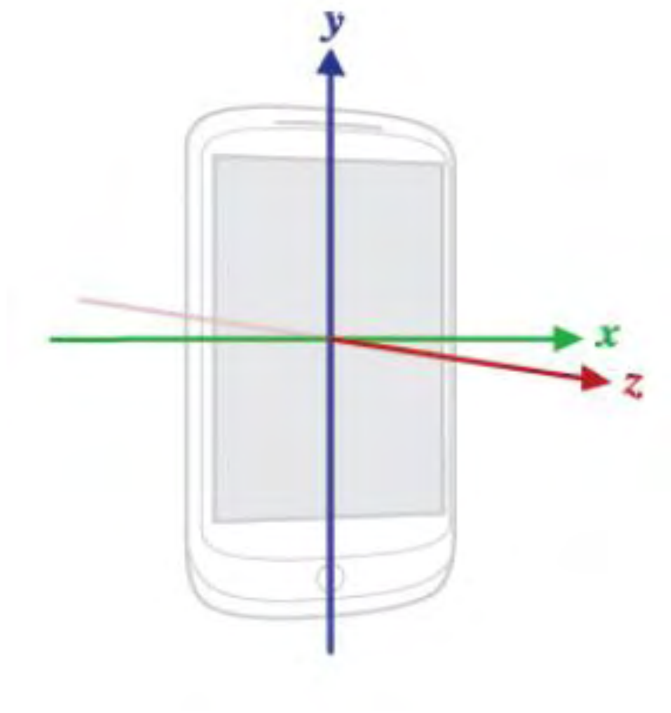


Figure 3.1: 3-axes coordinate system from [7]

Accelerometer's usage is motion detection such as phone shake and tilt. It is used too much in games applications. Furthermore accelerometer is used in many positioning and movement detection applications.

3.3.2 Gyroscope

Gyroscope [9] is other one motion sensor built-in Android devices. It measures a device's rate of rotation around each of the three physical axes as they described above. Rotation is positive in the counter clock-wise. This is the standard mathematical definition of positive rotation. Gyroscope's unit of measurement is rad/s. Common gyroscopes provide raw rotational data without any filtering or correction for noise. As a result gyroscope introduces errors. Generally gyroscope is responsible for many lags in game applications. Moreover, many delays in device's orientation change are due to gyroscope usage and its bad returning values. The uses of gyroscope are about rotation detection. Especially spin and turn detection. This sensor allows the device to rotate from portrait to landscape when a user turns the device. Also it is used as a controller in many mobile games.

3.3.3 Magnetic Field Sensor

Magnetic field sensor [10, 13] measures the ambient geomagnetic field for all three physical axes. Its module is μT. In other words magnetic field sensor detects changes in the earth's magnetic field. The fact that a magnetic sensor is built-in in a mobile device leads to many problems in sensor's correct use. This disadvantage comes from lots of electronics in the device which means lots of electromagnetic interference. The best technique that reduces this problem is Flight Mode option be enabled. This mode switches off all the transmitters. After that option the biggest single influence to magnetic field is the screen brightness. It would seem to be due to the electromagnetic field by the screen backlight. But if all measurements are taken with the screen in the same brightness then the effect of the backlight is compensated for. The magnetic field sensor commonly is used for detect positioning and the development of applications which represent a compass.

3.3.4 Proximity Sensor

Proximity sensor [10] is a position sensor as the magnetic field sensor. This sensor measures the proximity of an object in relative to the view screen of a device. Its measurement unit is cm. This sensor's result output is not in the 3-axes coordinate system. Its output is a single value that indicates the distance in cm. Instead of this in many phones' sensors the returning result is not arithmetic. This type of proximity sensor returns a binary far or near measurement. Proximity sensor always is placed at the front face of the phone near the handset as in Figure 3.2 at the next page. This sensor is typically used to determine how close handset and user's ear are. This is very helpful. When the device detects via proximity sensor that the user's ear are near to handset in a phone call then it switches off the screen and saves the battery's life.

3.3.5 Temperature Sensor

The temperature sensor [11] is an Environmental Sensor. It measures the temperature of a device in degrees Celsius (°C). But this sensor was deprecated in last Android versions. More exactly it was replaced with the ambient temperature sensor. This new sensor measures the ambient air temperature in °C too. It is not a common sensor. It exists in high-end devices or in some rugged phones. Many thermometer applications are based in ambient temperature sensor.

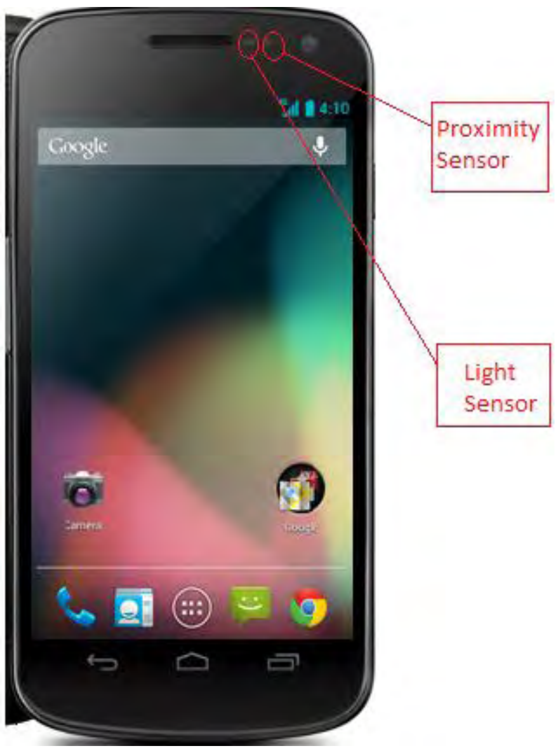


Figure 3.2: Light and Proximity Sensors

3.3.6 Pressure Sensor

It is other one Environmental Sensor [11]. Pressure sensor measures the ambient air pressure. This sensor has two units of measurement. They are hPa and mbar. It is also a non-common sensor which is found in expensive models. By using the atmospheric changes it is possible for an application be able to forecast the weather.

3.3.7 Humidity Sensor

In some Android devices there is a humidity sensor. This sensor called Relative Humidity sensor [11]. It measures the relative ambient humidity in percent (%). It is an Environmental type sensor. As an Environmental sensor, humidity sensor returns a single value result in percent and not a multidimensional array of values as motion sensors. The humidity sensor is monitoring absolute and relative humidity and dew point respectively in many applications.

3.3.8 Light Sensor

The light sensor [11] is an Environmental Sensor. It measures the ambient light level (illumination). Lx [14] is its unit of measurement. Similar to proximity sensor, the light sensor is appeared at the front face of the device near the handset and the front camera. There is an example in Figure 3.2. The light sensor is a very advantageous feature in Android devices. The detection of the ambient light gives the ability of controlling the screen brightness. This protects the life battery of the new Smartphones with high resolution screens and high size screens.

3.4 Programming Android Sensors

3.4.1 Sensor Framework

The Android sensor framework [7, 15] is part of the android.hardware package. This framework provides several classes and interfaces. They are the key for development a wide variety of sensor-related tasks. The most important framework's features are the following:

- Define all the sensors which are available on a device
- Determine sensor's maximum range and resolution
- Acquire raw sensor data
- Register and unregister sensor event listeners

All the above features are provided by framework's classes and interfaces. These are:

- **Sensor Manager:** This class creates an instance of the sensor service. Also it provides methods for accessing and listing sensors and methods for registering and unregistering sensor event listeners. Moreover sensor manager provides sensor constants which are capable to report sensor accuracy and to set data acquisition rates.

- **Sensor:** The Sensor is a class that creates an instance of a specific sensor. This class offers several methods which let a developer to determine a sensor's capabilities.

- **SensorEvent:** This class provides information about a sensor event. To achieve that it creates a sensor event object. The sensor event object includes information about: the type of sensor that generated the event, the raw sensor data, the accuracy of the data and the timestamp for the event.

- **SensorEventListener:** This interface provides two callback methods that receive notifications. These notifications or sensor events are received when sensor accuracy changes or when sensor values change.

3.4.2 Algorithm for programming with Android sensors

At the following table is represented a general algorithm for all Android sensor. They are explained the basic steps which help a developer to retrieve data from a single sensor.

Algorithm 3.1 Retrieves Data from an Android Sensor

```
Main Activity implements SensorEventListener  /*listener will get informed for
                                                        Sensor changes*/
    Sensor Manager variable declaration
    Sensor variable declaration

    onCreate  function starts
            Sensor Manager gets System Service
            Sensor gets Default Sensor type  /*takes a sensor type such as
                                                Accelerometer for example*/
            If Sensor == null  return  “there is not this sensor in the device”
                        Exit
    onCreate  ends

    onSensorChanged function starts
                Result = event’s value /*here is the sensors output*/
    onSensorChanged ends

    onResume function starts
            Sensor Manager Register Listener
    onResume ends

    onPause function starts
            Sensor Manager un-Register Listener
    onPause ends

Main Activity ends
```

This is a pattern that can be applied to all Android sensors. The output from event is a single value result or an array of three results. To determine a specific sensor it must be declared in getDefaultSensor method with the corresponded Sensor class’s constant. In Table 3.1 are displayed the Sensor's constants and the event's values for every sensor.

Built-in Sensor	Sensor Class Constant	Event Value
Accelerometer	TYPE_ACCELEROMETER	Event.value[0] Event.value[1] Event.value[2]
Gyroscope	TYPE_GYROSCOPE	Event.value[0] Event.value[1] Event.value[2]
Magnetic Field	TYPE_MAGNETIC_FIELD	Event.value[0] Event.value[1] Event.value[2]
Proximity	TYPE_PROXIMITY	Event.value[0]
Temperature	TYPE_TEMPERATURE Or TYPE_AMBIENT_TEMPERATURE	Event.value[0]
Air Pressure	TYPE_PRESSURE	Event.value[0]
Humidity	TYPE_RELATIVE_HUMIDITY	Event.value[0]
Light	TYPE_LIGHT	Event.value[0]

Table 3.1: Sensor's Constants and Event's Values

Chapter 4

Android's Software Sensors

4.1 Introduction

Software sensors are also known as virtual sensors or fusion sensors or composite sensors [7, 16]. As it is indicated by their name they are not built-in hardware parts of the device. This type of sensors is created by software developers. Software sensors are implemented from underlying base sensors on the device. In the newer editions of Android OS there are some custom virtual sensors. They are the rotation vector sensor, orientation sensor, step counter and step detector.

4.2 Custom Virtual Sensors

4.2.1 Orientation Sensor

The orientation sensor [10] acquires its data by using accelerate sensor and geomagnetic field sensor. This virtual sensor monitors the position of a device relative to the earth frame of reference. Data are providing in the following three dimensions:

- Azimuth. This is the angle between y axis of the device and the magnetic north. It represents the degrees of rotation around z axis. Its range of values is 0 degrees to 270 degrees.
- Pitch represents the rotation's degrees around x axis. Specifically it represents the way which z axis rotates towards y axis. The range of values is 180 degrees to -180 degrees.
- Roll introduces how the z axis rotates toward the x axis. Thus, it produces the degrees of rotation around the y axis. Its range of values is 90 degrees to -90 degrees.

