SCHOOL OF SCIENCE AND ENGINEERING

CAR TRACKING ANTI-THEFT SYSTEM

CAPSTONE DESIGN

Submission Date

April 2015

Author

Saad Benrouyne

Supervisor from Al Akhawayn University

Dr. Riduan Mohamed Abid, Assistant Professor of Computer Science

Acknowledgements

I am using this opportunity to express my gratitude to everyone who supported me throughout my capstone Project. For this occasion, I would like to thank:

Dr. Riduan Mohamed Abid, who is my supervisor within the university. He helped me and coached me during my capstone project by giving me feedback and tips, and by guide me with valuable advice through my capstone. I want to thank him for his professional spirit and valuable guidance.

Mr. Rachid Lghoul, who have shown precious technical and theoretical support throughout the course of this capstone. I am thankful for his aspiring guidance, constructive criticism and friendly advice concerning my project. I am grateful for his cooperation which considerably helped me during my capstone project.

My final thoughts go to my family and friends, for their priceless moral and financial support throughout my years at Al Akhawayn University, and without whom this project would not have been possible.

[illegible]

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 ABSTRACT

On average, 2,000 cases of car theft are reported each year in Morocco [1], and the number is still increasing. If not recovered soon, stolen vehicles are generally sold, revamped or even burned if the resale price is considered to be too low. Once a vehicle is stolen, it becomes hard to locate it and track it, which considerably decreases the chances of recovering it. Nowadays, insurance companies suggest to their clients to equip their vehicles with a Global Positioning System (GPS) that can locate in real time their cars all over the country. In this work, we present the design and implementation of a Car Tracking Anti-theft System that protects and secures vehicles. This project consists of building two main components: 1. hardware component; this implements a geolocation system running on top of a Raspberry Pi dynamic microcontroller [2]; 2. software component; this deploys a web application and an Android application whereby end users and/or car owners monitor and locate their cars in an interactive map anytime and anywhere. Using this geolocation system will help in locating the car and tracing it fast in case it is stolen. This will increase the chances of recovering the car intact. In addition to tracking cars in real time and anywhere, the proposed solution aims at providing anti-theft features such as live tracking, tracing and proximity alerts as well as additional information about the vehicle.

2 INTRODUCTION

2.1 PROJECT CONTEXT

The main objective of this capstone project is to design and implement a car tracking anti-theft system that that will enable tracing, locating and monitoring cars in real time all over the country. Indeed, this solution aims at providing anti-theft features such as live tracking, tracing as well as proximity alerts.

2.2 STEEPLE ANALYSIS

STEEPLE Analysis is "a model for strategic decision that takes into consideration seven main macro-environmental factors in the activity of analysis, assessment and forecast of the impact of the decision to be made" [3]. STEEPLE stands for societal, technical, environmental, ethical, political, legal and economic factors. In this project, we adopted to STEEPLE model to analyze and assess the potential impact of our solution. The following table identifies the seven macro-environmental factors related to our project which include societal, technical, environmental, ethical, political, legal and economic factors.

- **Societal:** Poverty, population growth and urban migration are considered among the main drivers of crime. In this context, rising thefts means rising demand and rising need for anti-theft solutions.
- **Technological:** The emergence of microcontrollers and single board nano-computers in the past few years has enabled the design of efficient anti-theft systems.
- **Ecological:** Tracking, tracing and monitoring a vehicle help in analyzing drivers behavior. In fact, analyzing driving behavior makes it possible to lessen risks of accidents and decrease fuel consumption, thus, CO2 emissions.
- **Ethical:** Ethics issues of liberty and privacy is a main concern in geolocation systems.

- **Political:** In April 2014, King Mohammed VI ordered the Minister of the Interior to make more efforts in dealing with criminal phenomena that threatens the security and serenity of citizens. [4]
- **Legal:** Moroccan Labor legislation does not address in a direct and clear way the concern regarding the use of geolocation systems. [5]
- **Economic:** Nowadays, many companies tend to use fleet management solutions to lessen their transportation costs. Furthermore, insurance companies in Morocco start embracing and mandating anti-theft solutions as part of the conditions for insurance.

2.3 PROBLEM STATEMENT

The global issue related to a constantly increasing crime rate needs to be urgently addressed by both developed and developing countries. In Morocco, 2,000 cases of car theft in average are reported each year in Morocco [6], and the number is still increasing. If not recovered soon, stolen vehicles are generally sold, revamped or even burned if the resale price is considered to be too low. Once a vehicle is stolen, it becomes hard to locate it and track it, which considerably decreases the chances of recovering it. In this work, we propose the design and implementation of a car tracking anti-theft system that will protect, secure vehicles.

2.4 REPORT SUMMARY

In the first phase of this project, we will discuss the hardware implementation of the system, which consists of a dynamic micro-controller connected to a GPS module and a 3G module. After gathering the geolocation data from the GPS module, the nano-computer sends this data through the 3G network to a WEB server. The second phase of this project involves the software implementation of the system, which consist of a responsive WEB application running on top the WEB server. This application processes the sensed geolocation data in real time, and displays it in an interactive map. Furthermore, the solution is aimed to provide anti-theft features such as live tracking and tracing and alert radiuses. The third and last phase of the project is the development of an Android application that will have the same features as the WEB application which will provide more interactivity and ease of use to the potential user.

3 FEASIBILITY STUDY

In the context of the capstone project, a feasibility study can be defined as an analysis of the viability of the project. Its main goal is to identify problems and opportunities by measuring existing needs and conditions associated with the project. This includes assessing the cost related to the project and describing possible scenarios and situations for the development of the project.

The objective of this capstone is to develop a car tracking anti-theft system that will protect and secure vehicles. The first step I followed is studying and investigating similar existing solutions to come up with new ideas that could help in preventing a theft or recovering a stolen car in an efficient way. As reported by the Moroccan authorities, nearly 2000 cars were stolen in Casablanca in 2012 [7]. Moreover, according to the National Security Head Office, the office receives between two and ten declarations of car thefts per day in Casablanca alone.

These shocking data make people more aware of the depth and the seriousness of this situation. Indeed, being aware of the situation makes these people desperate since they know their car could be stolen anytime and they do not know how to protect it or what to do in case it is stolen. This is why it is important to protect our cars using a car tracking anti-theft system that would considerably help in recovering stolen cars within hours.

In fact, by using a GPS/3G system running on top of a nano-computer and monitored from a web interface and/or an Android interface, these people will be able to locate their vehicles fast, and successfully recover their stolen vehicles. In order to develop this software/hardware solution, I had to perform a technical feasibility study by asking myself what hardware and software I need to use in my project. In other words, I had to determine the available hardware I had and the available software I can use in my project considering both time and budget constraints. This section is further explained in the "Technology Enablers" part.

4 DEVELOPMENT MODEL

The development of software and/or hardware involves four main phases: requirements specifications, design, implementation, and testing. The requirements specification phase, also called requirement engineering phase, consists of collecting specifications of a system and remodeling them in a simple way in order to determine who is going to use the system, how it going to be used, what data will be input to the system and what data will be output by the system, and finally what requirements can be incorporated considering both time constraints and budget constraints. This data will help in identifying how a car can be protected against theft using a hardware and software solution and will involve getting inspired by the daily encountered problems in a Moroccan culture perspective, and other existing geolocation solutions. Indeed, gathering the specifications will help in defining functional requirements, non-functional requirements and system users. In the design phase, we describe the technology enablers to be used in the system and we investigate and develop our algorithms by taking into account the interaction between the different modules of the system which is described using a use case diagram. In the implementation phase, we implement our algorithms based on the output of the design phase. Hence the implementation phase will be the longest phase of the system development and will involve, first developing an algorithm for the embedded system to communicate wirelessly to a remote server in order to store information about the car, and second developing a web interface and an Android application interface that will allow car owners to monitor, locate and trace in a map their cars anytime and anywhere. Finally after completing the coding and implementation, the testing phase will consist of unit testing, system testing, integration testing and acceptance testing to make sure that the system is correctly solving the addressed problem of car theft.

For any given software and/or hardware project, it is crucial to know how to interleave these four stages, and which methodologies and processes to select depending on the project goals.

In fact, several development life cycle models have been developed to accomplish different project goals and objectives, and each one of these models specifies the different stages of the process as well as the order in which they should proceed. Each development life cycle model falls within one of the three basic types of software development models : the sequential models, the evolutionary models and the incremental models.

Sequential models, such as the pure Waterfall model, suggest a systematic sequential approach to software development. Indeed, we need to completely understand and describe the problem before designing a solution. After specifying the design, all implementation is done before the testing phase. In other words, each stage must be completed before moving to the next stage. This sequential approach requires having a clear and fixed set of requirements, a heavy documentation and a stable product definition since it is impossible to go back to an earlier stage and make changes once a phase has been completed.

Incremental development lifecycle models suggest a different and more flexible approach to software development. Indeed, unlike sequential models, incremental models propose starting with a basic project design, and then working on small modules or increments, allowing for changes to be made during the whole development lifecycle.

In evolutionary models, such as the spiral model, it is unfeasible to know all the requirements in advance. These models are often used in research and development where the requirements are partially defined and refined in a series of cycles.

My first attempt towards developing this project was to apply the waterfall model. The reason behind choosing this model was mainly because I had only few knowledge about embedded systems; therefore, I needed a heavy documentation to be prepared and a clear set of requirements to be defined before starting the design and implementation of my project. However, over the course of the first weeks, I noticed that this model was very timeconsuming in terms of preparing and reading the documentation and defining stable and fixed requirements. I realized then that the requirement specifications stage could take a considerable amount of time before completing it, and only then I would be able to work on the design phase before starting the implementation of my system.

Because of the time constraints, the new technology that need to be learnt and used, and the instability of the requirements, I decided to move to the incremental development approach allowing the hardware/software system to be divided into small modules or increments. Indeed, I opted for the Agile development model which also follows the incremental approach.

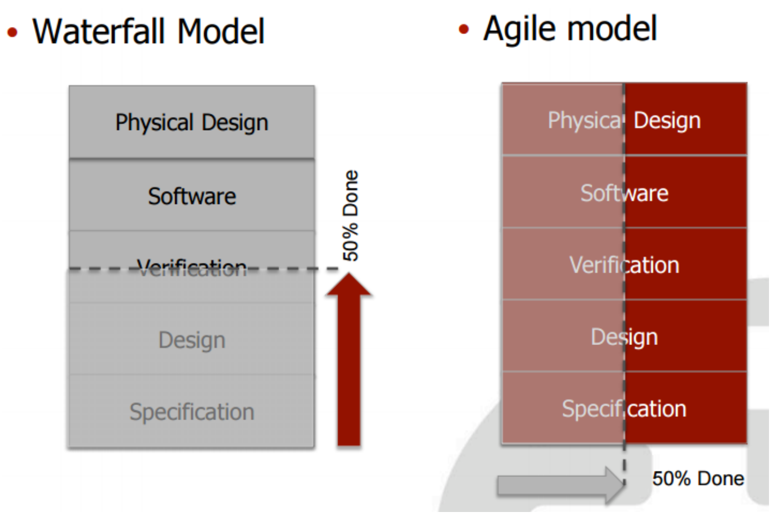


Figure 1: Waterfall Model Vs Agile Model

In this model, coding and implementation are the main concerns and documentation is kept to the minimum. I chose this model since it is useful in developing working functionality rapidly and early in the life cycle, and obtaining periodical results. Moreover, it is more flexible and less costly in terms of changing the scope of the project and/or the requirements.

5 HARDWARE DEVELOPMENT PROCESS

5.1 HARDWARE SPECIFICATIONS

5.1.1 FUNCTIONAL REQUIREMENTS

In the context of hardware development process, functional requirements can be considered as the functionalities or services the system must provide. In other words, functional requirements describe how a system is to react and behave given specific inputs. In this project we defined six main functional requirements, each of which is described in details in the table below.

Table 1: Hardware Functional Requirements

Functional Requirement Name	Description
setup_gps_sensor( )	Since we will be using an external GPS module, the system needs to have a functionality to setup the GPS receiver by installing the needed drivers and set the needed configuration for the receiver to be detected and read by the Raspberry Pi nano-computer.
setup_3g_module( )	Since we will be using an external 3G/GSM modem, the system needs to have a functionality to setup the 3G/GSM modem by installing the needed drivers and setting the needed configuration for the modem to be detected by the Raspberry Pi nano-computer.
switch_usb_mode( )	The Raspberry Pi nano-computer runs on top of a Linux based operating system. In Linux, USB modems are usually detected

	as USB storage devices. Hence, the system needs to be able to switch the USB 3G modem mode from "USB storage mode" to "Modem mode"
authenticate_3g( )	After setting up the 3G modem configuration and the system needs to authenticate the SIM card used by the 3G modem with a valid PIN. Once PIN is checked by the network provider, for instance "INWI", an IP address is attributed to Raspberry Pi and a connection to internet is opened.
read_raw_gps_data( )	The GPS receiver connected to a GPS antenna sends raw geolocation data to the dynamic micro-controller. Therefore the system has to be able to receive and read this geolocation data using geolocation libraries.
format_raw_gps_data( )	Since the read GPS data is raw, it is still not readable. This is why the system needs to reformat this geolocation data to become readable.
parse_readable_data()	Once the GPS data is readable, the system parses the data into different variables including, longitude, latitude, altitude, speed as well as other geolocation variables.
send_gps_data( )	Once GPS data is parsed and stored into separate variables the last step is to send this data to be processed by a WEB server using POST HTTP requests.

5.1.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements can be considered as constraints on the functionalities and services offered by the system. Constraints can be either time constraints, constraints on the development process or standards to be followed. These requirements are generally dictated by the client and vary from input and output device capability and storage requirements to system response time, used programming language and development tools. Since we do not have yet a client for our project, we defined a set of non-functional requirements to be followed in our project. These non-functional requirements are delineated and described in details in the table below.

Table 2: Hardware Non-Functional Requirements

The GPS tracker should run on top of a Raspberry-Pi nano-computer
The programming language used for reading and sending GPS data is Python
The system should send geolocation data related to its current position every 1 second
Multi-threading will be used while reading GPS data in order to prevent a buffer overflow
Shell Programming is to be used for the setup and configuration of the 3G and GPS modules
In the design process we will follow an incremental approach based on Agile methodology

5.2 HARDWARE DESIGN

5.2.1 SYSTEM ARCHITECTURE

In the design phase, we describe the technology enablers to be used in the system and we investigate and develop our algorithms by taking into account the interaction between the different modules of the system. This section presents the architectural design of our "Anti-Theft Car Tracking System". The following architecture provides an overall view of the main system components which involve a RaspberryPi-based GPS tracker that will collect and send the GPS data, a load balancer in order to distribute the workload across multiple WEB servers, HTTP servers that will store geolocation data into a database and fetch it from database when it is needed to be displayed, a MySQL database that will store geolocation data about each car, and the end users that can see in real time the location of their vehicle in a map either from a web interface or an Android-based smartphone.