Figure 4.1 is a schematic display of these three dimensions:

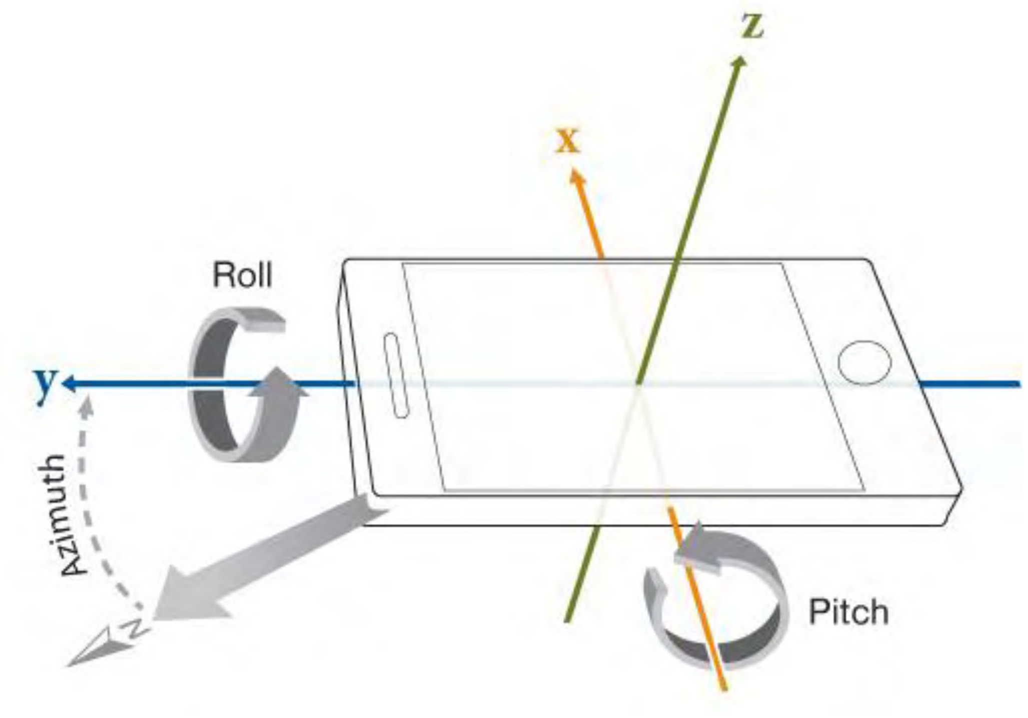


Figure 4.1: Orientation Sensor's Coordinate System from [17]

The orientation sensor has to retrieve and to process raw data from accelerometer and geomagnetic sensor. As a result of this heavy processing is the reduction in the accuracy and the precision of the orientation sensor. This is a factor that many times makes the use of this sensor inappropriate.

4.2.2 Rotation Vector Sensor

The rotation vector [10, 16] provides the orientation of the device as a combination of an angle and an axis, in which the device has rotated through an angle θ around an axis (x, y, or z). Its coordinate has the following characteristics:

- X is the vector product Y x Z. It points approximately East.
- Y points toward the geomagnetic North Pole.
- Z is perpendicular to the ground and points toward the sky.

The above coordinate system is presented in Figure 4.2:

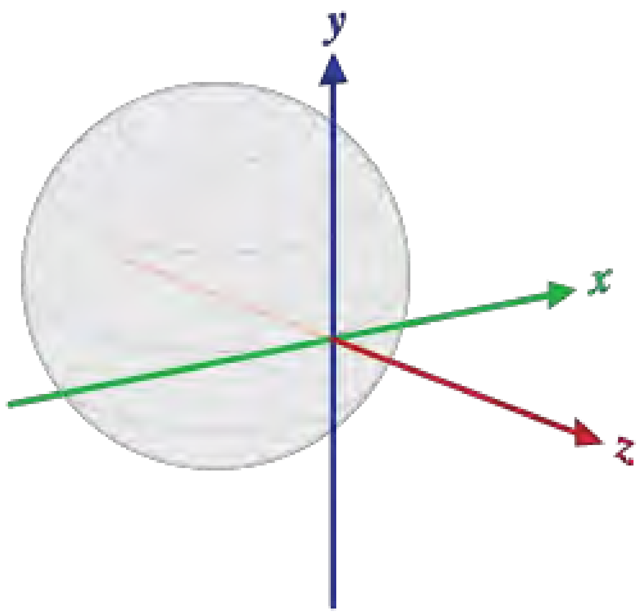


Figure 4.2: Rotation Vector Sensor's Coordinate System from [16]

This sensor combines raw data from accelerometer, gyroscope and geomagnetic field sensor. Against orientation sensor, rotation vector sensor provides more accurate data. Similar to this sensor is the Game Rotation Vector Sensor. It is almost the same sensor with the difference that the last one does not use the geomagnetic field and its Y axis does not point to the North.

4.2.3 Step Counter and Step Detector Sensor

The step counter sensor [16] calculates the number of steps taken by the device's user. The step counter sensor [16] counts since the last reboot while the sensor was activated. On the other hand the step detector sensor activates an action every time a user takes a step. This sensor returns value 1 and an event is generated. The expecting latency is about 2 seconds. The step counter sensor has more latency. It is up to 10 seconds. But this sensor provides more accuracy than the step detector sensor.

4.2.4 Gravity Sensor

Gravity [9, 16] sensor measures the force of gravity that is applied to a device on all three physical axes. It is the same coordinate system it is used by acceleration sensor. Gravity sensor's unit of measurement is m/s^2. The difference between gravity sensor and accelerometer occurs to the fact that accelerometer gives the sum of all forces are applied to the device. Contrary the gravity sensor returns only the influence of gravity. Especially it indicates the direction and magnitude of gravity. When the device is at rest then the gravity sensor should return the same output with accelerometer. This sensor is a motion sensor and it is used for motion detection such as tilt and shake.

4.3 Methodology for Android sensors combination

At the following table is represented a general algorithm for Android sensor combination. They are explained the basic steps which help a developer to retrieve simultaneously data from more than one sensors and to process them.

Algorithm 4.1 Retrieves Data Concurrently from two Android Sensors

```
Main Activity

  Sensor Manager variable declaration
  Sensor Listener variable declaration /*listener will get informed for
                                         Sensor changes*/

  onCreate  function starts
          Sensor Manager gets System Service
          Sensor gets Default Sensor type  /*takes a sensor type such as
                                             Accelerometer for example*/
          Sensor Listener starts
               onSensorChanged function starts

                    If(Sensor type == Sensor1)
                      Result1 = event's value

                    Else If(Sensor type == Sensor2)
                      Result2 = event's value

                    If(Result1 != null && Result2 != null)
                      Process and Combine Data
                      Return Virtual Sensor Result

               onSensorChanged ends
          Sensor Listener ends
  onCreate  ends

  onResume function starts
          Sensor Manager Register Listener for Sensor1
          Sensor Manager Register Listener for Sensor2

```

```
32.    onResume ends
33.
34.    onPause function starts
35.                Sensor Manager un-Register Listener for Sensor1
36.                Sensor Manager un-Register Listener for Sensor2
37.    onPause ends
38.
39.  Main Activity ends
```

This algorithm can be applied for every sensor. Also it can be applied for a larger number of sensors and not only for two of them.

In Table 4.1 are displayed the Virtual Sensor’s constants and the event’s values for every sensor.

Virtual Sensor	Sensor Class Constant	Event Value
Orientation Sensor	TYPE_ORIENTATION	Event.value[0] Event.value[1] Event.value[2]
Rotation Vector Sensor	TYPE_ROTATION_VECTOR	Event.value[0] Event.value[1] Event.value[2] Event.value[3]
Step Counter Sensor	TYPE_STEP_COUNTER	Event.value[0]
Step Detector Sensor	TYPE_STEP_DETECTOR	Event.value[0]
Gravity	TYPE_GRAVITY	Event.value[0] Event.value[1] Event.value[2]

Table 4.1: Software Sensor’s Constants and Event’s Values

Chapter 5

IODetector: A Generic Service for Indoor Outdoor Detection

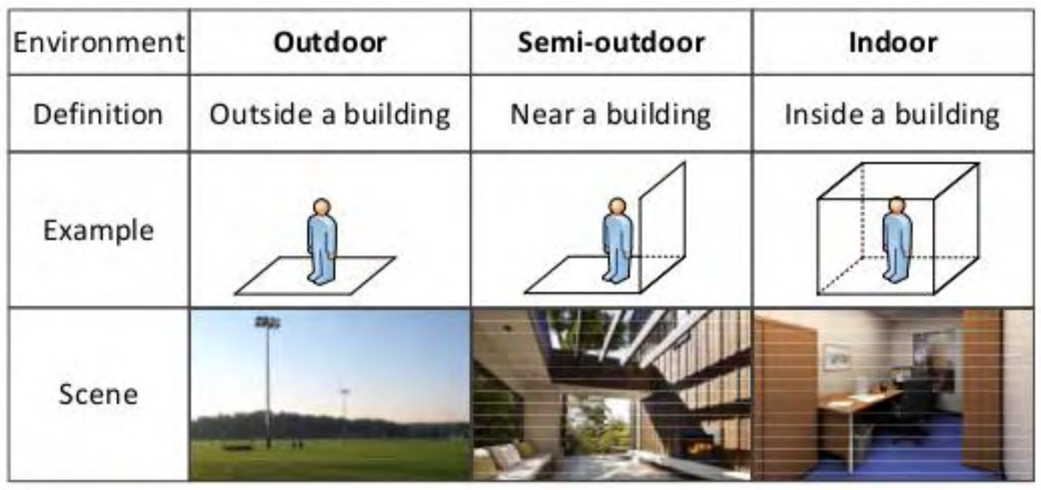
5.1 Introduction

In this chapter is analyzed a research [18] from Nanyang Technological University of Singapore which is referenced to indoor and outdoor detection using Android phones' sensors. This research is published in 2012 by Pengfei Zhou, Yuanqing Zheng, Zhenjiang Li, Mo Li and Guobin Shen. In this paper is presenting a sensing service which run on Android OS. This service named IODetector detects fast, efficient and accurate indoor/outdoor environments. IODetector uses common sensors which are available in almost all mainstream mobile phones. IODetector assumes no prior knowledge of environment and uses light sensor and magnetic sensor for detection. Except sensors it also uses cell tower signals.

5.2 Main Idea and Background

5.2.1 Main Idea

The main idea behind IODetector was the creation of a low energy consumption service which can effectively detect indoor and outdoor environment. IODetector are targeting to location based applications. An indoor/outdoor detection can provide to an application a variety of options. As an example, if an application uses GPS is extremely useful an indoor/outdoor detection. When an indoor environment is detected then the application can turn off the GPS because it is inappropriate for indoor places. IODetector is a light-weight and generic implementation. The use of low energy cost sensors such light sensor, magnetic sensor and cellular module makes it light-weighted. The energy costs of these sensors are far lower than the cost of GPS module or the cost of Wi-Fi module. The detection is about indoor environment, outdoor environment and semi-outdoor environment as they are displayed at the following picture:



Environment	Outdoor	Semi-outdoor	Indoor
Definition	Outside a building	Near a building	Inside a building
Example			
Scene			

Figure 5.1 from [18]: Three indoor/outdoor environment types and their representative scenes from [18].