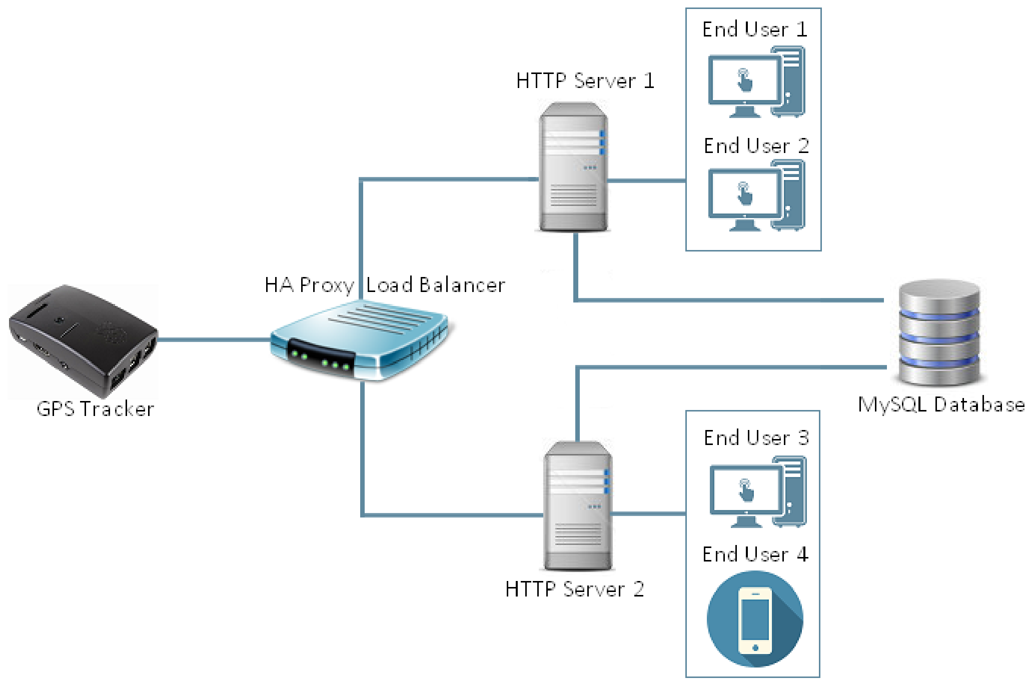


Figure 2: System Architecture

Table 3: System Architecture Components Description

Components	**Description**
GPS Tracker	The GPS tracker is composed by a Raspberry-Pi nano-computer connected to a 3G modem module and a GPS receiver. A detailed design of the GPS tracker with all its components is shown in the figure bellow.
HA Proxy Load Balancer	"HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for very high traffic web sites and powers quite a number of the world's most visited ones." [8] This load balancer in order to distribute the workload across multiple web servers. In our case you chose to balance

	the load between two web servers.
HTTP Server	An HTTP server, commonly named web server, is defined as an information technology that processes requests via the HTTP network protocol used to spread and share information on the World Wide Web. This server will be responsible for reading the HTTP requests sent by the GPS tracker process them and storing them in a MySQL database.
End User	In the context of this project, the end user can either access the application through a WEB interface or through an Android smartphone.
MySQL Database	MySQL is the most widely used open-source relational database management system, and used Structured Query Language SQL. In this project we will be working with MySQL.

5.2.2 TECHNOLOGY ENABLERS

The technology enablers are defined as a set of features and technology that will be used in the implementation of the application. The table bellow describes the set of hardware technology enablers we will be using in our project and lists them by "Manufacturer", "Reference" and "Description".

Table 4: Hardware Technology Enablers

Manufacturer	Reference	Brief Description
		The Raspberry Pi is a low cost, single card nano-

RASPBERRY PI FOUNDATION	Raspberry-Pi B+	computer that can be plugged into a TV or computer monitor, and hat can be accessed using a standard keyboard and/or a mouse.
ARDUINO	Arduino Uno R3	The Arduino Uno is a microcontroller board based on the single-ship ATmega328 micro-controller. This microcontroller has fourteen digital input and output pins, a USB port and a reset button.
TELEPHONICA	GSM/GPRS Shield	The GSM/GPRS shield is used to connect an Arduino to the internet using GPRS wireless network.
ITEAD	ITEAD GPS Shield	ITEAD GPS Shield is a GPS module breakout board based on the RoyalTek REB-4216 GPS module. It was designed as a Global Positioning System receiver that can store GPS data into an SD card.
VILROS	Wifi USB Dongle	This Wifi Adapter is based on the Realtek 8188CU Chip assures the Wifi connectivity to external networks.
ADAFRUIT	USB to Serial Cable	This cable is used to connect to Raspberry Pi serial console port. Inside the USB plug is a USB Serial conversion chip which distributes the received signal among four colored cables.
VILROS	Male/Female Cable	These cables are used for making wiring and jumpering between the different headers on PCB's.

HUAWEI	Huawei EC156	Huawei EC156 USB Modem is a wireless network terminal based on the CDMA2000 1x EV-DO protocol that offers wireless downloading at high speeds.

5.2.2.1 RASPBERRY PI B+

The Raspberry Pi is a low cost, single card nano-computer that can be plugged into a TV or computer monitor, and hat can be accessed using a standard keyboard and/or a mouse. This device enables its users to learn how to program in languages like Python work explore computing in an enjoyable way. This nano-computer also called dynamic micro-controller uses GPIO pins to communicate with external I/O devices such as sensors or touch screens which makes the bridge between software and hardware.

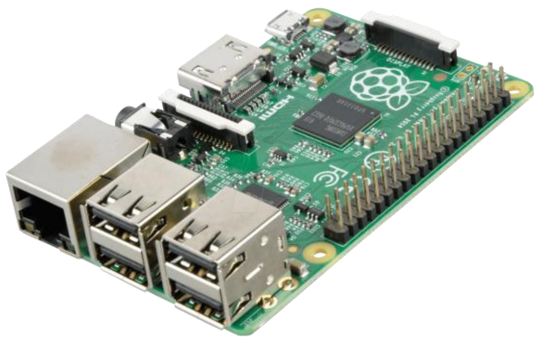


Figure 3: Raspberry Pi B+ nano-computer

Raspberry Pi Model B+ has 40 GPIO pins which offers an increase of 14 pins over over the original Raspberry Pi. These pins include 3V Power and 5V power pins, Ground Pins, TXD transmitter pins, RXD receiver pin as well as other pins we will not be working with in this project. The figure bellow describes the set of pins contained in the Raspberry Pi nano-computer.

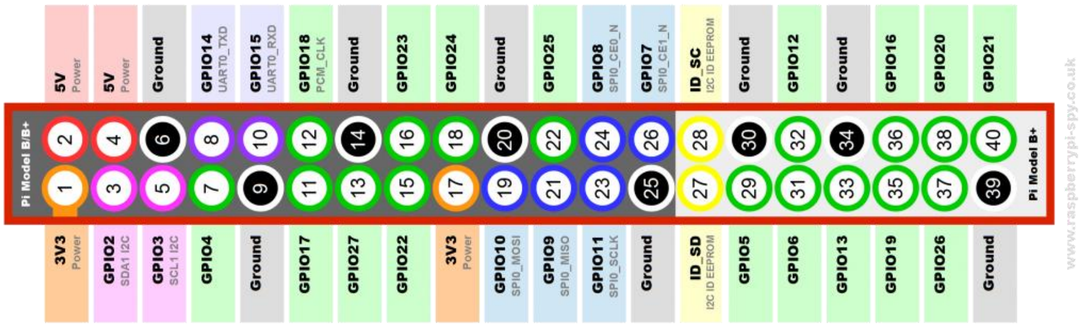


Figure 4: Raspberry Pi B+ GPIO Header Details And Pinout

Raspberry Pi B+ runs on top of a Linux-based operating system. In our project, we used the Raspbian operating system, an operating system based on Debian. The figure bellow represents a screenshot of the graphical user interface of the Raspian operating system installed in the Raspberry Pi nano-computer.

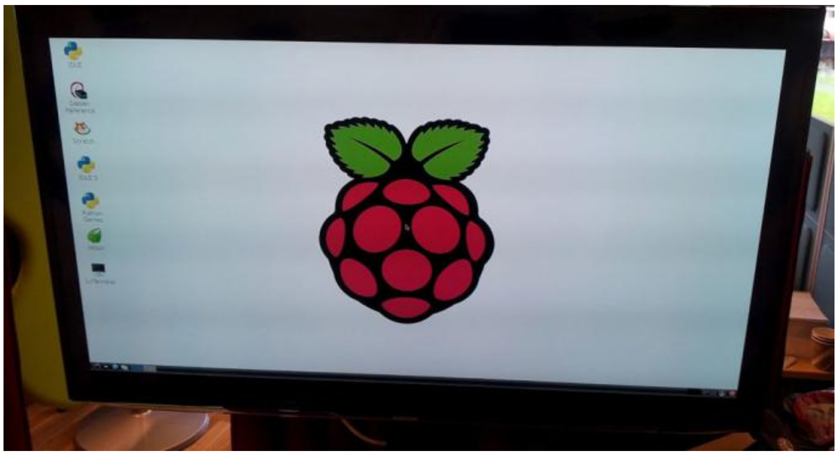


Figure 5: Raspian Graphical User Interface

5.2.2.2 ARDUINO UNO R3

The Arduino Uno is a microcontroller board based on the single-ship ATmega328 micro-controller. This microcontroller has fourteen digital input and output pins, a USB port and a reset button. It is connected to a computer using a USB cable and it is power either using an AC-to-DC adapter or a battery.

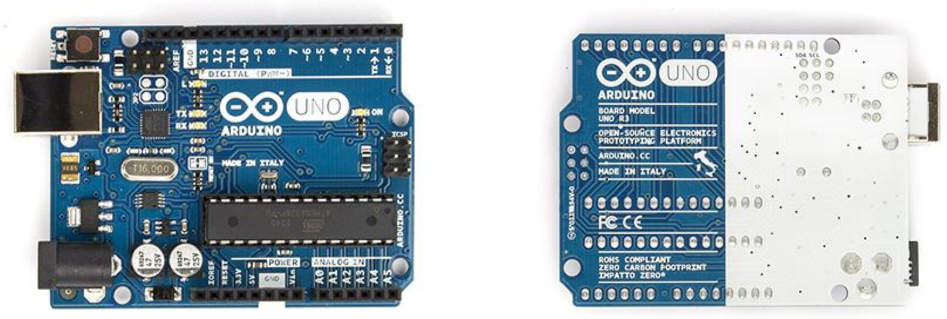


Figure 6: Arduino Uno R3

5.2.2.3 TELEPHONICA GSM/GPRS SHIELD

The GSM/GPRS shield is used to connect an Arduino to the internet using GPRS wireless network. After plugging the module onto the Arduino board and plugging in the module a SIM card from an operator that offers GPRS coverage, it becomes possible to make and/or receive calls, send and/or receive SMS messages and connect to the internet.

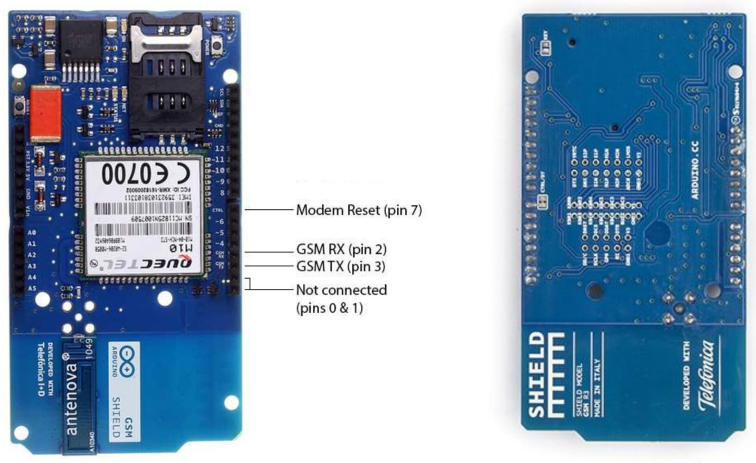


Figure 7: TELEPHONICA GSM/GPRS SHIELD

5.2.2.4 ITEAD GPS SCHIELD

ITEAD GPS Shield is a GPS module breakout board based on the RoyalTek REB-4216 GPS module. It was designed as a Global Positioning System receiver that can store GPS data into an SD card. Its main pins are RX which stands for receiver pin, TX which stands for transmitter pin, 5V pin and a Ground pin. In this project we will be using these pins to connect this ITEAD GPS Shield to Raspberry Pi.

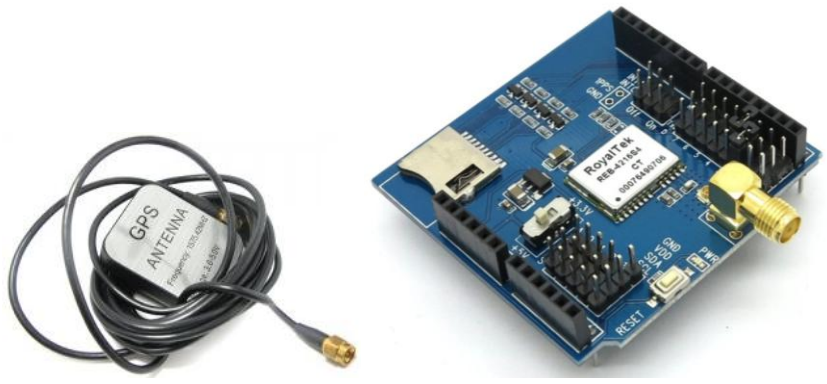


Figure 8: Itead GPS Shield and Antenna

5.2.2.5 USB WIFI DONGLE

This Wifi Adapter is based on the Realtek 8188CU Chip assures the Wifi connectivity to external networks.

Figure 9: USB Wifi Dongle

5.2.2.6 *USB TO TTL CONVERTER*

This cable is used to connect to Raspberry Pi serial console port. Inside the USB plug is a USB Serial conversion chip which distributes the received signal among four colored cables. The red cable is the power cable, the black represents the ground, the white cable is the RX receiver cable and the green cable represents the TX transmitter cable.

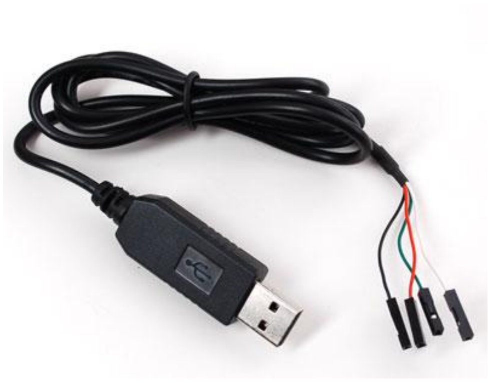


Figure 10: USB to TTL converter

5.2.2.7 *MALE/FEMALE CABLES*

These cables are used for making wiring and jumpering between the different headers on PCB's. In this project they will be used for a serial connection between the GPS shield and Raspberry.

Figure 11: Male/Female Cables

5.2.2.8 *MODEM HUAWEI EC156*

Huawei EC156 USB Modem is a wireless network terminal based on the CDMA2000 1x EV-DO protocol that offers wireless downloading at high speeds. In this project this USB modem will be used to get access to the internet in order to be able to send POST HTTP requests to the web server.

Figure 12: Modem HUAWEI EC156

5.2.3 GPS TRACKER DESIGN

The following figures illustrates the different components constituting the GPS Tracker. In the first figure the GPS Tracker is composed of a Telephonica GSM/GPRS shield that will be used to send HTTP requests to the WEB server, an Itead GPS shield used to receive current GPS data from GPS satellites, a 4 x 1.5 volt battery holder to power the whole GPS tracker and the Raspberry Pi nano-computer in which all data will be processed.

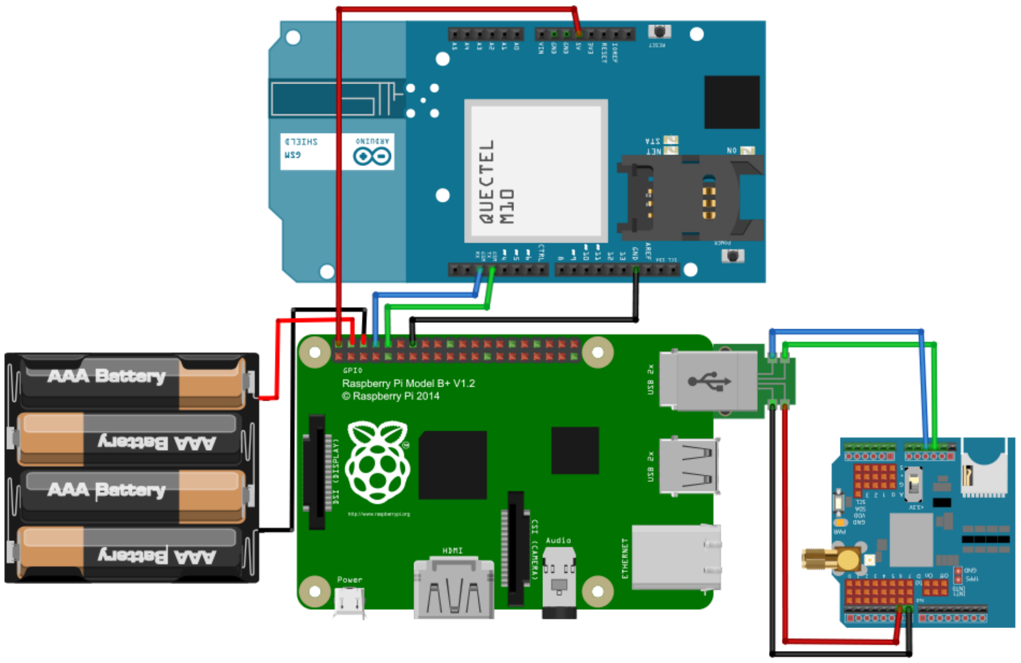


Figure 13: GPS Tracker Design using a GSM shield

After doing testing the GSM/GPRS Module with different SIM Cards from different providers, I was unfortunately unable to use the GPRS network to send HTTP Requests to my remote server since GPRS is only limited to few websites as well as WAP (Wireless Application Protocol) services, and I needed a special GPRS SIM card that is not sold to individuals in Morocco but only to companies. Therefore, I decided to investigate in the 3G technology and especially on its compatibility with the Raspberry Pi nano-computer running on top of Rasbian, a Debian Linux-based operating system. After doing research on the 3G, and investigating the compatibility of some USB 3G Modems with Raspberry Pi, I decided to buy this week a HUAWEI EC156 CDMA-based USB Modem, and I ended up with the following design.

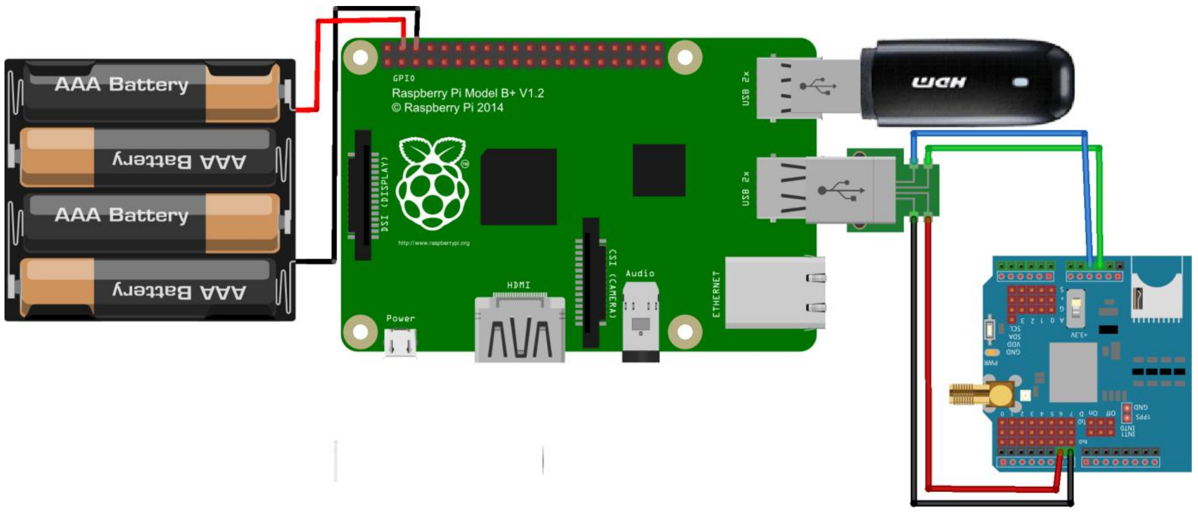


Figure 14: GPS Tracker Design using a 3G USB modem

5.3 HARDWARE IMPLEMENTATION

5.3.1 GPS TRACKER IMPLEMENTATION

After the design phase, we implemented two different GPS Trackers one of which is composed of a Telephonica GSM/GPRS shield that will be used to send HTTP requests to the WEB server, an Itead GPS shield used to receive current GPS data from GPS satellites, a 4 x

1.5 volt battery holder to power the whole tracker and the Raspberry Pi nano-computer in which all data will be processed.

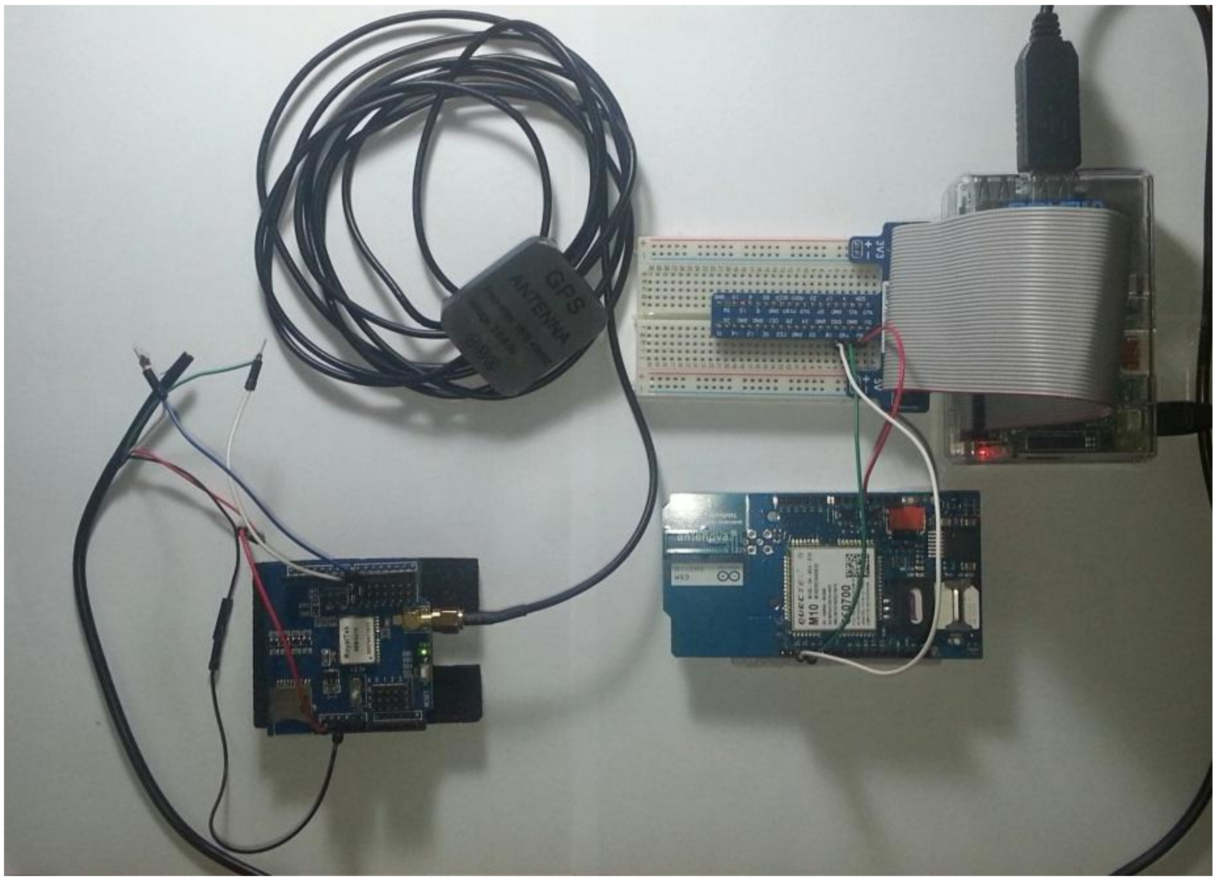


Figure 15: GPS Tracker Implementation using a GSM shield

After implementing the circuit above and testing the connectivity of the GSM/GPRS Module with different SIM Cards from different providers, I was unfortunately unable to use the GPRS network to send HTTP Requests to my remote server since GPRS is only limited to few websites as well as WAP (Wireless Application Protocol) services, and I needed a special data SIM card that is not sold to individuals in Morocco but only to companies. Therefore, I decided to investigate in the 3G technology and especially on its compatibility with the Raspberry Pi nano-computer running on top of Rasbian, a Debian Linux-based operating system. After investigating on the compatibility of USB 3G Modems with Raspberry Pi, I decided to implement the same circuit but with a USB 3G modem instead of a GSM/GPRS module. The following pictures shows the final implemented circuit which represents our GPS tracker.

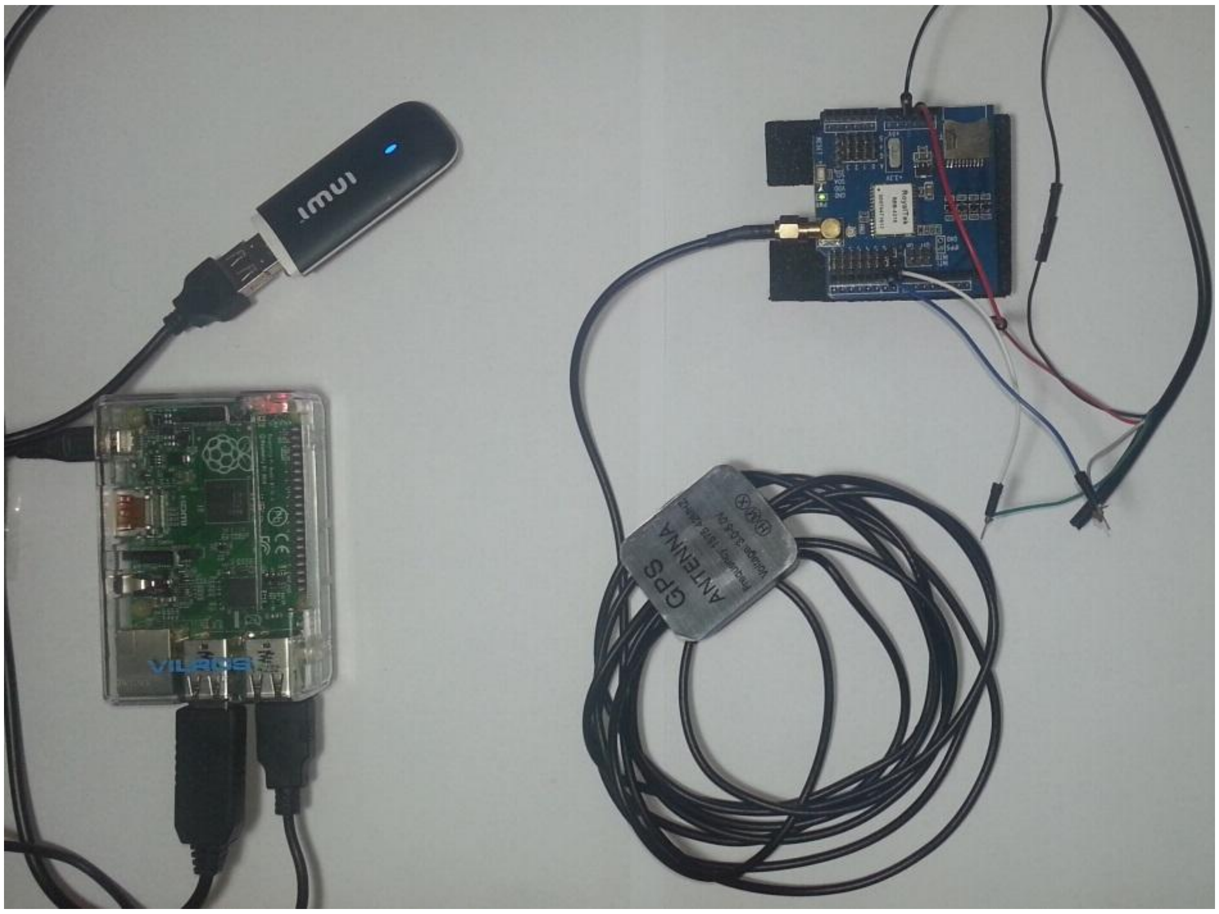


Figure 16: GPS Tracker Design using a 3G USB Modem

5.3.2 HARDWARE INSTALLATION

5.3.2.1 OPERATING SYSTEM INSTALLATION

The first step to configure our hardware was installing the Raspbian operating system in the Raspberry Pi nano-computer. To do so, we downloaded the Raspbian Linux distribution from the Raspberry Pi official website: https://www.raspberrypi.org/downloads.

Afterwards, we used a tool called Win32 Disk Imager to burn the Rasbian Linux distribution into the Raspberry Pi Micro-SD card by selecting the disk image file containing the Rasbian Linux distribution and by selecting the device into which the image will be burned.

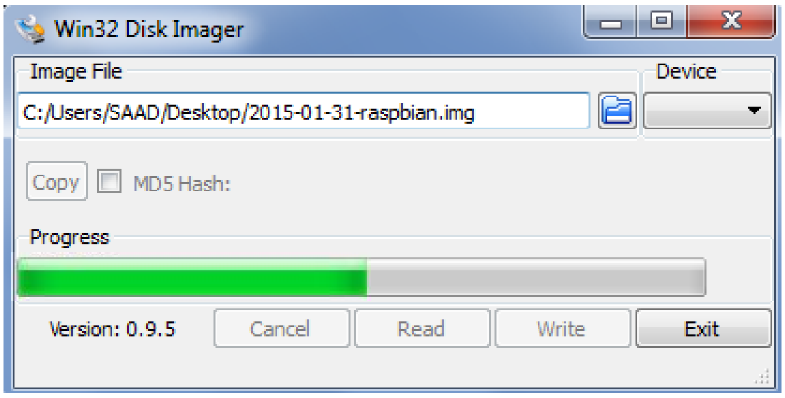
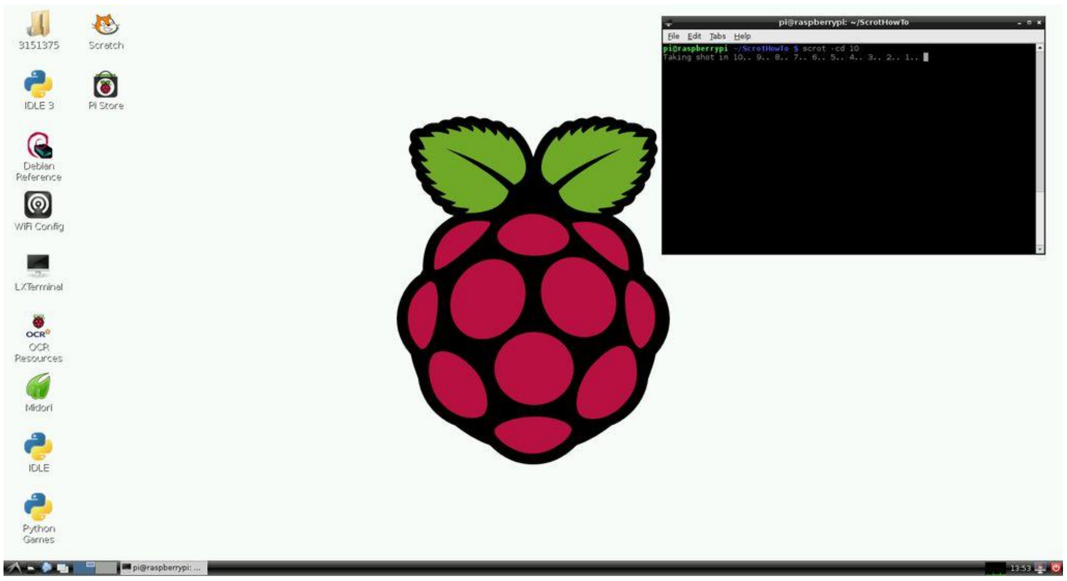


Image File: C:/Users/SAAD/Desktop/2015-01-31-raspbian.img

Version: 0.9.5

Figure 17: Win32 Disk Image Burning a Disk Image

After burning Raspbian into the micro-SD card, we inserted the micro-SD card into Raspberry Pi. We connected Raspberry Pi to a TV using an HDMI cable and we powered it up. The following picture represents the graphical user interface of the Rapbian operating system running on top of Raspberry Pi B+.



```
pi@raspberrypi ~/ScrotHowTo $ scrot -cd 10
Taking shot in 10.. 9.. 8.. 7.. 6.. 5.. 4.. 3.. 2.. 1..
```