In this research is observed that the light intensity, the cell tower signal and the intensity of magnetic field exhibit distinct patterns in the indoor and outdoor environments. With these patterns is possible the classification of different environments. More precisely:

- Light intensity is completely different when it captured inside a building rather than it captured outside. This due to the difference between the man-made lights and the natural one. The biggest challenge of using ambient light intensity is its variation over the time. The measurements are influenced by people movement, by phone position and by cover of sight.
- Cell tower signal which is received by a mobile phone changes dramatically from outdoor to indoor. This happens because of the walls in an indoor environment [19]. The walls block the line of sight paths between mobile device and cell tower. The variation of signal strength at different mobile phone models is the bottleneck in measurements. This factor makes difficult the classification of indoor and outdoor environments.
- The intensity of magnetic field has different behavior in indoor than in outdoor environments. Inside a building the intensity of magnetic field varies significantly when a person walks across this building. This behavior due to steel structures and electric appliances which are met in an indoor environment. On the other hand the variance of magnetic field in outdoor places is significantly less. An accurate reading of magnetic field intensity requires a careful calibration. Otherwise an error-prone will be included in readings.

The readings of these three sensor units are combined to produce an accurate and effective detection result but with low energy cost at the same time.

5.2.2 Background

The problem of indoor/outdoor recognition had not been thoroughly studied before the IODetector research. Applications which needed indoor/outdoor place detection used the GPS module for this reason. It is known that GPS performs poorly in an indoor environment. The unavailability of GPS is a result of blocking the line of sight paths to GPS satellites. So, the usage of GPS signal was an easy way to detect indoor/outdoor environment. However, typical GPS modules spend large amounts of energy and they need much time to conduct the GPS satellites. These two factors set them inappropriate because performance and low energy consumption are two basic parameters in software/hardware design nowadays. Except of the GPS there were some other works which tried to detect localization by sensing the surrounding environment. More precisely these works used fingerprinting ambient signals such as floor color, sound or user’s movement. But these fingerprints were needed a central server which first stored them and later it answered to users’ queries. For this reason works like these were unsuitable for a mobile phone which could not be connected all the time to internet. Another factor which indicates their unsuitability was the problem with image recognition. The problem of the indoor/outdoor image classification due to the fact that the images could not be applied automatically but they need manual input by user. As it becomes understood, there was the need of a generic and light weighted indoor/outdoor detector such as IODetector. IODetector needs to meet the following four design requirements:

1. *High Accuracy* because IODetector’s result will be used from other applications about human navigation, about logical localization and for power management.
2. *Prompt Response* is a second requirement. An outdated result generally is less valuable information for other applications.
3. *Energy Efficiency* is also an important design factor. Mobile devices’ energy budget limitation leads to an energy efficient design which uses inexpensive sensing resources.
4. *Universal Applicability* is the forth requirement for an efficient design. To reach this goal IODetector should not use prior knowledge or user’s feedback. Moreover it should avoid using of special sensors to ensure wide applicability.

5.3 System Design

5.3.1 System Overview

To meet design requirements IODetector uses sensors for detection but it also uses some sensors to ensure the correctness of the detection from the first sensors.

Except sensors use, IODetector uses some others services such as time. Figure 5.2 illustrates the system architecture of IODetector.

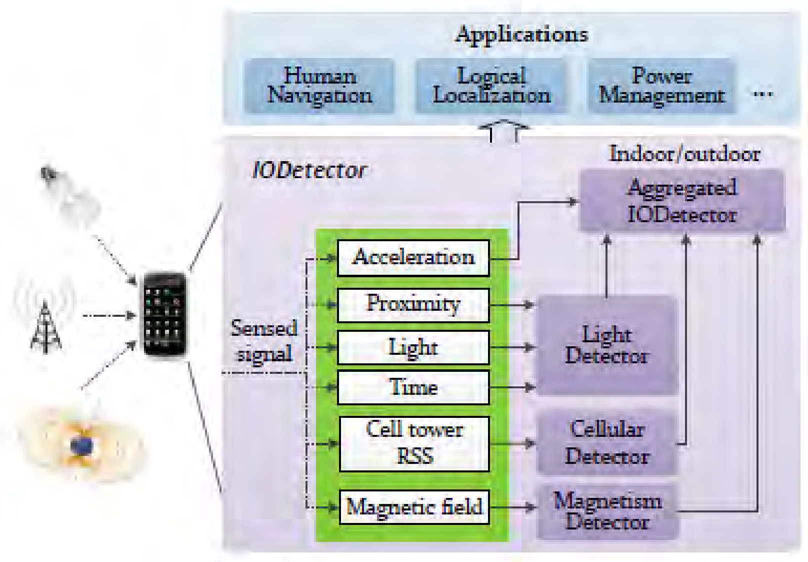


Figure 5.2: System Architecture of IODetector from [18]

As seen from the Figure 5.2, IODetector consists of three sub-detectors. They are Light Detector, Cellular Detector and Magnetism Detector.

- Light detector uses light sensor for measurement of ambient light. This measurement is compared with time. Furthermore light detector uses proximity sensor to secure a realistic result from light sensor.
- Cellular detector detects the reduction in of cellular signals caused by obstacles as walls.
- Magnetism detector takes advantage of the disturbance of magnetic field inside building or near to them. So it can distinguish indoor and semi-outdoor environments from outdoors.

The patterns of these three components in different environment are empirically studied. Every one of the above components has advantages and disadvantages in different environments. Thus, the IODetector aggregates the individual results before it generates its final decision. Before this, IODetector uses accelerometer to detect if the user moved. Only if he moved, the results of cellular detector and magnetism detector are appropriate for usage.

5.3.2 Light Detector

1) Light Intensity Measurements

The sunlight is the primary light in outdoor environments during the day time.

On the other hand in the indoor places are met artificial light sources such as fluorescent lamps. The light detector based on the fact that the light intensity in the outdoors and the semi-outdoors places are much higher than the intensity in the indoor places. This assumption is true even in rainy or cloudy days. The major reason for this difference is the higher intensity of sunlight in visible spectrum against the ordinary lamps' intensity. Furthermore, light sensor is able to detect light also in invisible spectrums like infrared and ultraviolet. So, even when the brightness looks like the same, the light sensor can detects the sunlight's luminous flux which is much higher. That means that light sensor can offer an accurate and secure result. At the night the indoor environments' light intensity is higher than in the outdoor environments. Experiments show that during a whole day the indoor light intensity varies from 100 to 1000 lux.

2) Detection Process in Light Detector

First the availability of light sensor is examined. For this examination the proximity sensor is used. This sensor detects the presence of nearby objects which may block the light sensor. The result of proximity sensor is an answer, if the light sensor is available or not. Light detector does not care about the exact distance between phone and an object. It wants to know only if an object is close enough to set light sensor inappropriate. So, when the light sensor is available then it measures the light intensity *L*. When *L* is bigger than a threshold *σ1* that means that is an outdoor/semi-outdoor environment. If not, the light detector have to investigate if is day or night. When the clock indicates a daytime then the light detector confirms an indoor environment. In contrast, if clock does not indicate a daytime, the *L* value would be compared with a second threshold *σ2*. If $\sigma 2 < L < \sigma 1$, it indicates an indoor environment with confidence equal to $\frac{\sigma 1 - L}{\sigma 1}$. Else, when $L \leq \sigma 2$ the device is an outdoor/semi-outdoor environment with confidence equal to $\frac{\sigma 2 - L}{\sigma 2}$. Figure 5.3 summarizes the work flow of the light detector.

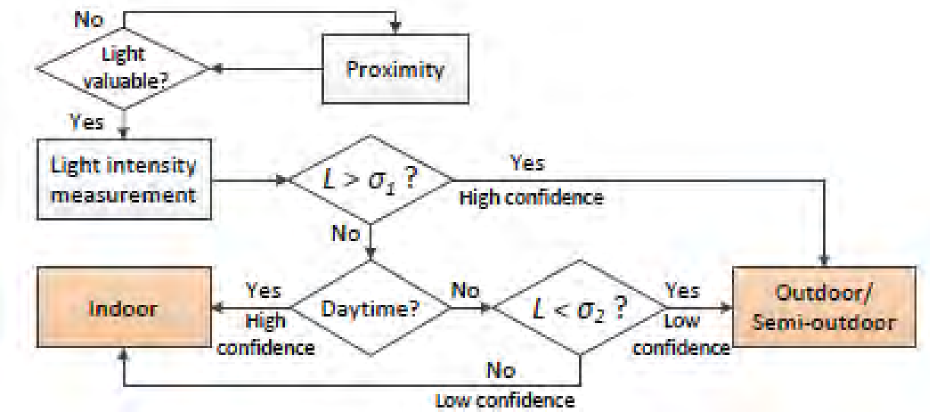


Figure 5.3: Detection flow of light detector component from [18]

5.3.3 Cellular Detector

Mobile phones are continuously connected to nearby cell towers. Thus, the collection of received cellular signal strength (RSS) demands negligible energy consumption. This is the main reason which explains why cellular signal picked over Wi-Fi signal for this paper. Since mobile phone has to stay connected for its basic communication, the cellular signal strength is available with no additional energy. Conversely a continuous scanning for other types of signal such as Wi-Fi leads to unnecessary energy consumption. In addition to this, signals such as Wi-Fi have short wavelength and suffer from the shielding effect of surrounding objects. Even the human body can bring noises to their detections. In contrast, cellular signal does not be influenced by these factors.

1) Associated Cell Tower Signal

The absolute value of cellular RSS provides limited information. This limitation comes from the fact that RSS varies across times, places and mobile phones models. Contrary to this, the variation of cellular RSS is able to offer efficient information for detection. So it is impossible to map RSS values to environments but it is possible to map RSS variation to different environments. Thus, the cellular detector is independent of mobile phone model. The variation in a semi-outdoor place is much bigger than in an indoor place or in an outdoor place. This fact helps cellular detector to differentiate semi-outdoor from indoor/outdoor environments. Generally this RSS variation is too effective in detection of transition between indoor and outdoor environment.

2) Visible Cell Signal Strength

The information of a single cell tower has some limitations. The RSS variation is not sure that shows a transition from indoor to outdoor place. It could be mean that phone just handover to another cell tower. Furthermore, it is possible that RSS variation due to corner effect [19]. This effect is too often in semi-outdoor environments because of the change of the line of sight to cell towers. For example this happens when a user turns into a corner. These limitations indicate the need for information from multiple cell towers. In general, a mobile phone is within the coverage range of multiple cell towers and connects to one with the higher signal strength. So, the information of the others cell towers can be available. The cellular detector takes advantage of this, using the RSS variations of all visible cell towers. As a result, the limitation of phone's movement between two towers is extinct. Furthermore the second limitation about corner effect is mitigated.
From algorithmic view, the cellular detector starts reading the RSS for every visible cell tower i. This RSS called R_i. The variation within a time interval Δ is $V_i(t) = R_i(t + \Delta)$

- $R_i(t)$. The variation threshold is named *v*. N_+ refers as the number of cell towers whose RSS increases more than *v*. N_- refers as the number of cell towers whose RSS decreases more than v respectively. The stability of cell tower RSS is defined as $N_o(t) = n - N_+(t) – N_-(t)$, *n* is the total number of visible cell towers. In IODetector, *Δ* is set to 10sec and *v* is set to 15dB. Generally, if a user moves from an outdoor environment to an indoor one, the RSS of the most of cell towers will be decrease and vice versa. This is what the algorithm which is described above, detects. Cellular detector limitation due to limitations in the number of visible cell towers. If this number is small, then the detector's performance may be poor.

5.3.4 Magnetism Detector

The disturbance of Earth's magnetic field is much larger inside a building rather than out of it. Electric appliances and steel structures lead to this disturbance. They turn the geomagnetic field to electromagnetic field. Thus, it can be an efficient pattern for indoor/outdoor detection. The magnetism detector detects the variance of magnetic field within τ time. Then it examines if the variance is bigger than a threshold α. When it is bigger magnetism detector detects indoor/semi-outdoor environment. Otherwise, it detects outdoor environment. The selection of threshold α is critical. If it is small, then the indoor/semi-outdoor environments will be detected extremely well. The problem is that many outdoor places will be detected as indoor places too. On the other hand, if α is too high then the detection of outdoor places will be correct but the detection of indoor/semi-outdoor environments will not be. In IODetector the threshold α is set to 18 and τ is equal to 10.

5.3.5 Aggregated IODetector

The three sub-detectors have advantages and disadvantages. Their common advantage is that they can detect the ambient environment. In contrast, their disadvantages vary. More precisely:

- The light detector needs free space for detection. So it is unable to make detection inside pockets or bags.
- The cellular detector needs sufficient cell tower coverage. Otherwise, its detection would not be confident and efficient.
- The magnetism detector needs the user's movement. If user is static, then the magnetism detector is unavailable.

The solution to these problems from the use of individual sub-detectors is the sub-detectors' integration. Firstly the sub-detectors run alone and they produce their detection results. These results are a triplet of confidence levels. There is one confidence level for every one of the three environments. After this, the confidence levels from all three sub-detectors are summed. The highest summed confidence level indicates the final decision. In other words, the environment with highest confidence level is the one which is detected by IODetector. Figure 5.4 shows the stateless IODetector composed of the three sub-detectors. The pair *D*, *C* in each sub-detector is its final decision and its set of confidence levels respectively.

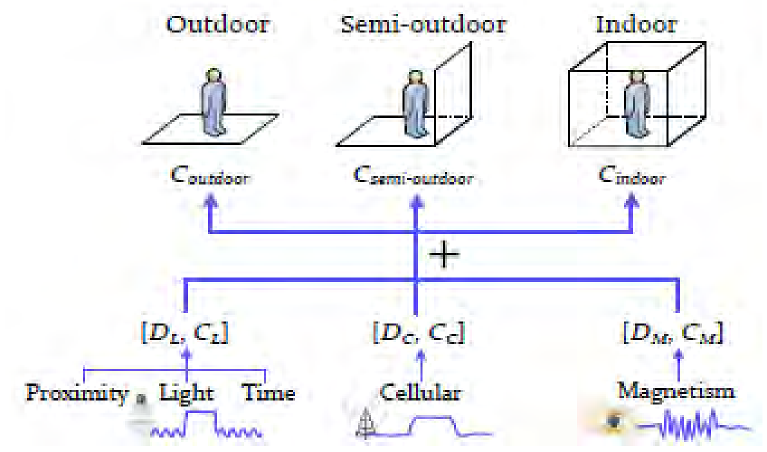


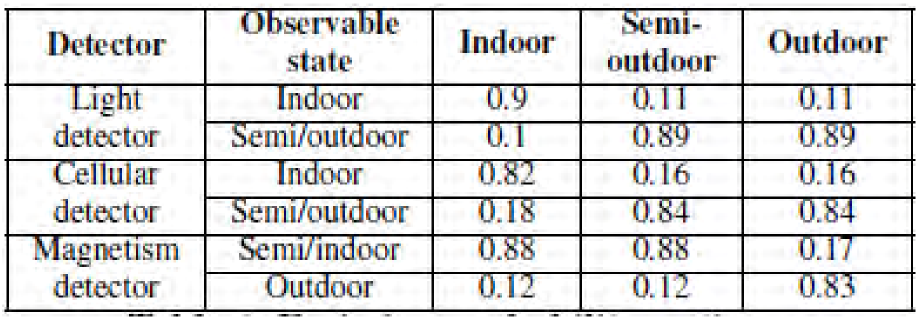
Figure 5.4: Stateless IODetector from [18]

The stateless IODetector does not consider previous results. This could be a disadvantage. The previous results are able to give more information and they will improve the detection's performance. There are three basic parts for the creation of a non stateless IODetector:

1. All the sub-detectors produce continuous and sequential results. These results are hidden states. The hidden states are integrated using Hidden Markov Model (HMM) [20]. HMM is able to estimate the most likely sequence of hidden states. So IODetector will infer the most likely hidden state. The Viterbi algorithm [21] is applied to estimate the most likely environment in the HMM according to sub-detectors results.
2. Also it was possible the observation of some transition probabilities. For example, when a user is in an indoor environment, the most possible is that after 10 seconds the user will stay in an indoor place or the user will be in a semi-outdoor place. Generally the transition probability *T* from one environment to another is:
   - *T(S, I) = T(S, S) = T(S, O) = p1 = 1=3.*
   - *T(I, I) = T(I, S) = p2 = 1=2.*
   - *T(O, O) = T(O, S) = p3 = 1=2.*
   - *T(O, I) = T(I, O) = p4 = 0.*

Elaborated as I for Indoor, S for Semi-outdoor and O for Outdoor.

3. The emission probability is the probability an observable state to be observed in a specific environment. Table 5.1 describes the emission probability for IODetector.



Detector	Observable state	Indoor	Semi-outdoor	Outdoor
Light detector	Indoor	0.9	0.11	0.11
	Semi/outdoor	0.1	0.89	0.89
Cellular detector	Indoor	0.82	0.16	0.16
	Semi/outdoor	0.18	0.84	0.84
Magnetism detector	Semi/indoor	0.88	0.88	0.17
	Outdoor	0.12	0.12	0.83

Table 5.1: Emission probability settings from [18]

The stateful IODetector needs all the sensors will be continuously working. As a result it may consume extra energy than the stateless IODetector. However, the stateful IODetector provides better detection accuracy than the stateless one.

5.4 Evaluation

5.4.1 Experimental Methodology

For the needs of this paper were collected data in 84 different sites; 23 of them were outdoor segments, 27 were semi-outdoor segments and 34 were the indoor segments. The user was walking along these places and the mobile devices were performing continuous detections. The duration of this collection was over one month and the data were captured from 5.00 to 22.00. Three different mobile phone modes were used:

1. Samsung Galaxy S2 i9100. It is equipped with a dual-core 1.2GHz Cortex A9 processor and 1GB RAM.
2. HTC Desire S. It has a single-core 1GHz Scorpion processor and 768 MB RAM.
3. HTC Sensation G14. It has a dual-core 1.2GHz Scorpion processor and 768 MB RAM.

All these models were equipped with magnetism sensor, light sensor, proximity sensor and accelerometer.

5.4.2 Performance of Sub-detectors

The three individual sub-detectors were tested independently. Each sub-detector reported the environment type with the highest confidence level. The results are presented in Figure 5.5. As seen, the light detector and the cellular detector have higher accuracy than the magnetism detector. Moreover, the light detector and the cellular detector perform higher accuracy when they detect an outdoor/semi-outdoor place than when they detect an indoor environment.

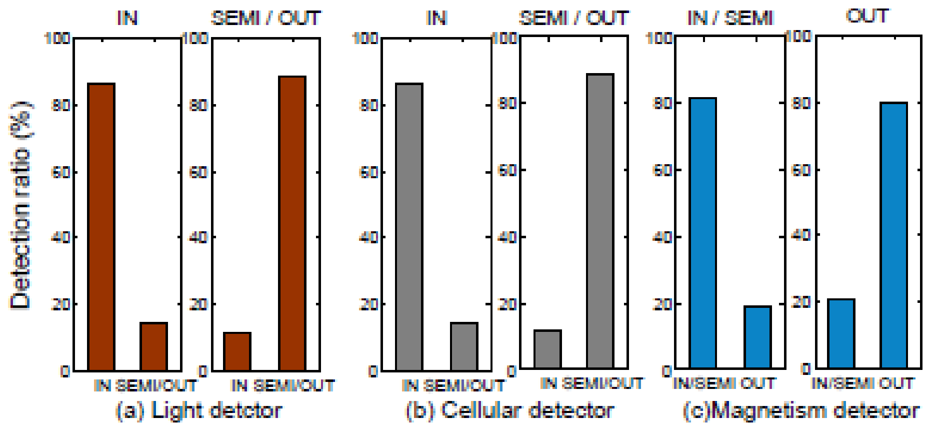


Figure 5.5: Sub-detectors' Performance from [18]

5.4.3 Performance of aggregated IODetector

There were two approaches about performance measurement of IODetector which were studied. The first was the stateless IODetector and the second was the stateful IODetector. It was found that the accuracy of the stateful IODetector was higher than the accuracy of the stateless IODetector. Also, it was found that IODetector had better performance than the individual sub-detectors had. Figure 5.6 shows these results.

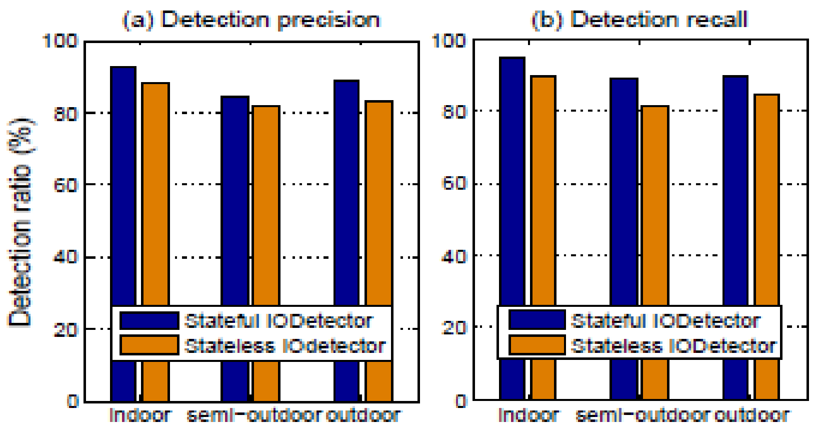


Figure 5.6: Stateful' and Stateless' IODetector Performance from [18]

Also it was found that IODetector did not introduce a large energy overhead. Every one of the three sub-detectors consumed a little bit of energy. This extra energy consumption did not affect significantly to the phones' battery duration.