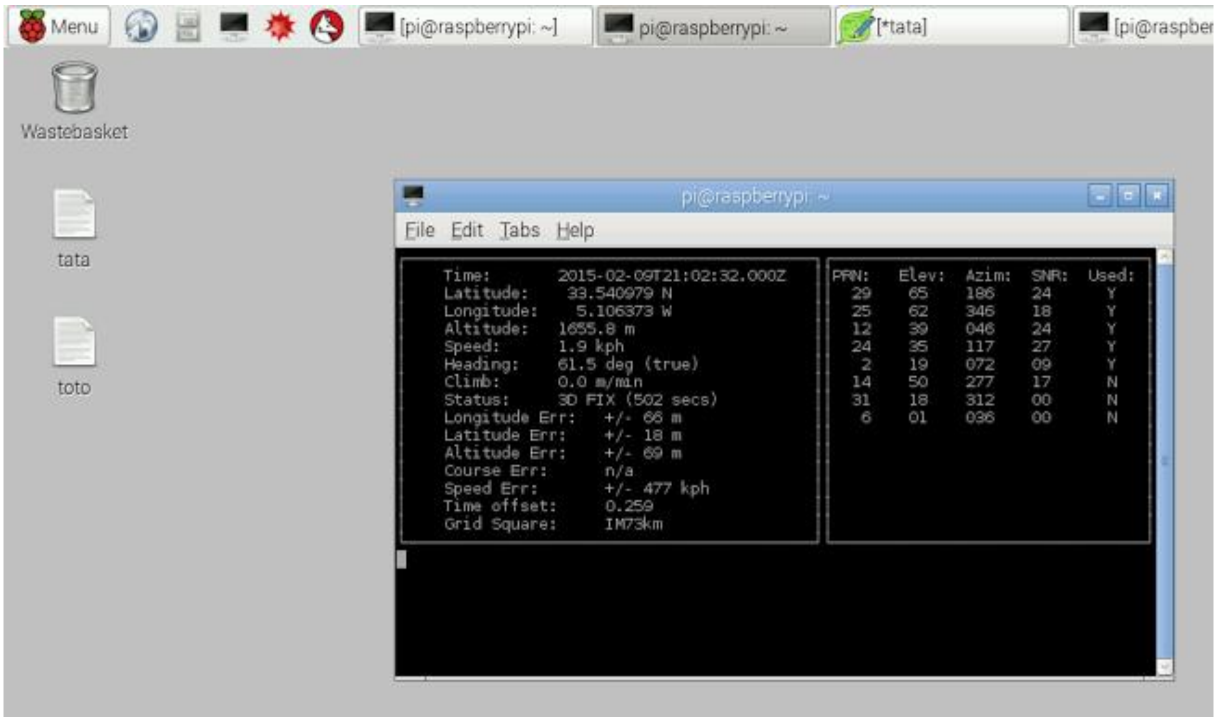
Figure 18: Raspberry Pi Graphical User Intertface

5.3.2.2 GPS SHIELD CONFIGURATION

To set up our GPS shield, we first installed a utility called "gpsd" to be able to listen to our GPS shield and the "gpsd-client" and "python-gps packages" which provide tools such as cgps, gpscat or xgps used to test the GPS communication with Raspberry Pi through the serial port. The following code was used to set up our GPS shield.

```
1. sudo apt-get install gpsd gpsd-clients python-gps
2. sudo gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock
3. cgps -s
```

After installing the necessary packages (Line 1), we pointed out the GPS daemon to listen to our GPS shield on the USB to TTL converter cable, by setting the destination of the port on which the USB to TTY adapter is connected (Line 2). In our case the destination is "/dev/ttyUSB0". After few seconds a local socket is opened and GPS data start flowing in. The following screenshot illustrates the set of GPS data read by the GPS receiver and displayed on the shell console.



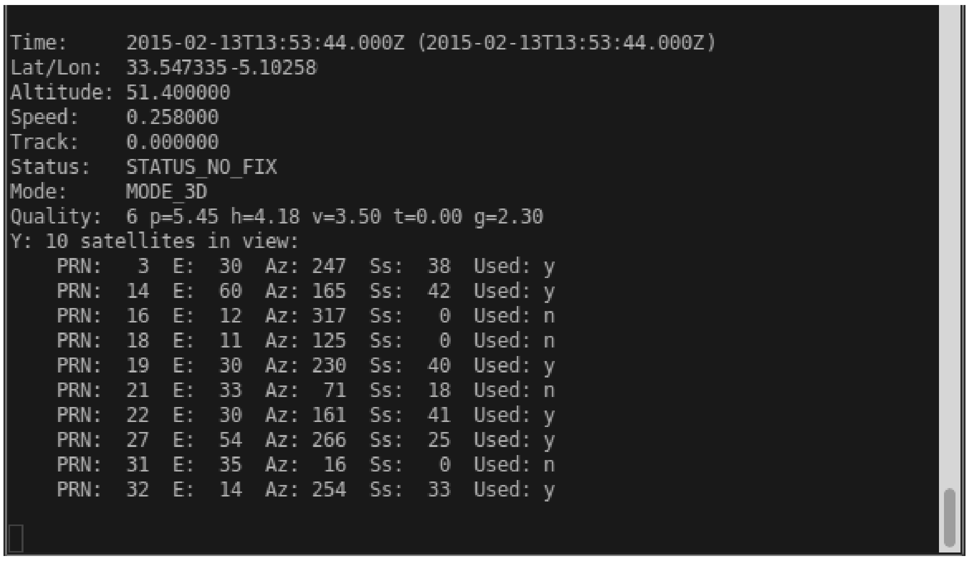
```
    Time:          2015-02-09T21:02:32.000Z      PRN:   Elev:  Azim:  SNR:  Used:
    Latitude:       33.540979 N                   29     65     186    24     Y
    Longitude:       5.106373 W                   25     62     346    18     Y
    Altitude:      1655.8 m                       12     39     046    24     Y
    Speed:         1.9 kph                        24     35     117    27     Y
    Heading:       61.5 deg (true)                 2     19     072    09     Y
    Climb:         0.0 m/min                      14     50     277    17     N
    Status:        3D FIX (502 secs)              31     18     312    00     N
    Longitude Err:    +/- 66 m                     6     01     036    00     N
    Latitude Err:     +/- 18 m
    Altitude Err:     +/- 69 m
    Course Err:       n/a
    Speed Err:        +/- 477 kph
    Time offset:      0.259
    Grid Square:      IM73km
```

Figure 19: Data Read by the GPS receiver

In order to store the GPS data to be used afterwards, we reused a Python code that was found on the internet and that was written by a famous developer named Dan Mandle [9]. This code uses multi-threading to prevent the buffer overflows. We tested the code bellow using the Python integrated development environment IDLE.

```
import os
from gps import *
from time import *
import time
import threading
import urllib2, urllib

gpsd = None # seting the global variable

os.system('clear') # clear the terminal (optional)

class GpsPoller(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        global gpsd # bring it in scope
        gpsd = gps(mode=WATCH_ENABLE) # starting the stream of info
        self.current_value = None
        self.running = True # setting the thread running to true

    def run(self):
        global gpsd
        while gpsp.running:
            gpsd.next() # this will continue to loop and grab EACH set of gpsd info
                          to clear the buffer
if __name__ == '__main__':
    gpsp = GpsPoller() # create the thread
    try:
        gpsp.start() # start it up
        while True:
            #It may take a second or two to get good data
            #print gpsd.fix.latitude,', ',gpsd.fix.longitude,'  Time: ',gpsd.utc

            os.system('clear')

            print
            print ' GPS reading'
            print '----------------------------------------'
            print 'latitude    ' , gpsd.fix.latitude
            print 'longitude   ' , gpsd.fix.longitude
            print 'time utc    ' , gpsd.utc,' + ', gpsd.fix.time
            print 'altitude (m)' , gpsd.fix.altitude
            print 'eps         ' , gpsd.fix.eps
            print 'epx         ' , gpsd.fix.epx
            print 'epv         ' , gpsd.fix.epv
            print 'ept         ' , gpsd.fix.ept
            print 'speed (m/s) ' , gpsd.fix.speed
            print 'climb       ' , gpsd.fix.climb
            print 'track         ' , gpsd.fix.track
            print 'mode          ' , gpsd.fix.mode
            print

            time.sleep(1) # the time to wait before closing the loop
again

    except (KeyboardInterrupt, SystemExit): # when you press ctrl+c
        print "\nKilling Thread..."
        gpsp.running = False
        gpsp.join() # wait for the thread to finish what it is doing
    print "Done.\nExiting."
```

Running the program above generates the following output which contains GPS data including the current time, latitude, longitude, speed, track, status as well as other GPS data.



```
Time:     2015-02-13T13:53:44.000Z (2015-02-13T13:53:44.000Z)
Lat/Lon:  33.547335 -5.10258
Altitude: 51.400000
Speed:    0.258000
Track:    0.000000
Status:   STATUS_NO_FIX
Mode:     MODE_3D
Quality:  6 p=5.45 h=4.18 v=3.50 t=0.00 g=2.30
Y: 10 satellites in view:
    PRN:   3  E:  30  Az: 247  Ss:  38  Used: y
    PRN:  14  E:  60  Az: 165  Ss:  42  Used: y
    PRN:  16  E:  12  Az: 317  Ss:   0  Used: n
    PRN:  18  E:  11  Az: 125  Ss:   0  Used: n
    PRN:  19  E:  30  Az: 230  Ss:  40  Used: y
    PRN:  21  E:  33  Az:  71  Ss:  18  Used: n
    PRN:  22  E:  30  Az: 161  Ss:  41  Used: y
    PRN:  27  E:  54  Az: 266  Ss:  25  Used: y
    PRN:  31  E:  35  Az:  16  Ss:   0  Used: n
    PRN:  32  E:  14  Az: 254  Ss:  33  Used: y
```

Figure 20: Output of the Python GPS program

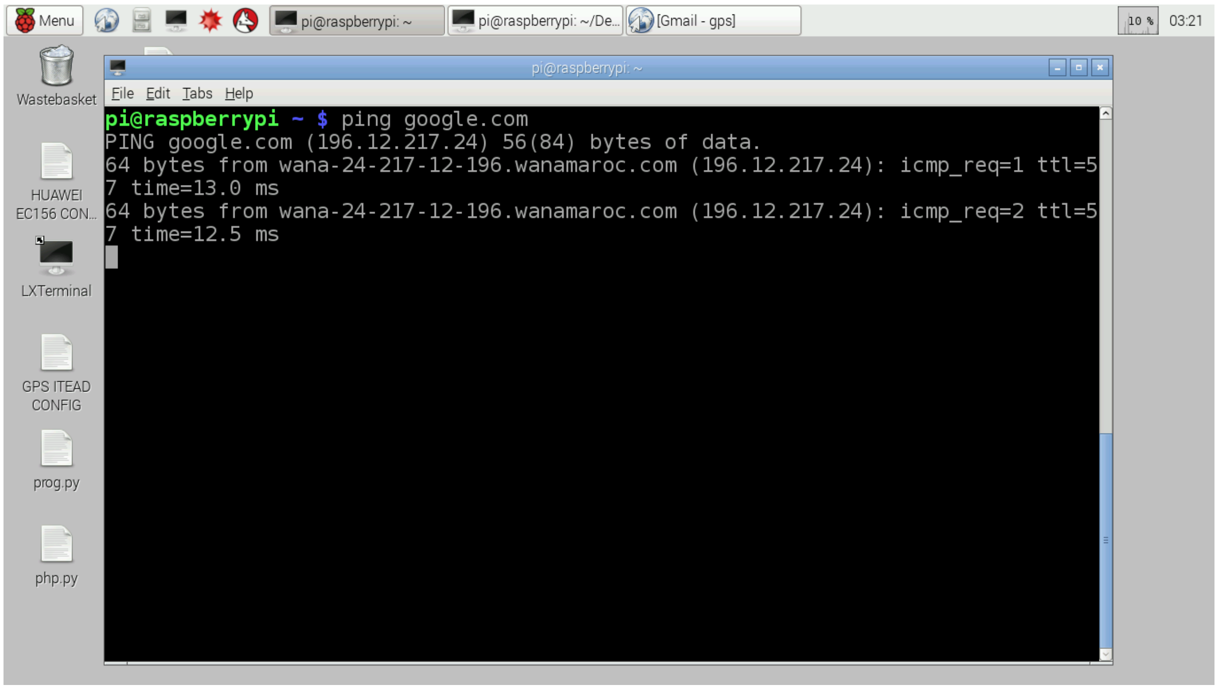
5.3.2.3 3G MODEM CONFIGURATION

In order to be able to send the sensed GPS data to a distant WEB server, we first need to be able to access the internet. In Linux, USB modems are usually detected as USB storage devices. Hence, the system needs to be able to switch the USB 3G modem mode from "USB storage mode" to "Modem mode". This switching is performed using the "usb-modeswitch" library. We also use "wvdial", a utility to make modem-based connections to Internet and "ppp", a point-to-point protocol that manages network connections in Linux. Before switching the USB 3G modem mode from "USB storage mode" to "Modem mode", we need to change the vendor code and the product code to the ones corresponding to our HUAWEI EC156 modem. Afterwards, we create a file containing the authentication information related to the network provider. In our case, we used INWI as a network provider which username is "wana", password is "wana" and dial number is "#777". After switching the USB mode to modem mode and setting this configuration, we open an internet connection using the command "sudo wvdial".

```
sudo apt-get update
sudo apt-get install ppp usb-modeswitch wvdial

cd /etc/usb_modeswitch.d
sudo vi 12d1:1505

DefaultVendor= 0x12d1
DefaultProduct=0x1505

MessageContent="55534243123456780000000000000011062000000100000000000000000000"

sudo leafpad /etc/wvdial.conf

[Dialer Default]
Init1 = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
Stupid Mode = 1
Modem Type = Analog Modem
ISDN = 0
Phone = #777
Modem = /dev/ttyUSB2
Username = wana
Password = wana
Baud = 460800

```

```
cd /etc/usb_modeswitch.d
sudo usb_modeswitch -I -W -c 12d1\:1505

sudo wvdial

sudo route add default gw 192.168.181.12
```

In order to test our code we can ping a remote server IP address for instance "google.com" IP address, and we can see that packet are being received successfully.



```
pi@raspberrypi ~ $ ping google.com
PING google.com (196.12.217.24) 56(84) bytes of data.
64 bytes from wana-24-217-12-196.wanamaroc.com (196.12.217.24): icmp_req=1 ttl=5
7 time=13.0 ms
64 bytes from wana-24-217-12-196.wanamaroc.com (196.12.217.24): icmp_req=2 ttl=5
7 time=12.5 ms
```

Figure 21: Pinging an IP address to test the connection

5.3.3 SENDING GPS DATA TO WEB SERVER

Now that we can correctly read GPS data and connect to Internet, we reused the Python code that reads GPS data and we added a new functionality to the program to be able send POST HTTP requests. The code bellow represents the latest version of the Python program that reads GPS data from the GPS receiver and sends the data to a WEB server via POST HTTP requests.

```python
import os
from gps import *
from time import *
import time
import threading
import urllib2, urllib

gpsd = None #seting the global variable

os.system('clear') #clear the terminal (optional)

class GpsPoller(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        global gpsd #bring it in scope
        gpsd = gps(mode=WATCH_ENABLE) #starting the stream of info
        self.current_value = None
        self.running = True #setting the thread running to true

    def run(self):
        global gpsd
        while gpsp.running:
            gpsd.next() #this will continue to loop and grab EACH set of gpsd info to clear the buffer

if __name__ == '__main__':
    gpsp = GpsPoller() # create the thread
    try:
        gpsp.start() # start it up
        while True:

            os.system('clear')

            mydata=[('longitude',gpsd.fix.longitude),('latitude',gpsd.fix.latitude)]  #The first is the var name the second is the value
            mydata=urllib.urlencode(mydata)
            path='http://afakwebmaster.dx.am/add.php' #The url we want to POST to
            req=urllib2.Request(path, mydata)
            req.add_header("Content-type", "application/x-www-form-urlencoded")
            page=urllib2.urlopen(req).read()


            time.sleep(1) #set to whatever

    except (KeyboardInterrupt, SystemExit): #when you press ctrl+c
        print "\nKilling Thread..."
        gpsp.running = False
        gpsp.join() # wait for the thread to finish what it's doing
    print "Done.\nExiting."
```

6 SOFTWARE DEVELOPMENT PROCESS

6.1 SOFTWARE SPECIFFICATION

6.1.1 FUNCTIONAL REQUIREMENTS

In the context of software development process, functional requirements can be considered as the functionalities or services the system must provide. In other words, functional requirements describe how a system is to react and behave given specific inputs. In this project we defined six main building blocks: Manage Users, Manage Cars, Track Cars and Protect Cars each of which listing a set functional requirements.