Chapter 6

Indoor/Outdoor Detector Implementation

6.1 Basics of the implementation

This chapter presents the implementation of an indoor/outdoor environment detector. This implementation has the following assumptions and conventions:

- The implementation called InOutDetector
- InOutDetector is implemented in application layer framework
- InOutDetector is designed for Android devices
- It is developed in Eclipse IDE [22] using Android SDK Tools [23]
- Its target is Android Jelly Bean
- InOutDetector does not need information from other applications or from a database or from network.
- If the user turns off the application, the InOutDetector is terminated.
- If the user set the application in the background then the InOutDetector unregister the sensors and it stops reading
- InOutDetector returns an Indoor/Outdoor prediction and the percentage of its prediction.

6.2 Using built-in sensors

For InOutDetector implementation are used almost all of the sensors that are available in a common Android device. All these sensors are used for two reasons. The first reason is for detection and the second is to be ensured the availability and the correctness of detection. More precisely the sensors are used:

- *Light Sensor.* This sensor measures the light intensity. Light sensor is extremely effective to distinguish an indoor from an outdoor environment especially in the day time.
- *Magnetic Sensor.* This sensor measures the intensity of magnetic field in device reference.

- *Proximity Sensor.* This sensor used to ensure light sensor's availability. If the proximity sensor detects an object very near to mobile phone, means that light sensor is blocked and it cannot measure light intensity.
- *Accelerometer.* The magnetic detector and cellular detector returns useful results only if the user is moving. Thus, accelerometer detects user's motion and ensures the correctness of these two sub-detectors' results.
- *Gravity Sensor.* This sensor helps the transformation of magnetic field from device to global reference frame.
- *GetRotationMatrix*. As the gravity sensor, getRotationMatrix is used to convert magnetic field from device into global reference frame.

Except sensors, other one built-in component which is used for the implementation of InOutDetector is the cell tower's signal receiver. This receiver measures the signal's strength which is received from the cell tower of the network provider.

6.3 Implementation's details

6.3.1 Sub-detectors

Three sub-detectors have been implemented. Detection happens every 10 seconds. In these 10 seconds the data of built-in sensors were collected. Then every one of sub-detectors read these data and produces its own detection for indoor/outdoor environment.

1) Light Detector

Light detector takes its decision comparing the light intensity value with a threshold. This threshold has been set empirically after many experiments in different environments and different day times. Furthermore the threshold is completely different in the day time than in the night time. The day threshold is much higher than the night threshold. During the day when the light intensity is higher from threshold then the light detector predicts an outdoor environment. In contrast, at the night, a light intensity higher than threshold means an indoor environment. The threshold for day time is set to 1000. Figure 6.1 shows the light intensity in indoor environments and in outdoor environments. On the other hand, the threshold for night time is equal to 4. Except these two cases, it is possible for light detector to be inappropriate for detection. This happens when proximity sensor detects an object close to the mobile device. The following page's flowchart explains in a better way the cases were described above.

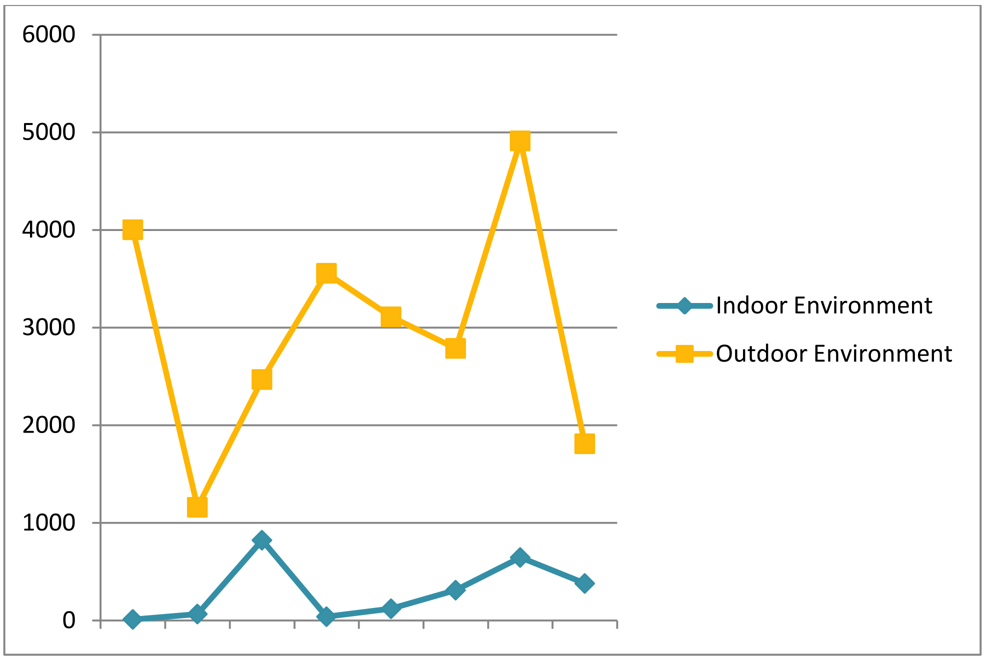


Figure 6.1: Light Intensity in day time

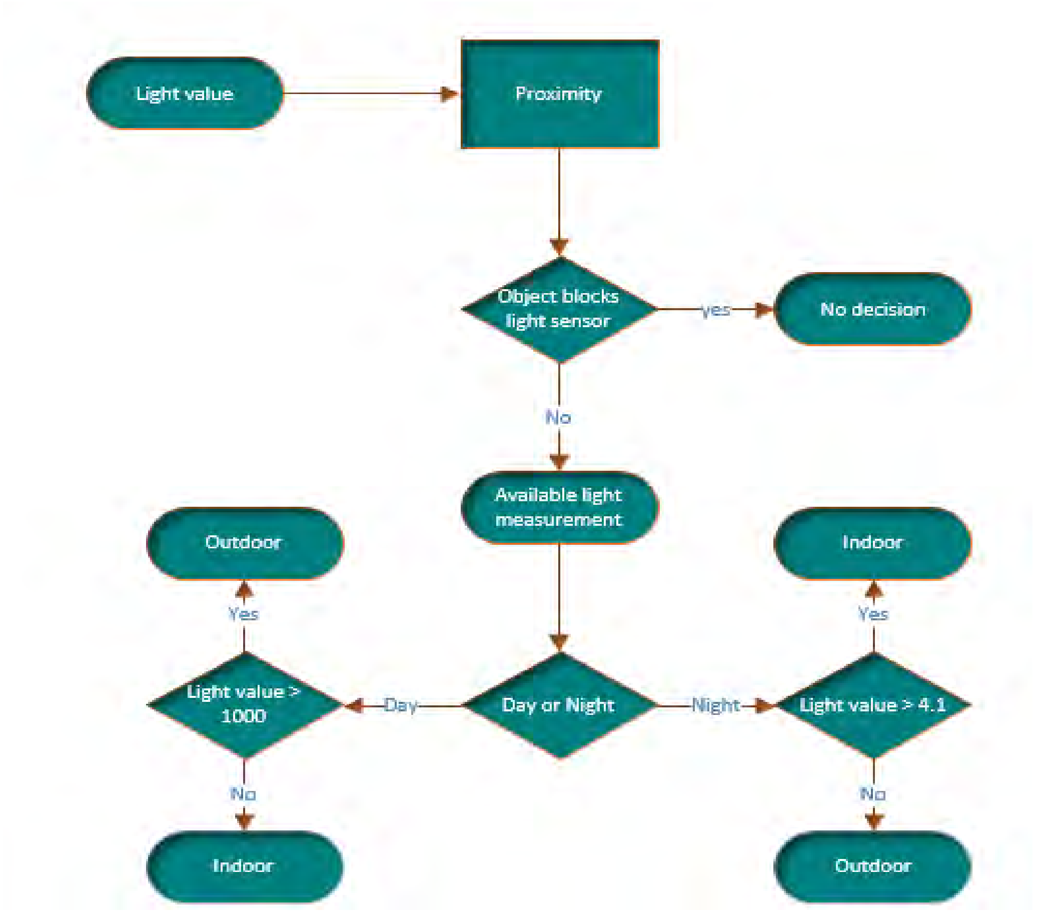


Figure 6.2: Light detector's decision flow

2*) Magnetic Detector*

The magnetic detector distinguishes indoor/outdoor environments using the variance of magnetic field. If the variance is higher than the threshold then the magnetic detector detects an indoor environment. Otherwise, an outdoor environment is detected. Also in this case, the threshold is set empirically after experiments. Its value is 1.5. Figure 6.3 displays Magnetic field's variance.

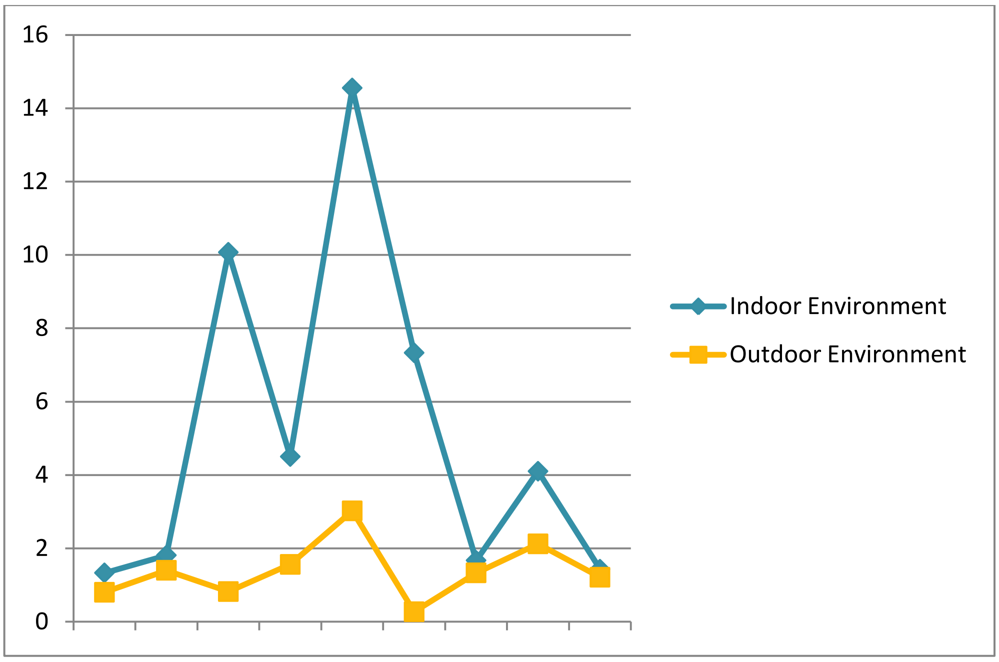


Figure 6.3: Magnetic field's variance

But before this decision, some steps for the calculation of magnetic field's variance are needed:

- The magnetic sensor measures magnetic field value in three axes (x, y, z). But these three values are device referenced. That means that they would change if the device is turned but the user stays in the same position. This is undesirable in current implementation. So, the values must turn to global reference frame. This transformation will minimize this undesirable phenomenon. For this reason the computation of rotation matrix is needed. In Android, the rotation matrix is computed by calling getRotationMatrix. This function uses the values of magnetic sensor and gravity sensor as input and returns the rotation matrix as output. After that, it is possible the conversion of magnetic field vector with basis (x, y, z) to a magnetic field vector with a global referenced basis (wx, wy, wz). The magnetic field vector with global referenced basis is the cross product between the rotation matrix and the values that the magnetic sensor has measured.