Manage Users

- create_user();
- authenticate();
- change_password();
- recover_password();
- change_user_info();

Manage Cars

- create_car();
- change_car_info();
- assign_car_to_user();

Track Car

- display_current_location_in_map();
- display_current_speed ();
- display_current_altitude();
- display_current_longitude();
- display_current_latitude();

Protect Car

- set_alert_radius();
- enable_proximity_alerts();
- disable_proximity_alerts() ();

6.1.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements can be considered as constraints on the functionalities and services offered by the system. Constraints can be either time constraints, constraints on the development process or standards to be followed. These requirements are generally dictated by the client and vary from input and output device capability and storage requirements to system response time, used programming language and development tools. Since we do not have yet a client for our project, we defined a set of non-functional requirements to be followed in the design of our system. These non-functional requirements are delineated and described in details in the table below.

Table 5: Non-Functional Software Requirements

Only 4 months to have a software with working functionalities.
Android Studio should be used for Android application development.
Google Maps Android API v2 is to be used as the application programming interface for embedding maps into our Android Application.
Google Maps JavaScript API v3 is to be used as the application programming interface for embedding Google maps into web pages using JavaScript.
The marker representing the current location of the car should be updated every second without refreshing the whole map.
JSON should be used as a format to transmitting data between the MySQL database server

and the web and/or Android application.
Parse Android SDK should support user login and registration in the Android application.

6.2 SOFTWARE DESIGN

6.2.1 USE CASE DIAGRAM

UML or Unified Modeling Language is "a standard notation for the modeling of real-world objects and systems" [10]. This language has various types of diagrams which are divided into two categories: structural diagrams and behavior diagrams. Structural diagrams emphasizes element that must be present in the system that is being modeled whereas behavioral diagrams illustrates what must happen in the system that is being modeled.

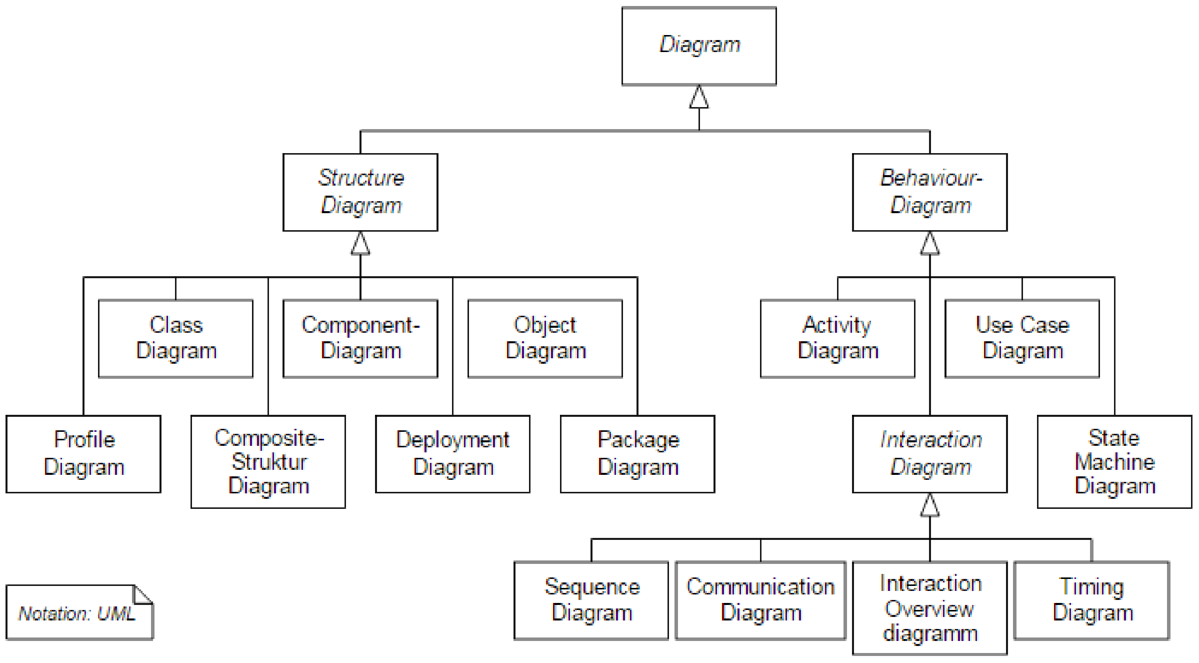


Figure 22: UML Diagrams Overview

The use case diagram belongs to the category of behavioral diagrams which describe general types of behavior. In fact, a use case diagram is a graphic representation of the interactions between the different elements of a system. This methodology is usually used in system analysis phase to identify and organize the system requirements. In this context, the word "system" refers to the anti-theft car tracking system that is being developed.

The use case diagram bellow represents the interaction between the different actors of the system including the end user, the administrator and the GPS tracker, and identifies the different system requirements.

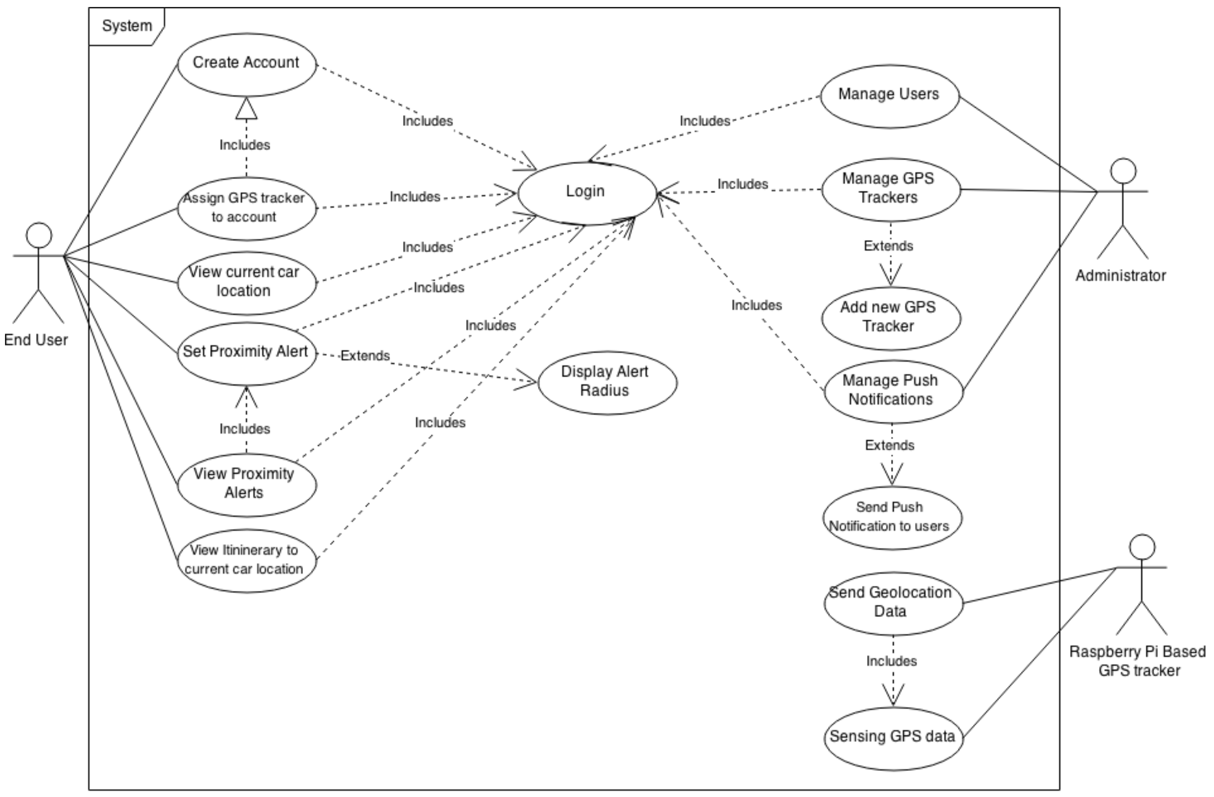


Figure 23: Anti-theft Car Tracking System UML Diagram

6.2.2 USE CASE SPECIFICATIONS

In order to use the functionality of a use case, and reflect the sequential use case dependencies, we documented each functionality with preconditions, post conditions, actors involved in the use case, and a description of the requirement.

6.2.2.1 LOGIN

Use case name:	Login
Use case ID:	001
Description:	End users and the administrator log in the system by providing their username and password.
Actors:	End User, Administrator
Precondition:	System is Idle, showing welcome message and a login interface
Main flow:	1. The Use Case starts when a user or administrator selects user selects login. 2. The user or administrator enters the login username and password. 3. The user or administrator is logged in the system.
Post conditions:	1. The user or administrator is logged in the system

6.2.2.2 CREATE ACCOUNT

Use case name:	Create account
Use case ID:	002
Actors:	End User
Description:	A user will be able to create an account by proving required information including username, email address, password and IMEI of the GPS tracker.
Precondition:	A valid IMEI of the GPS tracker must be provided by the end user
Main Flow:	1. This use case is required when a new request of creating a new account is received. 2. The user starts this use case by clicking on create account on the menu.

	3. The user provides a valid IMEI of a GPS tracker. 4. The user fills the required fields of the creation form of the desired account.
Post conditions:	1. Three tabs will be displayed. The user will have the choice to view current car location, set proximity alerts or display itinerary to current car location.

6.2.2.3 VIEW CURRENT CAR LOCATION

Use case name:	View current car location
Use case ID:	003
Description:	End user views the current location of their car
Actor:	End user
Precondition:	System is Idle, showing an option to display current car location
Main flow:	1. The Use Case starts when the end user selects view current car location. 2. The end user selects to view current car location. 3. Live positioning of the car is displayed as the car moves
Post conditions:	1. Current position of the car is displayed as the car moves

6.2.2.4 ASSIGN GPS TRACKER TO ACCOUNT

Use case name:	Assign GPS tracker to account
Use case ID:	004
Description:	The end user assigns a GPS tracker to his account
Actor:	End User
Precondition:	User has chosen to register a new account
Main flow:	1. The use case starts when a user chooses to register

	2. User enters the GPS tracker IMEI 3. GPS tracker IMEI is validated. 4. User selects and click the register button . 5. The GPS tracker is assigned to the newly created account
Post conditions:	1. GPS tracker is assigned to an account and inserted in the database.

6.2.2.5 SET PROXIMITY ALERT

Use case name:	Set Proximity Alert
Use case ID:	005
Description:	The user sets a proximity alert
Actor:	End User
Precondition:	The system is idle, showing an option to set a proximity alert.
Main flow:	1. The use case starts when a user selects the proximity alert tab. 2. The user enter an alert radius. 3. The user selects the enable proximity alerts button 4. The Proximity Alert is set and enabled 5. A confirmation message pops up.
Post conditions:	1. Whenever the car leaves the alert radius the user will receive a push notification and an alarm will be beeping until the car is back inside the alert radius.

6.2.2.6 DISPLAY ALERT RADIUS

Use case name:	Display Alert Radius
Use case ID:	006
Description:	An alert Radius is displayed on the map
Actor:	End User
Precondition:	The user sets a proximity alert
Main flow:	1. The use case starts when a user sets a proximity alert 2. The user selects the enable proximity alerts button 3. An alert Radius is displayed on the map
Post conditions:	1. An alert Radius is displayed on the map

6.2.2.7 VIEW PROXIMITY ALERTS

Use case name:	View proximity alerts
Use case ID:	007
Description:	End user views the proximity alerts
Actor:	End user
Precondition:	System is Idle, showing an option to display the list of proximity alerts
Main flow:	1. The Use Case starts when the end user selects display proximity alerts 2. The end user selects to display the proximity alerts. 3. The list of previous proximity alerts are displayed
Post conditions:	1. The list of previous proximity alerts are displayed

6.2.2.8 VIEW ITINERARY TO CURRENT CAR LOCATION

Use case name:	View itinerary to current car location
Use case ID:	008
Description:	End user views itinerary to current car location
Actor:	End user
Precondition:	System is Idle, showing an option to display the itinerary to current car location
Main flow:	1. The Use Case starts when the end user selects display itinerary to current car location 2. The end user selects to display the itinerary to current car location 3. The itinerary to current car location is computed and displayed
Post conditions:	1. The itinerary to current car location is computed and displayed

6.2.2.9 MANAGE USERS

Use case name:	Manage users
Use case ID:	009
Actors:	Administrator
Description:	The Administrator will be able to delete or modify the information of the users already added in the system.
Preconditions:	A request must be made to edit or delete the user.
Main Flow:	1. This use case is required when a new request of editing or deleting a user is received. 2. The administrator starts this use case by selecting then clicking

	on Edit or Delete link on the users list. 3. The agent confirms the action.
Post conditions:	1. The profile of the recently edited user will be changed

6.2.2.10 MANAGE GPS TRACKERS

Use case name:	Manage GPS trackers
Use case ID:	010
Actors:	Administrator
Description:	The Administrator will be able to add, delete GPS trackers to the system
Preconditions:	A request must be made to add or delete a GPS tracker.
Main Flow:	3. This use case is required when a new request of adding or deleting a GPS tracker is received. 4. The administrator starts this use case by selecting then clicking on add or delete link on the GPS trackers list. 3. The administrator confirms the action.
Post conditions:	1. The GPS tracked will be either added to the system or removed from the system

6.2.2.11 ADD GPS TRACKER

Use case name:	Add GPS tracker
Use case ID:	011
Actors:	Administrator

Description:	The Administrator will be able to add GPS trackers to the system
Preconditions:	The Administrator selects manage GPS trackers
Main Flow:	1. This use case is required when a new request of adding a GPS tracker is received. 2. The administrator starts this use case by selecting then clicking on add button on the GPS trackers list. 3. The administrator confirms the action.
Post conditions:	1. The GPS tracked will be added to the system

6.2.2.12 MANAGE PUSH NOTIFICATIONS

Use case name:	Manage Push Notifications
Use case ID:	012
Actors:	Administrator
Description:	The Administrator will be able to manage push notifications
Preconditions:	A request must be made to manage push notification
Main Flow:	1. This use case is required when a new request of managing push notifications is received 2. The administrator starts this use case by selecting manage push notifications 3. 3. The administrator confirms the action
Post conditions:	1. A menu to view or send push notifications is displayed

6.2.2.13 SEND PUSH NOTIFICATIONS

Use case name:	Send push notifications
Use case ID:	013

Actors:	Administrator
Description:	The Administrator will be able to send a push notification to a specific user or to all users
Preconditions:	A request must be made to send Push notifications
Main Flow:	1. This use case is required when a new request of sending push notifications is received 2. The administrator starts this use case by entering the message to be sent 3. The administrator selects the target user(s) to which the message will be sent 4. 3. The administrator confirms the action.
Post conditions:	1. A push notification is sent to the target user(s) and is displayed on their device

6.2.2.14 SEND GEOLOCATION DATA

Use case name:	Send Geolocation Data
Use case ID:	014
Actors:	Raspberry-Pi based GPS tracker
Description:	The GPS tracker sends its geolocation data to the system at each interval of time
Preconditions:	The GPS tracker is turned on.
Main Flow:	1. This use case is when the GPS tracker is turned on 2. After every interval of time the Raspberry-Pi based GPS tracker sends its geolocation data including its longitude, latitude, altitude and speed.

Post conditions:	1. Geolocation data will be intercepted by a Web server and stored in a MySQL database

6.2.2.15 SENSE GPS DATA

Use case name:	Sense GPS Data
Use case ID:	015
Actors:	Raspberry-Pi based GPS tracker
Description:	The GPS tracker senses its geolocation data using a Ublox based GPS receiver and a GPS antenna
Preconditions:	The GPS tracker is turned on.
Main Flow:	1. This use case is when the GPS tracker is turned on 2. After every interval of time the Raspberry-Pi based GPS tracker will sense its geolocation data including its longitude, latitude, altitude and speed as well as other geolocation data
Post conditions:	1. Geolocation data will stored in an array for further uses such as sending geolocation data.

6.2.3 TECHNOLOGY ENABLERS

The technology enablers are defined as a set of features and technology that will be used in the implementation of the application. The table bellow describes the set of software technology enablers we will be using in our project and lists them by "Name" and "Description".

Table 6: Software Technology Enablers

Name	Desciption
PHP	PHP is a server-side scripting language that was designed for web development, however, it is also used as a general-purpose programming language. In our project, it is used to read POST HTTP requests sent by Raspberry Pi, and store the GPS data in a MySQL database. It is also also to generate a JSON file containing the GPS data related to the current position of a car.
MySQL Database	MySQL is the most widely used open-source relational database management system, and used the Structured Query Language SQL. In this project we will be working with MySQL. In this project, it is used to store information about the car including GPS coordinates, moving speed and followed itinerary.
HTML & CSS	HyperText Markup Language, or HTML, is the standard markup language used for web pages creation. Cascading Style Sheets also called CSS, are usually used with HTML in order to define the layout and the look of text and other elements. These languages are used to format our web pages.
JavaScript	JavaScript is a dynamic programming language that allows client-side scripts to interact with users or communicate asynchronously with a server. This language is mainly used to include and manipulate the GPS MAP and live tracking as well as security alerts radiuses.

AJAX	AJAX or Asynchronous JavaScript and XML, is a new way of exchanging data with a server, and updating parts of a web page without reloading the whole page. In this project, it is used to asynchronously download GPS data from a generated XML file.
Python	Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.
Shell Programming	Shell programs are programs designed to be run by the Linux shell, a command line interpreter. They are used to configure network interface (3G Modem), and the GPS shield, install all necessary libraries and programs to correctly read data from the GPS module and correctly send it through the 3G Modem, and finally automate the process of sending GPS data to the Web Server.
Java	Java is a object-oriented programming language that is class-based and that was specifically designed to have only a small number of implementation dependencies. In our project it is used along with XML to design our Android application.
JSON	JSON or JavaScript Object Notation, is an open standard format that has a simple readable structure, and that serves in transmitting data between a server and web application. In this project JSON file will be generated using a PHP program and will contains GPS data such

	as longitude, latitude, altitude and speed.
XML	XML or Extensible Markup Language is a markup language that defines a set of rules to encode documents. In our case, it is used to design UI layouts and screen elements constituting our Android application.
Google Maps Android API v2	This application programming interface is used by Android developers for embedding maps into an activity as a fragment using a simple XML code.
Google Maps JavaScript API v3	This application programming interface is used by web developers and especially JavaScript familiar users for embedding Google maps into web pages using JavaScript.
Parse	Parse.com is a framework that provides backend services to developers. Since our Android application requires a backend on the internet to login, register, and/or recover forgotten passwords, we chose to work with the Parse Android SDK . In fact Parse also offers server maintenance, push notifications management, cloud data storage and easy social integration.

6.2.4 DATABASE DESIGN

After defining the main modules using the UML diagram, we designed our MySQL database using the following SQL queries.

```
CREATE SCHEMA `gps_data` ;

CREATE TABLE `gps_data`.`account` (
  `username` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `account_type` VARCHAR(3) NOT NULL,
  PRIMARY KEY (`username`));
```

```sql

CREATE TABLE `gps_data`.`driver` (
`driver_cin` VARCHAR(8) NOT NULL,
`driver_last_name` VARCHAR(45) NOT NULL,
`driver_first_name` VARCHAR(45) NOT NULL,
`driver_phone` INT NOT NULL,
PRIMARY KEY (`driver_cin`)
ADD CONSTRAINT `username_fk`
FOREIGN KEY (`username`)
REFERENCES `gps_data`.`account` (`username`)
ON DELETE NO ACTION
ON UPDATE NO ACTION);

CREATE TABLE `gps_data`.`car` (
`car_plate_number` VARCHAR(10) NOT NULL,
`car_brand` VARCHAR(45) NOT NULL,
`car_model` VARCHAR(45) NOT NULL,
`car_driver` VARCHAR(45) NOT NULL,
`zone_id` INT NOT NULL,

PRIMARY KEY (`car_plate_number`),
INDEX `driver_fk_idx` (`car_driver` ASC),
CONSTRAINT `driver_fk`
  FOREIGN KEY (`car_driver`)
  REFERENCES `gps_data`.`driver` (`driver_cin`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION);

  CREATE TABLE `gps_data`.`zone` (
`zone_id` INT NOT NULL,
`description` VARCHAR(45) NOT NULL,
`latitude` DOUBLE NULL,
`longitude` DOUBLE NULL,
`radius` DOUBLE NULL,
PRIMARY KEY (`zone_id`));

CREATE TABLE `gps_data`.`proximity alert` (
`alert_id` INT NOT NULL,
`alert_time` DATETIME NOT NULL,
`car_plate_number` VARCHAR(45) NOT NULL,
`zone_id` INT NULL,
PRIMARY KEY (`alert_id`),
INDEX `car_plate_numberfk_idx` (`car_plate_number` ASC),
INDEX `zone_idfk_idx` (`zone_id` ASC),
CONSTRAINT `car_plate_numberfk`
  FOREIGN KEY (`car_plate_number`)
  REFERENCES `gps_data`.`car` (`car_plate_number`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION);

```

6.2.5 ENTITY RELATION DIAGRAM

After defining the main modules using the UML diagram, we designed our MySQL database and generated the following entity relation diagram.

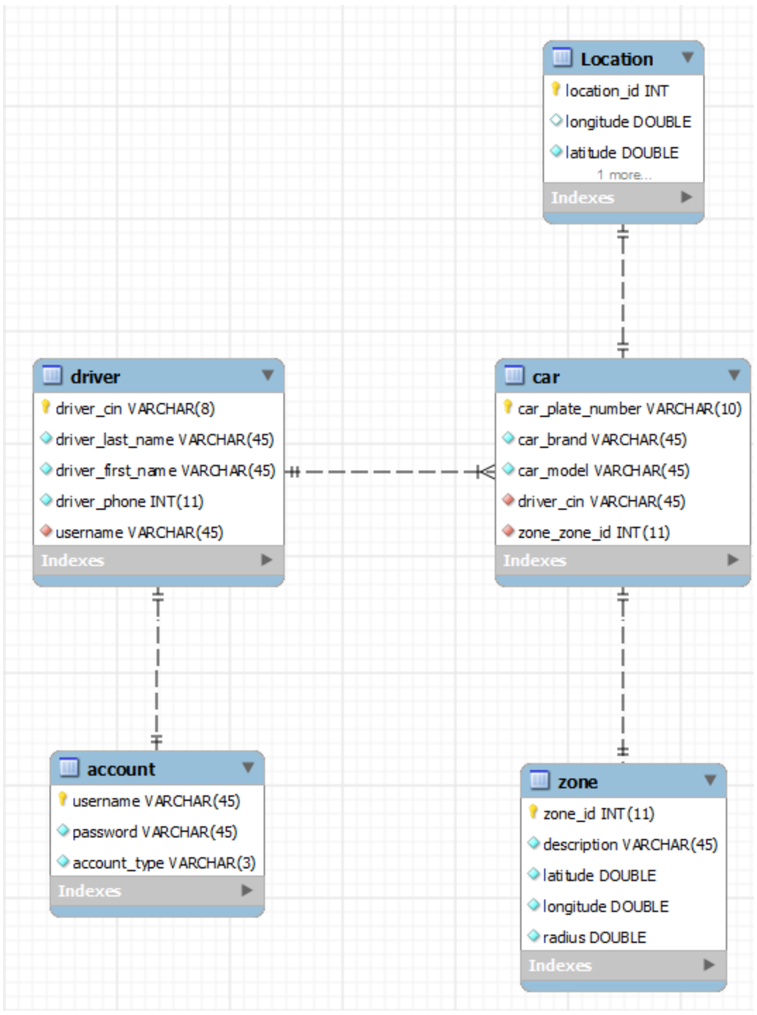


Figure 24: Entity Relation Diagram

6.3 SOFTWARE IMPLEMENTATION

6.3.1 WEB APPLICATION

6.3.1.1 HANDLING GPS TRACKER REQUESTS

6.3.1.1.1 READING SENT GPS DATA

This PHP program serves to connect to our MySQL database using our server adress, database name, database username and database password. This program will be called by the second one.

```php
<?php

    function Connection(){
        $server="fdb4.awardspace.net";
        $user="1246684_2371";
        $pass="xxxxxxxxxxxx";
        $db="1246684_2371";

        $connection = mysql_connect($server, $user, $pass);

        if (!$connection) {
            die('MySQL ERROR: ' . mysql_error());
        }

        mysql_select_db($db) or die( 'MySQL ERROR: '. mysql_error() );

        return $connection;
    }
?>
```

Whenever the GPS tracker sends HTTP requests, this program will intercept these requests and store the received variables into local variables.

```php
<?php

    $link=Connection();

    $longitude=$_POST["long"];
    $latitude=$_POST["lat"];

    header("Location: index.php");
?>
```

6.3.1.1.2 STORING GPS DATA IN DATABASE

After reading the GPS variables and storing into local variables, we attempt to connect to our MySQL database, and on success, we store the received variables in their corresponding table columns. This MySQL Database contains a table called "markers" that contains a set of variables, longitude, the altitude, the moving speed as well as other positioning data as well as the exact time the data was received.

```php
<?php
    include("connect.php");

    $link=Connection();

    $longitude=$_POST["long"];
    $latitude=$_POST["lat"];


    $query = "INSERT INTO `markers` (`latitude`,`longitude`)
        VALUES ('".$longitude."','".$latitude."')";

    mysql_query($query,$link);
    mysql_close($link);

    header("Location: index.php");
?>
```

6.3.1.2 CREATING A MAP USING GOOGLE MAPS API JAVASCRIPT API V3

This is the code provide by Google to create and display a map using the Google Maps JavaScript API V3.

```
<!DOCTYPE html>
<html>
<head>
<script src="http://maps.googleapis.com/maps/api/js"></script>
<script>
function initialize() {
  var mapProp = {
    center:new google.maps.LatLng(51.508742,-0.120850),
    zoom:5,
    mapTypeId:google.maps.MapTypeId.ROADMAP
  };
  var map=new google.maps.Map(document.getElementById("googleMap"),mapProp)
;
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>

</html>
```

6.3.1.3 IMPLEMENTING THE TRACKING FEATURE

In order to implement the tracking feature, we used two main functions, one is called load(), and creates the new Google by initializing it with an initial longitude and latitude. The second main function is called loader(), and is run every second. The loader() function is responsible for updating the position of the marker in the map. In this function, we implement another function called downloadUrl("phpsqlajax_genxml3.php", function(data)) in which the first parameter represents the XML file to be downloaded which is in our case generated by the php file "phpsqlajax_genxml3.php", and the second parameter represent the function to be called on success. This function the AJAX functionality to download the current tracker position every second and updates the position of the marker without refreshing the whole page. This following figure [11] illustrates how AJAX works:

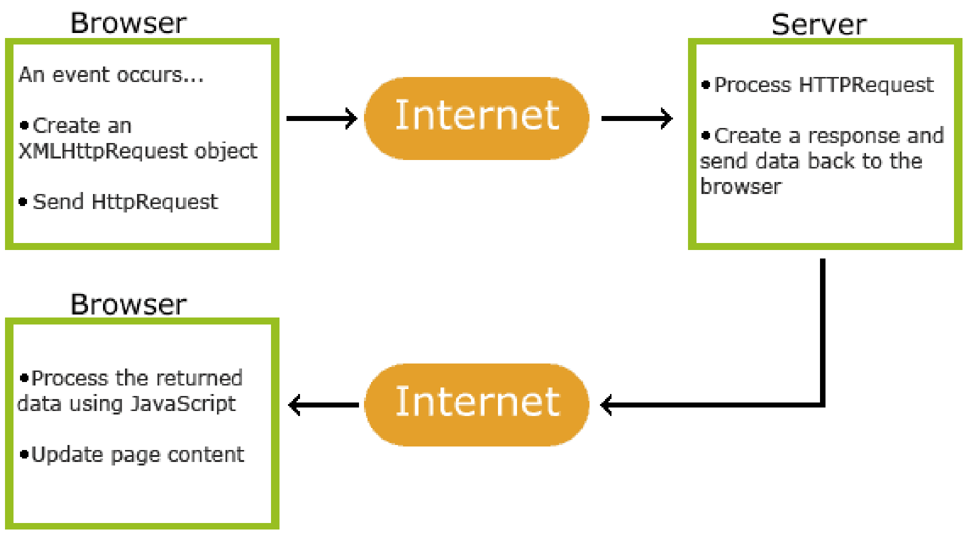

Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

Figure 25: How AJAX works

After calling downloadUrl function, the XML file name is passed and the retrieved text in stored in a variable called "data". The function "data.responseXML" is called to get the response data from the Web server as an XML data. Finally, the function xml.document Element.getElementsByTagName("marker") is used get an array of all records containing the tag "marker". Each record illustrates a marker, which has specific attributes. In order to access a marker attribute, we use the function .getAttribute("name"), name being the attribute we would like to get and we store it in a variable.

```
downloadUrl("phpsqlajax_genxml3.php", function(data) {
       var xml = data.responseXML;
       var markers = xml.documentElement.getElementsByTagName("marker");
       //for (var i = 0; i < markers.length; i++) {
         var i=0;
         var name = markers[i].getAttribute("name");
         var address = markers[i].getAttribute("address");
         var type = markers[i].getAttribute("type");
         var point = new google.maps.LatLng(
             parseFloat(markers[i].getAttribute("lat")),
             parseFloat(markers[i].getAttribute("lng")));
       array[j]=new google.maps.LatLng(markers[i].getAttribute("lat"),markers[i].getAttribute("lng"));
       jj=j+1;

       if (marker==null) {

       // If the marker does not exist we create it
       marker = new google.maps.Marker({
       map: map,
       position: point,
       icon: image,
           });

       }

       else {
       marker.setPosition(point);
       map.panTo(point);
       }

       bindInfoWindow(marker, map, infoWindow, html);

      });

     }
```

A complete code of the live tracking implementation can be found in Annex A.

By using an XML file as an intermediary between my database and my Google Map allows a faster initial page load, a more flexible map application, and an easier debugging. A table called "markers" contains information about the actual position of the car including the "lat" column which stands for latitude and the lng column which stands for longitude.

```sql
UPDATE markers SET lat=33.542155 ,lng=-5.105209 WHERE id=0;
SELECT SLEEP(3);

UPDATE markers SET lat=33.542071 ,lng=-5.105263 WHERE id=0;
SELECT SLEEP(3);

UPDATE markers SET lat=33.540676 ,lng=-5.105821 WHERE id=0;
SELECT SLEEP(3);

UPDATE markers SET lat=33.541077 ,lng=-5.105799 WHERE id=0;
SELECT SLEEP(3);

UPDATE markers SET lat=33.542071 ,lng=-5.105263 WHERE id=0;
SELECT SLEEP(3);

UPDATE markers SET lat=33.540676 ,lng=-5.105821 WHERE id=0;
SELECT SLEEP(3);
```

For simulation purposes, we executed a set of SQL queries shown in the figure above into a MySQL database management system, and we noticed that whenever positioning data changes, the map gets automatically updated, and the marker gets to its new position instantaneously as shown on the figure bellow.