- Also the magnetic field strength must be computed. The three values of the vector represent the magnetic field in the three axes. But the magnetic detector needs a general magnetic value and not three specific values in many axes. The asking value is the magnetic field strength. It is computed with the following mathematical type *: Strength* = $\sqrt{wx^2 + wy^2 + wz^2}$ [24]
- In the last step, the variance of magnetic field strength is computed using the type: *variance (Strength)* = $E\ \{(Strength - E\ [Strength])^2\}$ [25], *E* is the average value of the magnetic field strength.

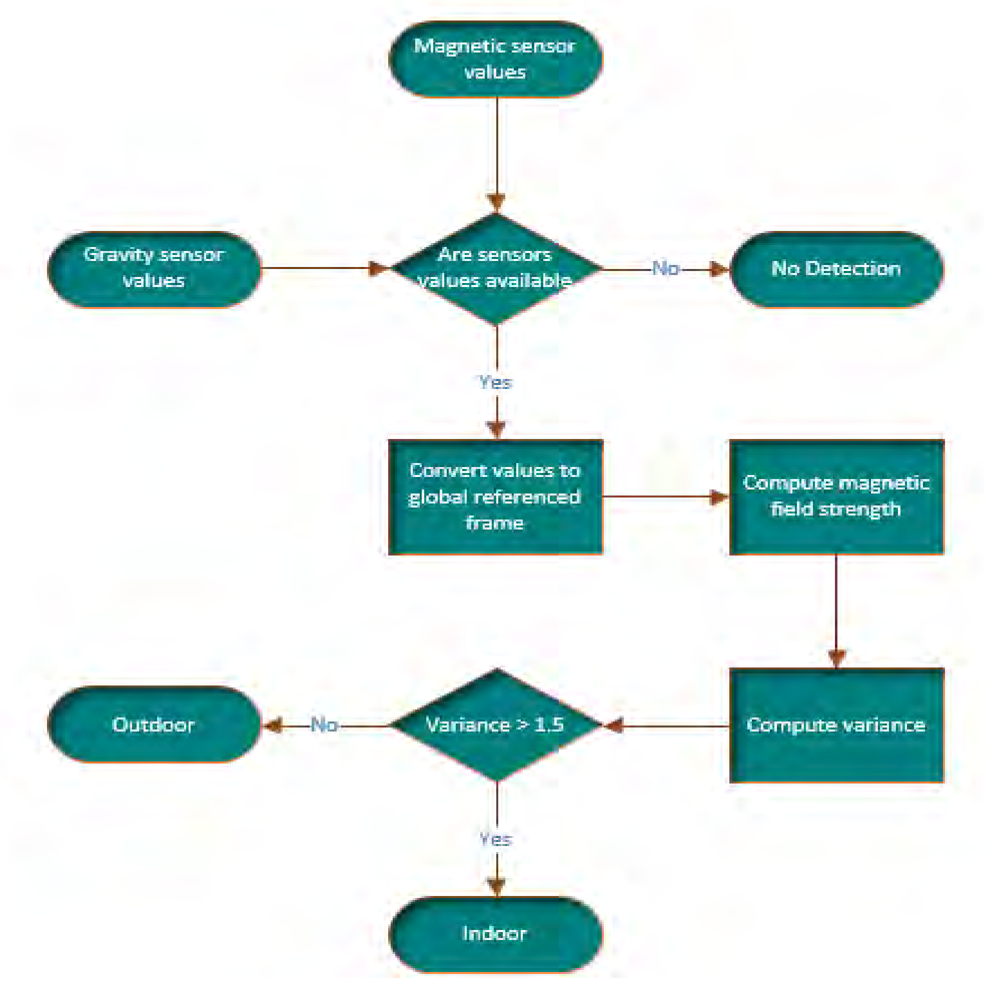


Figure 6.4: Magnetic detector's decision flow

3) Cellular Detector

Cellular detector uses the signal strength of cellular tower to distinguish the indoor environment from the outdoor one. For this implementation is used only the signal of the connected cellular tower and not the signals from all the visible towers. The second approach which offers data from more than one cellular tower is more accurate than the first approach. The drawback of receiving data from all visible cellular towers is that this functionality is not provided from Android OS for all devices. More precisely this functionality exists but in many devices it does not returns the information of the neighbor cellular towers. Thus, the cellular detector will not work in many devices. So, the cellular detector follows the first approach using the signal strength of the connected tower. The cellular detector compares the GSM signal variance with a threshold. If the variance is higher than the threshold then an indoor environment would be detected. The threshold was found empirically and it is 0.22. Figure 6.5 shows cell signal's variance. The variance is computed using the type: *variance (Strength) = E {(Strength – E [Strength])^2}* [25], *E* is the average value of the signal strength. Unfortunately the variance alone does not produce accurate results. When the user moves from an indoor environment to an outdoor environment the variance is much higher than the threshold and the cellular detector will recognize wrongly an indoor environment. To avoid this wrong prediction the cellular detector computes the trend of cellular tower's signal. If the trend increased that means there is a movement from indoor environment to outdoor environment. This ensures a more accurate result of cellular detector.

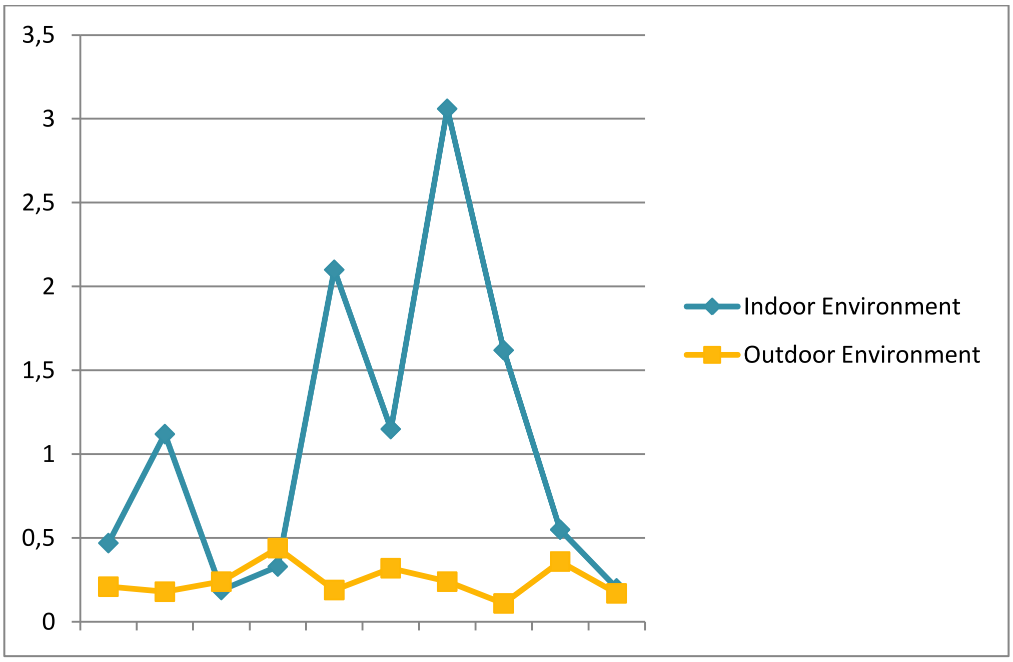


Figure 6.5: Cell signal's variance

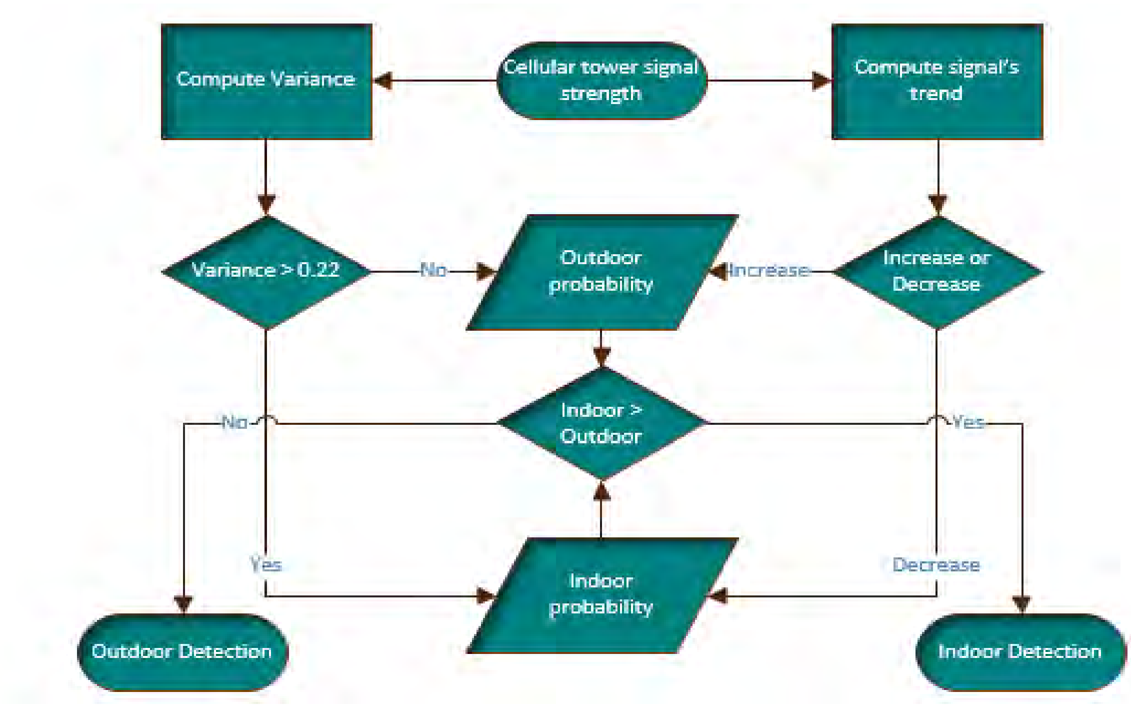


Figure 6.6: Cellular detector's decision flow

6.3.2 Aggregated InOutDetector

This is the final stage of the implementation of InOutDetector. The results of the three sub-detectors' results are combined and produce the final decision of indoor/outdoor environment detection. After experiments were found that light detector produces more accurate results than the other two sub-detectors. Thus, its decision is considered more than the decisions of other two sub-detectors. The light detector is able to distinguish outdoor/semi-outdoor places from indoor places with higher confidence than the others sub-detectors. For this reason its decision is preferred. InOutDetector decision is a prediction between indoor environment and outdoor environment. Also InOutDetector returns the percentage of its prediction. For example, if InOutDetector predicts an outdoor environment with high percentage for its prediction that means all the sub-detectors predict outdoor environment. On the other hand, if InOutDetector predicts an outdoor environment with low percentage of prediction means that the magnetic detector and cellular detector predicted indoor environment and the light detector just cover them. This environment, with high light intensity as outdoor place and high variances of magnetic field and cell tower's signal as an indoor environment, is a semi-outdoor environment. This is the way for InOutDetector to distinguish outdoor of semi-outdoor environments. In the case of the light sensor is blocked, is impossible for InOutDetector to take a decision between outdoor and semi-outdoor environments. Furthermore, when the light sensor is blocked and the user is not moving at the same time then the InOutDetector is unable to make any predictions. For the detection of user's movement, InOutDetector adopts a function which counts the steps of the user. The code of this function is developed from Mr. Artem Tartatakynov. This code is embedded to InOutDetector with very small changes and offers an efficient detector of the user's motion. In the next page's Figure 6.4 is presented the decision flow of InOutDetector. The InOutDetector which is implemented is stateful. The InOutDetector produces a prediction every 10 seconds. During these 10 seconds, it is keeping results of all sensors readings and finally the InOutDetector uses them for its prediction.