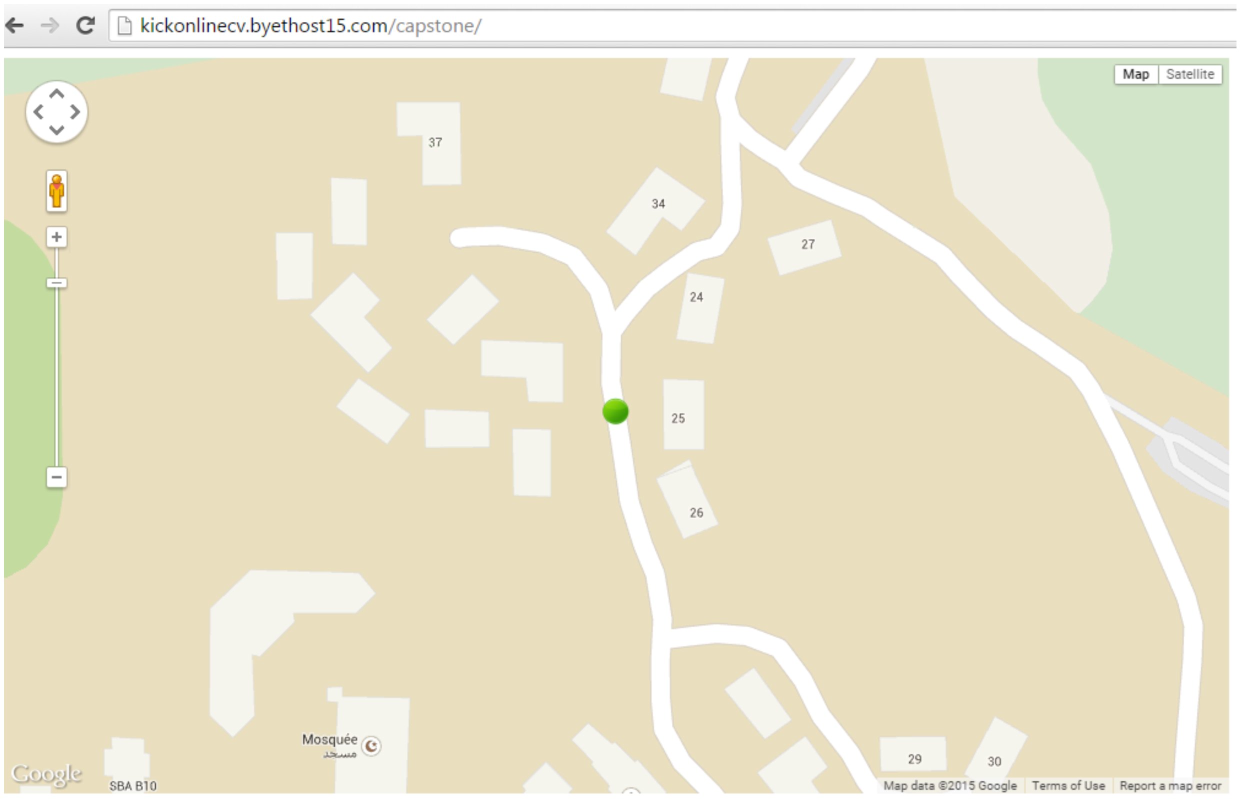


Figure 26: Screenshot of our web application

6.3.2 ANDROID APPLICATION

6.3.2.1 ANDROID OPERATING SYSTEM

Android is an operating system, currently developed by Google, which is based on the Linux kernel. It often described as a stack of software components divided into four main layers as shown in the architecture diagram below. The bottom layer is the Linux Kernel and provides basic system functionality such as memory management and/or process management. On the top of the Linux Layer, we can find the set of libraries used by the operating system. Furthermore, the application layer provides various higher level services to applications.

These services are provided in the form of Java Classes. Finally, the top layer represents the application to be developed.

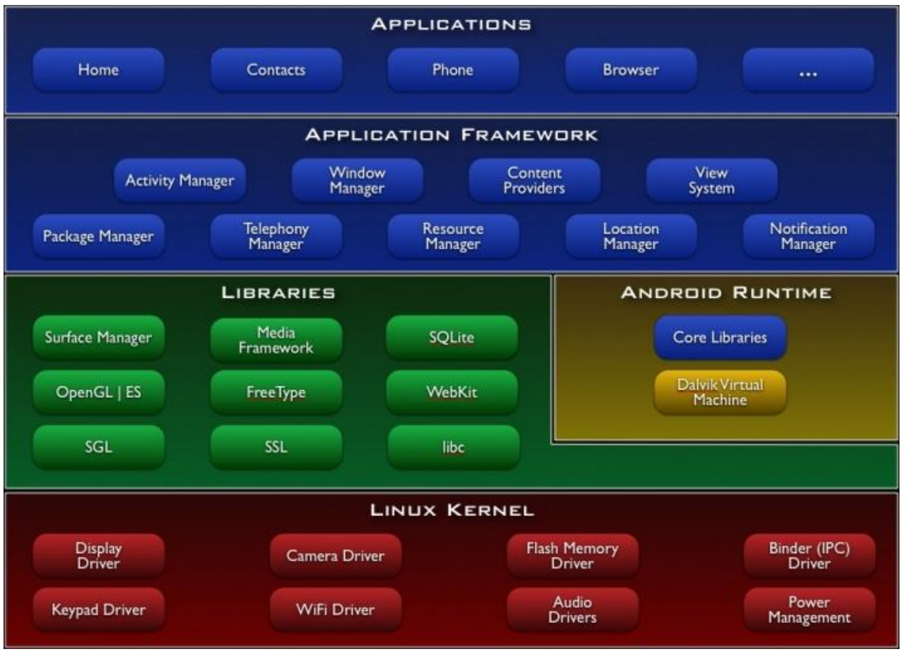


Figure 27: Android Architecture Diagram [12]

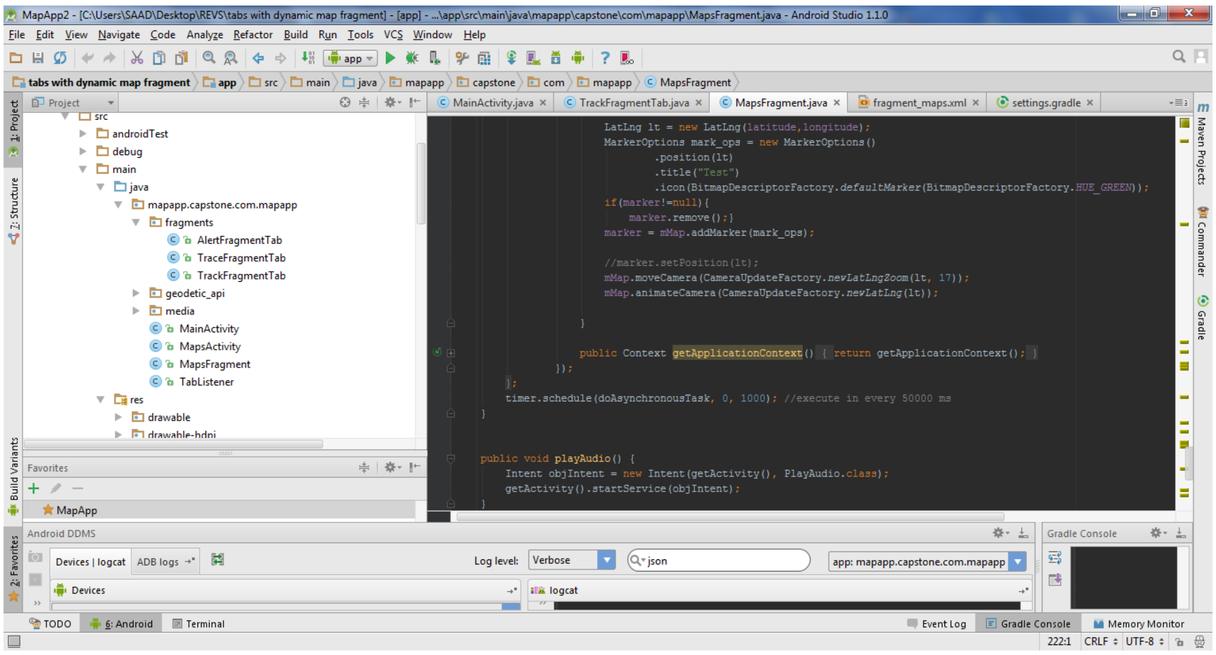
In our project, we will mainly work on the application framework to make use of its services in our Android application. Indeed we will be using the Activity Manager as well as the Notification Manager.

An Activity is an application component providing a screen for users to interact with the operating system to perform actions such as dialing a phone or viewing a map. Usually, each activity uses a window that fills the screen in order to draw its user interface. Typically, applications consist of multiple activities interconnected to each other, one of which is the main activity that can start other activities to perform other actions.

In an activity, the user interface can be divided into multiple portions called fragments. Each fragment represents a modular section of the activity, and each fragment has its own life cycle. Therefore, it has its own input events and acts as a sub-activity in the system.

6.3.2.2 ANDROID STUDIO

Android Studio is the official Integrated Development Environment for Android application development. We have chosen this IDE for the development of our mobile application.



```
                LatLng lt = new LatLng(latitude,longitude);
                MarkerOptions mark_ops = new MarkerOptions()
                        .position(lt)
                        .title("Test")
                        .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));
                if(marker!=null){
                    marker.remove();}
                marker = mMap.addMarker(mark_ops);

                //marker.setPosition(lt);
                mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(lt, 17));
                mMap.animateCamera(CameraUpdateFactory.newLatLng(lt));

            }

            public Context getApplicationContext() { return getApplicationContext(); }
        });
    };
    timer.schedule(doAsynchronousTask, 0, 1000); //execute in every 50000 ms
}


public void playAudio() {
    Intent objIntent = new Intent(getActivity(), PlayAudio.class);
    getActivity().startService(objIntent);
}
```

Figure 28: Screenshot of Android Studio IDE

6.3.2.3 MAIN ACTIVITY AND FRAGMENT ACTIVTY IMPLEMENTATION

This class represents our main activity which extends the ActionBarActivity for our application to use the default action bar, and in which we declare our three main fragment: Tracking fragment, Alert Fragment and Tracing fragment.

```
public class MainActivity extends ActionBarActivity {


    // We declare our tabs and fragments.
    // ActionBar.Tab trackTab, tracking, traceTab, alertTab;
    ActionBar.Tab tracking, traceTab, alertTab;

        Fragment alertFragmentTab = new AlertFragmentTab();
        Fragment traceFragmentTab = new TraceFragmentTab();
        Fragment trackingFragmentTab = new MapsFragment();

    @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            ActionBar actionBar = getSupportActionBar();
```

```java

            actionBar.setDisplayShowHomeEnabled(false);

            actionBar.setDisplayShowTitleEnabled(false);

            actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

            alertTab = actionBar.newTab().setIcon(R.drawable.alert_icon);
            traceTab = actionBar.newTab().setIcon(R.drawable.trace_icon);
            tracking = actionBar.newTab().setIcon(R.drawable.track_icon);

            // Tab listeners.
            alertTab.setTabListener(new TabListener(alertFragmentTab));
            traceTab.setTabListener(new TabListener(traceFragmentTab));
            tracking.setTabListener(new TabListener(trackingFragmentTab));

            // Tabs to the ActionBar.
            actionBar.addTab(tracking);
            actionBar.addTab(alertTab);
            actionBar.addTab(traceTab);

        }

    public boolean onCreateOptionsMenu(Menu menu) {

        MenuInflater inflater = getMenuInflater();

        inflater.inflate(R.menu.menu_bottom, menu);

        return true;

    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle item selection
        switch (item.getItemId()) {
            case R.id.about:
                showAbout();
                return true;
            case R.id.exit:
                finish();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    }
```

The following code represents the Tracking fragment, one of our three fragments. Once this fragment is called, its onCreateView method is invoked and its corresponding XML layout is inflated.

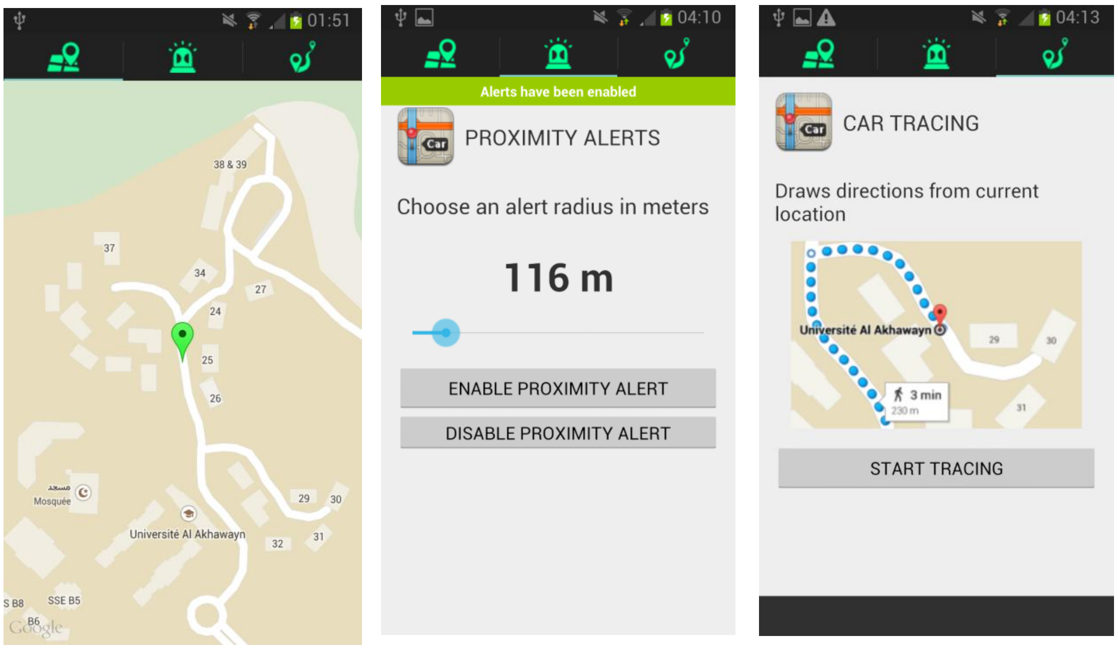
```java
package mapapp.capstone.com.mapapp.fragments;

import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import mapapp.capstone.com.mapapp.MapsActivity;
import mapapp.capstone.com.mapapp.R;

public class TrackFragmentTab extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.track_layout, container, false);

        Intent firstpage= new Intent(getActivity(),MapsActivity.class);
        getActivity().startActivity(firstpage);

        return rootView;
    }
}
```

6.3.2.4 MAP FRAGMENT IMPLEMENTATION USING GOOGLE MAPS ANDROID API V2

We first obtained an API key from Google developers website "https://console.developers.google.com/project" in order to be able to use the Google Maps Android API V2. Moreover, once the Fragment is initialized, its corresponding XML layout is inflated. In the case of our tracking fragment, the following layout is the one we used to implement the map.