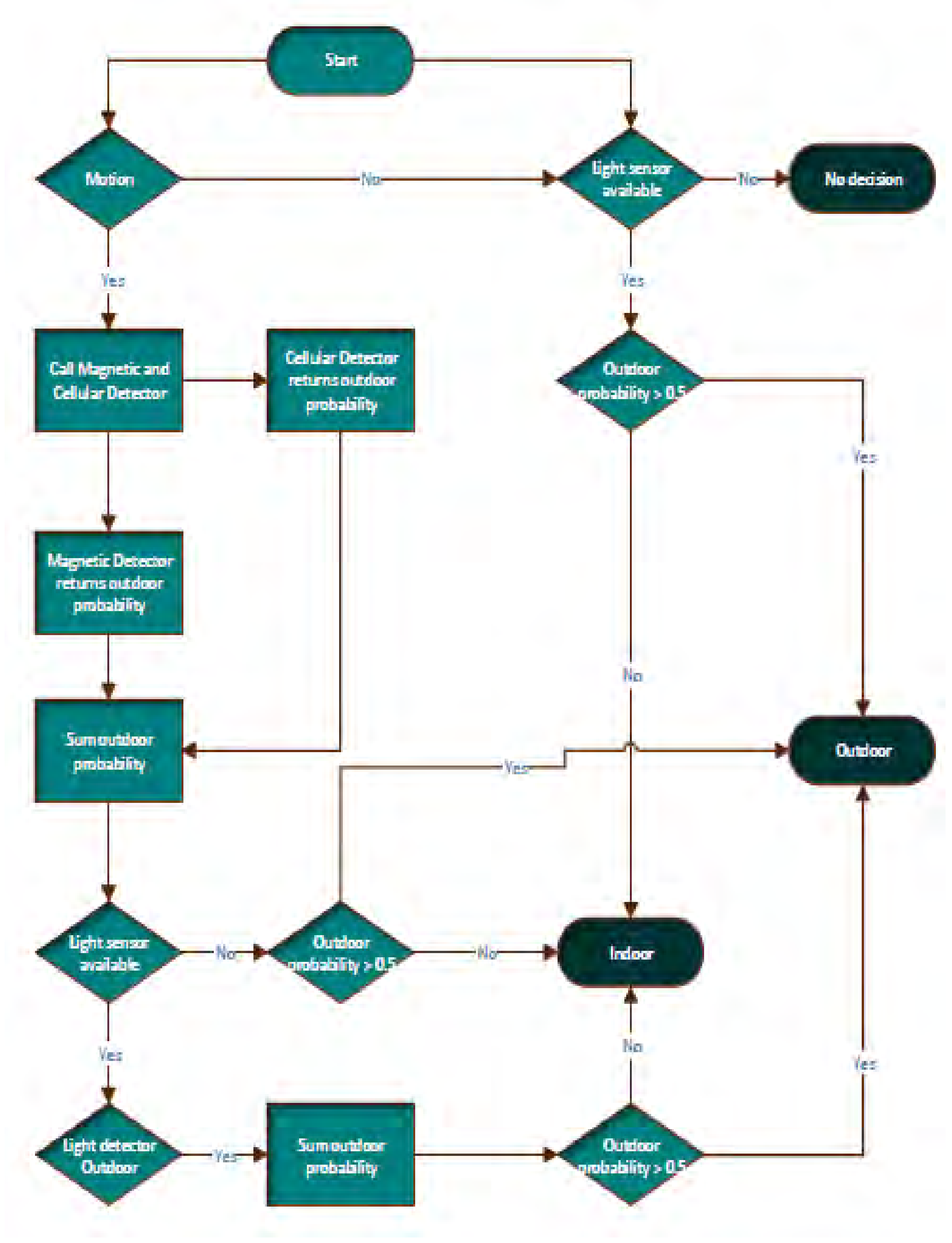

Start
Motion
No
Light sensor available
No
No decision
Yes
Yes
Call Magnetic and Cellular Detector
Cellular Detector returns outdoor probability
Outdoor probability > 0.5
Yes
Magnetic Detector returns outdoor probability
No
Sum outdoor probability
Yes
Outdoor
Light sensor available
No
Outdoor probability > 0.5
No
Indoor
Yes
No
Yes
Light detector Outdoor
Yes
Sum outdoor probability
Outdoor probability > 0.5

Figure 6.7: InOutDetector decision flow

Chapter 7

Experimental Results and Conclusions

7.1 Experimental Methodology

For the needs of this thesis were collected data from 19 different places. 9 of them were outdoor/semi-outdoor places and the other 10 were the indoor places. The selected places were attempted to be places with different characteristics. As an example, for indoor environment were collected data from places such as home's rooms, a garage, a loft, an office, a bar and a shopping center. Conversely the places for outdoor environments data collection were balconies, a garden, village's streets, big city's streets and a farm. All the data were collected in a time space of 2 months. The data were captured from 8.00 to 23.00. The mobile device which used was a Samsung Galaxy Nexus. It is equipped with a dual-core 1.2GHz, Cortex A9 processor and 1GB RAM. Of course it is equipped with all the needed built-in sensors.

7.2 Sub-detectors' Performance

In this section the three individual sub-detectors were tested independently. Firstly the experimental results of light detector are presented. The experiments were about the total performance of light detector or about the individual performance of light detector in day time or in night time. All these experimental results are presented in the graphs of the following page. As seen, the light detector has higher performance in the day time that in the night time. Especially in the case of an indoor environment which is examined. This result comes from light sensor's inability to capture very low artificial lights. Except the cases which the user is in an indoor room with no light or in a bar which is almost totally dark; it is possible the artificial light of a room to be so low that the light sensor cannot capture its value. In all these cases the light detector detects wrongly these environments as outdoor. For this reason the performance of light sensor which can be about 95% for day time decreases to 80% for night time for indoor environments detection. The third graph shows that the overall performance of light detector is over 86% for indoor places detection and over 90% for outdoor places.

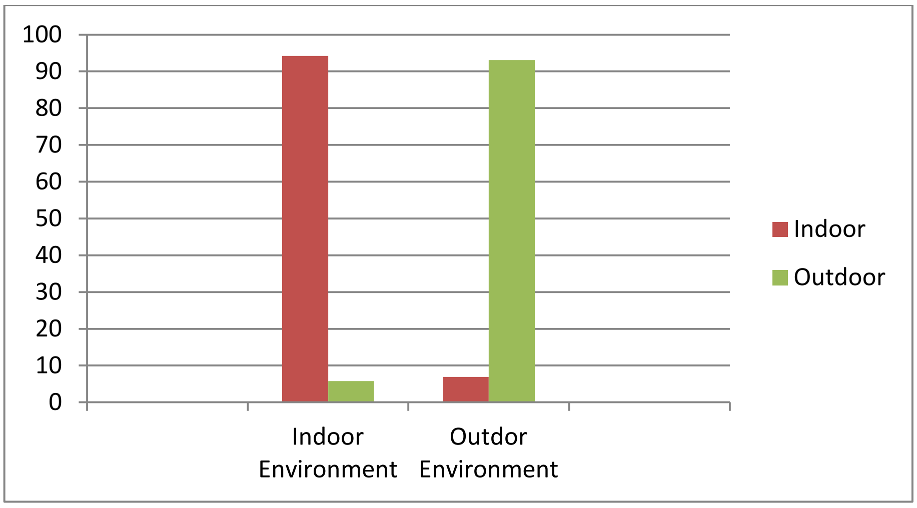


Figure 7.1 Light Detector's performance in day time

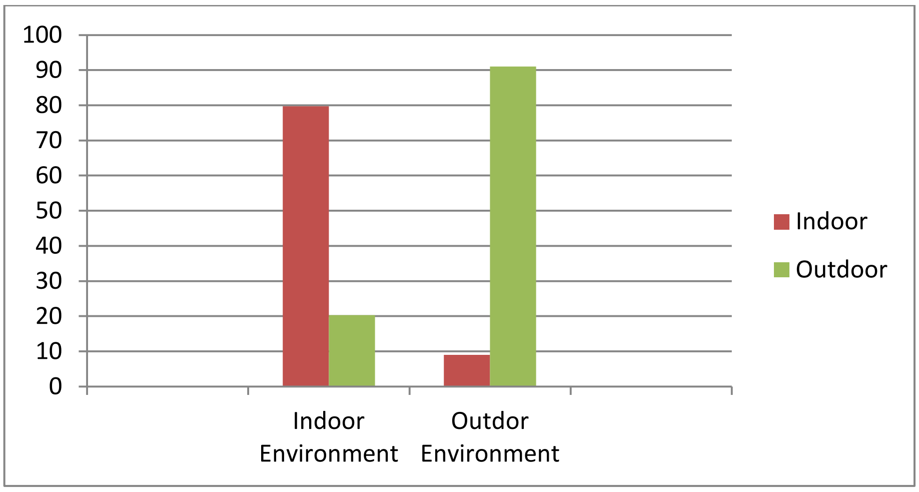


Figure 7.2 Light Detector's performance in night time

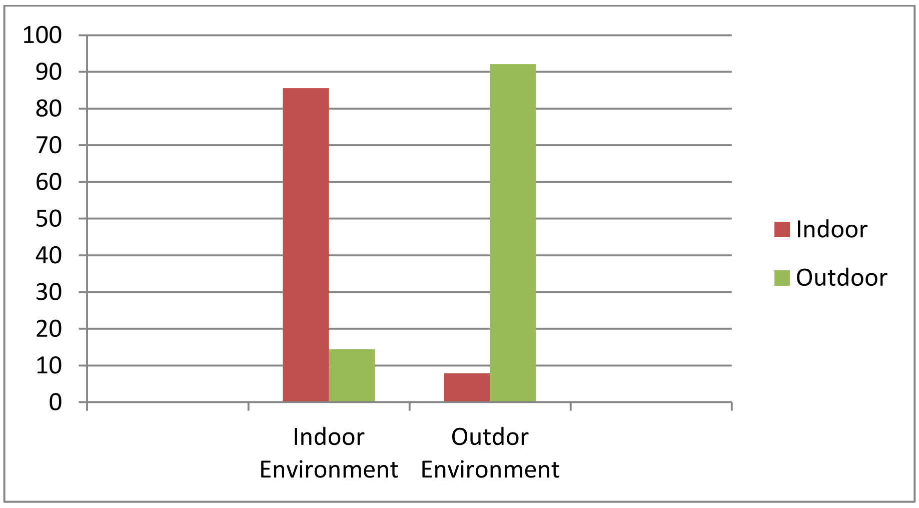


Figure 7.3 Light Detector's overall performance

In this page is presented the performance of magnetic detector. In this detector there is no difference between day and night. So only the total performance of magnetic detector is measured. As seen in the next figure the magnetic detector's performance is much lower than the performance of light detector. The performance of magnetic detector is less than 70% when the overall performance of light detector is almost 90%. This is a result of magnetic field's unpredictable behavior. It is very hard to set a threshold for magnetic field because it is difficult to find a pattern for its behavior. This makes the classification of indoor/outdoor environments a very hard work. There are many indoor places with lower variance of magnetic field than the outdoor places. Normally, exactly the opposite must happen. For example, in an outdoor environment such a city street with many shops and cars the magnetic field's variance was counted much higher than the variance inside a normal home. Except the difficulty to pattern the different environments there are factors such as the mobile device's movement or the electronics of the device itself which affect to the magnetic field.

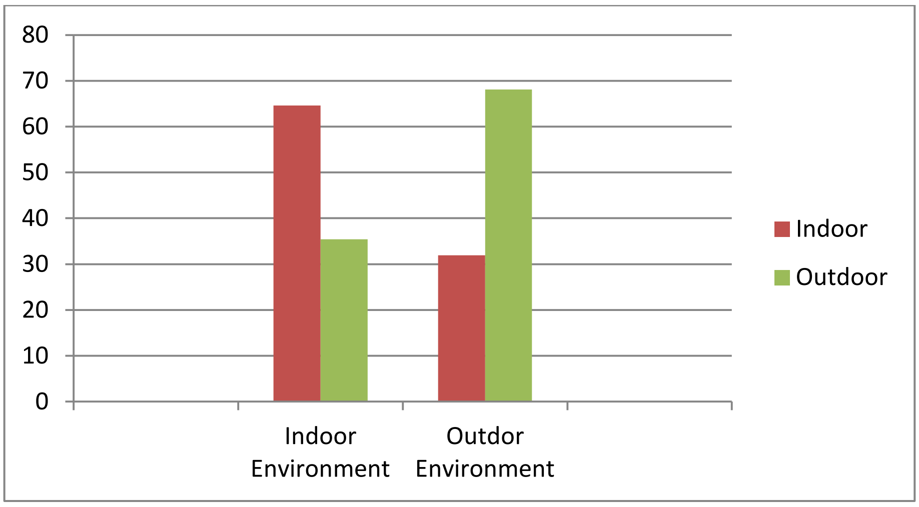


Figure 7.4 Magnetic Detector's performance

Last, the performance of cellular detector is measured. Its performance is not as high as the performance of light detector but it is higher than the performance of magnetic detector. The cellular detector's performance is about 80%. So, this sub-detector can be characterized very efficient. Figure 7.5 presents the cellular detector's performance.

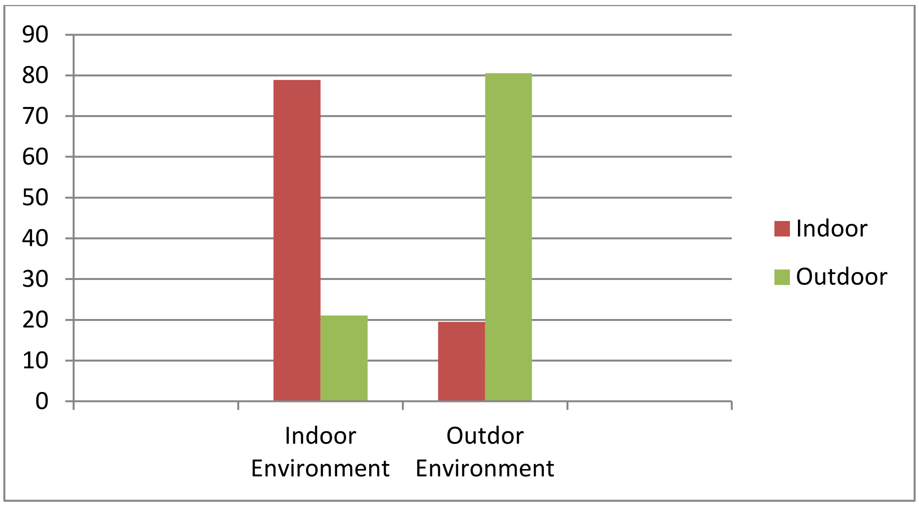


Figure 7.5 Cellular Detector's performance

7.3 InOutDetector's Performance

In this section is presented the performance of InOutDetector. Its performance is between 84% and 91%. Comparing this performance with the performance of IODetector which is described in a previous chapter can see that the InOutDetector of this thesis is a little bit more efficient. Especially this is visible in the detection of outdoor environments. As seen the maximization of light detector's usage improves the overall performance.

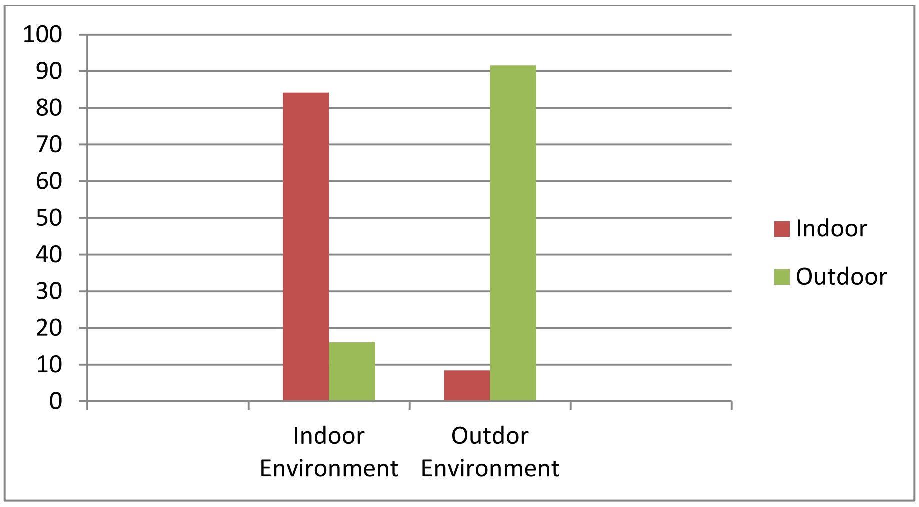


Figure 7.6 InOutDetector's performance

7.4 Conclusions and Future Work

After the experimental results’ observation and after a general overview of this Master thesis the following conclusions can be extracted:

- Android sensors can offer many features to applications. Also they can open new ways for developers.
- The sensors’ behavior varies in different Android models. This fact makes their usage an extremely hard work.
- The energy consumption of sensors is low.
- A combination of Android sensors can offer an efficient mechanism for indoor/outdoor environments detection. Its performance can be over 85%.
- There are many cases in which an indoor/outdoor detector can be completely wrong in its detection.
- The light intensity presents a consistent pattern for the classification indoor/outdoor environment.
- The magnetic field is quite unpredictable.
- Programming with sensors demands a lot of time for measurements taken and observation. It is very hard to understand their behavior and find appropriate patterns of this behavior.

Some improvements in InOutDetector can be added in a future work. They will be three specific improvements.

1. The possibility of cellular detector to choose its own politic for detection would be added. If the mobile device offers information about all visible cell towers then this information will be used in detection. Else if this information is not available then the approach with the connected cell tower will be chosen.
2. A fourth sub-detector will be added. This sub-detector will detect the temperature of an environment. The detection’s measurements will be compared with the information of a simple data base written in SQLite which will be built-in in InOutDetector. This sub-detector will be used only when the device is equipped with a temperature sensor.
3. If the mobile device is running an Android version 4.4 or newer the step counter sensor could be used instead of the function which detects user’s motion in current InOutDetector.

References

[1] Mobile Operating Systems Wiki, http://en.wikipedia.org/wiki/Mobile_operating_system

[2] Android OS Wiki, http://en.wikipedia.org/wiki/Android_(operating_system)

[3] Android Logo, http://www.famouslogos.us/android-logo/

[4] Android OS Evolution, http://www.xda-developers.com/android/the-evolution-of-android-part-i/

[5] Android Architecture – The Key Concepts of Android OS http://www.android-app-market.com/android-architecture.html

[6] Geoffrey Blake, Ronald G. Dreslinki and Trevor Mudge: A Survey of Multicore Processors [A review of their common attributes] (2009)

[7] Android Developers – Sensors Overview http://developer.android.com/guide/topics/sensors/sensors_overview.html

[8] Tamer Nadeem, Old Dominion University, Department of Computer Science: App Development for Smart Devices http://www.cs.odu.edu/~cs495/materials/Lec-05_Android-Sensors.pdf

[9] Motion Sensors, http://developer.android.com/guide/topics/sensors/sensors_motion.html

[10] Position Sensors, http://developer.android.com/guide/topics/sensors/sensors_position.html

[11] Environment Sensors, http://developer.android.com/guide/topics/sensors/sensors_environment.html

[12] Sensor Wiki, http://en.wikipedia.org/wiki/Sensor

[13] Earth’s Magnetic Field, http://en.wikipedia.org/wiki/Earth's_magnetic_field

[14] LUX Wiki, http://en.wikipedia.org/wiki/Lux

[15] Università Degli Studi di Parma, Android Development, Sensors and Media Management

[16] Android Composite Sensors, https://source.android.com/devices/sensors/composite_sensors.html

[17] Android Sensor Support from Matlab, http://www.mathworks.com/matlabcentral/fileexchange/40876-android-sensor-support-from-matlab/content/sensorgroup/Examples/html/CapturingAzimuthRollPitchExample.html

[18] Pengfei Zhou, Yuanqing Zheng, Zhenjiang Li, Mo Li, and Guobin Shen, Nanyang Technological University, Singapore IODetector: A Generic Service for Indoor Outdoor Detection

[19] N. D. Tripathi, J. H. Reed, and H. F. VanLandingham. Handoff in cellular systems. IEEE Personal Communications, 5:26–37, 1998

[20] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09), pages 85–98, 2009

[21] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory, 13(2):260–269, 1967

[22] Eclipse IDE, https://www.eclipse.org/

[23] Android SDK Tools, http://developer.android.com/tools/sdk/tools-notes.html

[24] Spherical Coordinate System, http://en.wikipedia.org/wiki/Spherical_coordinate_system

[25] Variance, http://en.wikipedia.org/wiki/Variance