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <com.google.android.gms.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
</RelativeLayout>
```

The following figure illustrates our three main fragments embedded into three separate tabs: the first one is the tracking fragment, the second one is for the proximity alerts and the last one is the tracing fragment.



Choose an alert radius in meters

Draws directions from current location

Figure 29: Anti-theft Car Tracking System Android Fragments

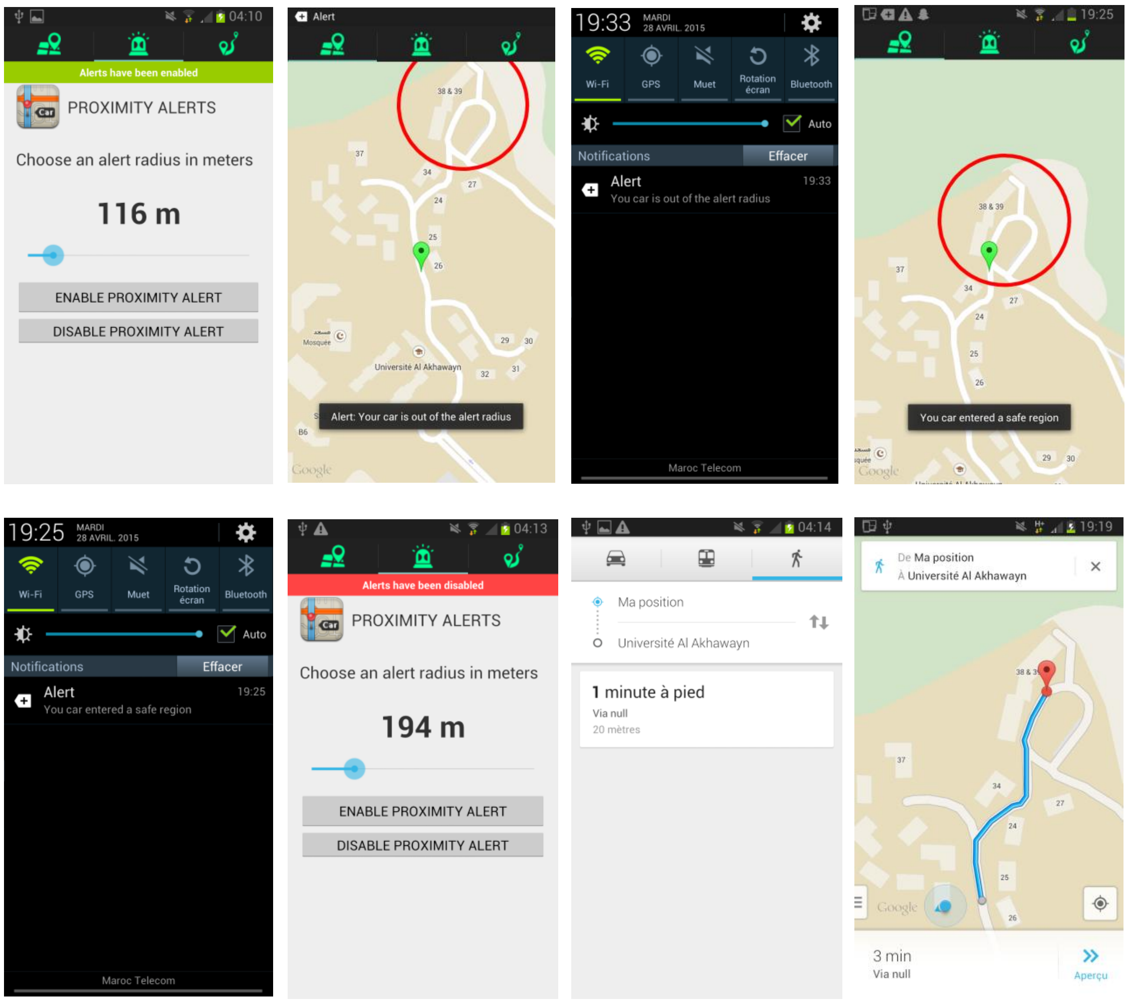
6.3.2.5 *IMPLEMENTING TRACKING, TRACING AND PROXIMITY ALERT FEATURES*

Now that we created our map, we changed the tracking fragment to support live tracking and proximity alerts. Indeed, once the fragment is opened, a Google map is created, and a marker is added to an initial position. Since we are supposed to have 3 main tabs, one for tracking, one for alerts and one for tracing, we will need to have computations to be made on background thread. This is why we use the class AsyncTask. In the following program, it is used to asynchronously download GSON file using an HTTP request, and it is also used to determining whether a car is inside an alert radius or not, using the Geodetic API which accounts for the curvature of earth while computing the distance between the car and the center of the alert radius. Indeed, whenever the car moves outside of a predefined alert radius, the user will receive a Push notification alerting him that his car is moving out of the radius, and an alarm will be beeping until the car goes back to a safe radius.

```java
    public class downloadtask extends AsyncTask<String, Void, String> {

        StringBuilder str = new StringBuilder();

        @Override
```

```
      protected String doInBackground(String... urls) {

        try {

         final String urll = "http://www.afakwebmaster.dx.am/test/genjson.php";


         URL url = new URL(urll);

          BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));
          String line;

          while ((line = in.readLine()) != null) {
            str.append(line);
          }
          in.close();

          setString(str.toString());

        } catch (MalformedURLException e) {
          System.out.println("Malformed URL: " + e.getMessage());
        } catch (IOException e) {
          System.out.println("I/O Error: " + e.getMessage());
        }

        return str.toString();

      }

      @Override
      protected void onPostExecute(String result) {
      }
    }
```

A complete code of the Live Tracking, Tracing and Proximity Alerts can be found in Annex B. The following figures illustrate the three main anti-theft features implemented in our project: the live tracking, the proximity alerts and the tracing.


Alerts have been enabled

PROXIMITY ALERTS

Choose an alert radius in meters

116 m

Alert: Your car is out of the alert radius

Alert — You car is out of the alert radius — 19:33

You car entered a safe region

Alert — You car entered a safe region — 19:25

Alerts have been disabled

PROXIMITY ALERTS

Choose an alert radius in meters

194 m

Ma position

Université Al Akhawayn

1 minute à pied

Via null

20 mètres

De Ma position

À Université Al Akhawayn

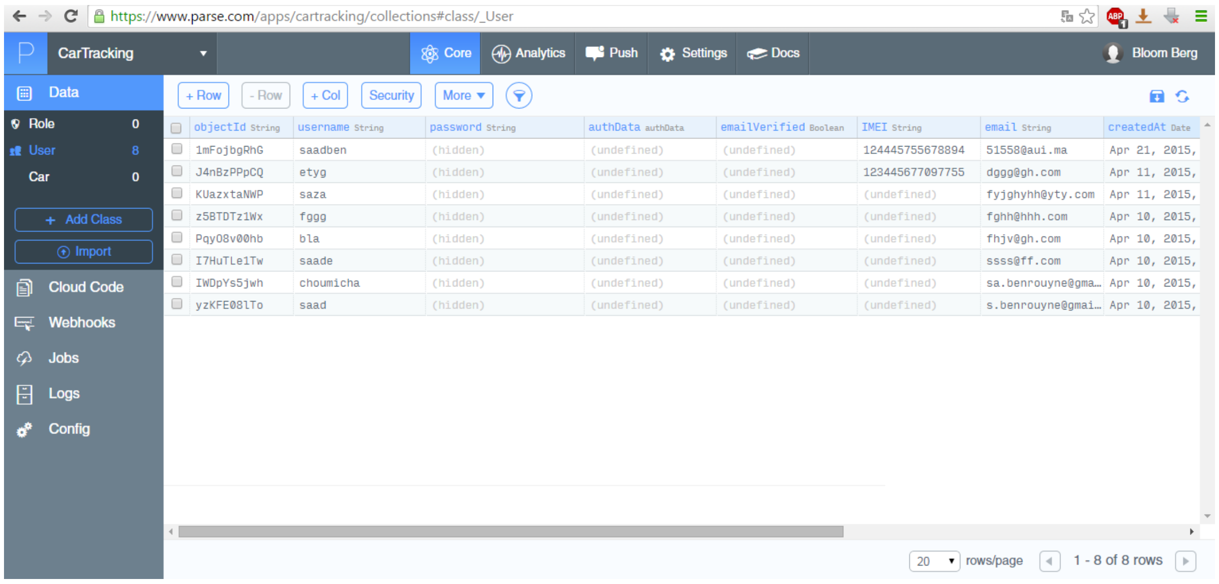
3 min

Via null

Figure 30: Tracking, tracing and proximity alerts features

6.3.2.6 USER AUTHENTICATION USING PARSE

Parse.com is a framework that provides backend services to developers. Since our Android application requires a backend on the internet to login, register, and/or recover forgotten passwords, we chose to work with the Parse Android SDK . In fact Parse also offers server maintenance, push notifications management, cloud data storage and easy social integration. In our project, once the user tries to login Parse checks whether the username is in its Cloud storage and whether the password is valid. This is what Parse interface looks like.



objectId String	username String	password String	authData authData	emailVerified Boolean	IMEI String	email String	createdAt Date
1mFojbgRhG	saadben	(hidden)	(undefined)	(undefined)	124445755678894	51558@aui.ma	Apr 21, 2015,
J4nBzPPpCQ	etyg	(hidden)	(undefined)	(undefined)	123445677097755	dggg@gh.com	Apr 11, 2015,
KUazxtaNWP	saza	(hidden)	(undefined)	(undefined)	(undefined)	fyjghyhh@yty.com	Apr 11, 2015,
z5BTDTz1Wx	fggg	(hidden)	(undefined)	(undefined)	(undefined)	fghh@hhh.com	Apr 10, 2015,
PqyO8v00hb	bla	(hidden)	(undefined)	(undefined)	(undefined)	fhjv@gh.com	Apr 10, 2015,
I7HuTLe1Tw	saade	(hidden)	(undefined)	(undefined)	(undefined)	ssss@ff.com	Apr 10, 2015,
IWDpYs5jwh	choumicha	(hidden)	(undefined)	(undefined)	(undefined)	sa.benrouyne@gma…	Apr 10, 2015,
yzKFE08lTo	saad	(hidden)	(undefined)	(undefined)	(undefined)	s.benrouyne@gmai…	Apr 10, 2015,

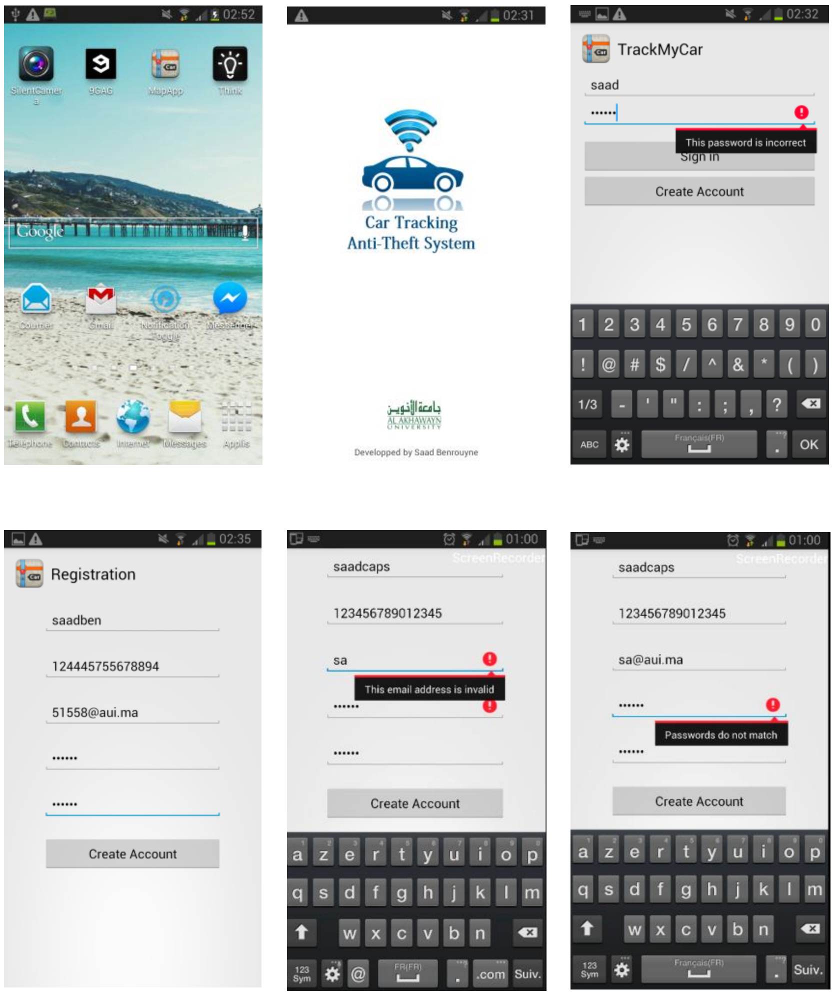
Figure 31: Parse Core Interface

The following code shows how simple a client registration can be implemented using the Parse Android SDK.

```java
ParseUser user = new ParseUser();
user.setUsername("Saad");
user.setPassword("capstone");
user.setEmail("s.benrouyne@aui.ma");

user.put("phone", "053-586-2222");

user.signUpInBackground(new SignUpCallback() {
  public void done(ParseException e) {
    if (e == null) {
    } else {
    }
  }
});
```

Using additional features provided by Parse Android SDK such as "Login" and "Password Recovery", we have been able to develop the registration, login and password recovery as seen on the following figure. After installing our application on an Android smartphone, an icon referring to the application is added to the desktop. In the registration view, valid and unique username, email and IMEI must be entered before validating the form. Indeed, Parse asynchronously creates a new user after making sure that both username and email are unique.

After validating the data, Parse securely hashes the password in the cloud and never stores it in plaintext.



saad

This password is incorrect

saadben

124445755678894

51558@aui.ma

saadcaps

123456789012345

sa

This email address is invalid

sa@aui.ma

Passwords do not match

Figure 32: Registration and Login Views

7 CONCLUSION

Developing this system, was a successful experience for me and a great accomplishment, as it allowed me not only to brush up a great deal of skills I developed during my studies in computer science, but also discover many other exciting technologies and learn how to use them in my project. Moreover, by working with a professional supervisor who supported me during my capstone project by giving me feedback and tips, I learned how to be responsible of a certain task, although being constrained by a deadline.

Designing and Implementing a completely working GPS tracker was the first challenge I took in this project since I had no previous experience in hardware design related to embedded systems. Indeed, my engineering curiosity to know how hardware devices work and my enthusiasm to develop a tangible product that could positively impact the society, got the best of me and led me to work on this project.

After successfully addressing my first challenge, I decided to work on the software part of the project, which involved developing a web application as well as an Android application, where my main goal was to be able to track in real time a car using the GPS tracker built before. Confidently, I managed to implement the live tracking feature in both applications, and I also managed to add the proximity alert feature in the Android application which addresses the problem of car theft in my Anti-theft car tracking system.

Finally working on this project helped me define what skills and what knowledge I need to improve in the future, and inspired me to explore more career opportunities either related to web design, Android design or hardware design.

8 REFERENCES

[1] Mohamed Z, *Près de 2000 voitures volées par an, au Maroc* [Online]. Available: http://www.yabiladi.com/article-societe-2475.html

[2] Raspberry Pi, *What is Raspberry Pi* [Online]. Available: https://www.raspberrypi.org/help/what-is-a-raspberry-pi/

[3] *PEST Analysis* [Online]. Available: http://en.wikipedia.org/

[4] Bladi.net, *Le Roi Mohammed VI ordonne au ministre de l'Intérieur de combattre le crime au Maroc* [Online]. Available: http://www.bladi.net/roi-mohammed-6-crime-maroc.html

[5] LaVieEco, *Laissez votre employeur vous licencier* [Online]. Available: http://www.lavieeco.com/news/connaissez-vos-droits-maroc/laissez-votre-employeur-vous-licencier--17904.html

[6] [7] Mohamed Z, *Près de 2000 voitures volées par an, au Maroc* [Online]. Available: http://www.yabiladi.com/article-societe-2475.html

[8] HA Proxy, [Online]. Available: http://www.haproxy.org/

[9] Dan Mandle, *Getting gpsd to work with Python* [Online]. Available:

http://www.danmandle.com/blog/getting-gpsd-to-work-with-python/

[10] TechTarger, *What is Unified Modeling language ?* [Online]. Available:

http://searchsoftwarequality.techtarget.com/definition/Unified-Modeling-Language

[11] W3Schools, *AJAX Introduction* [Online]. Available: http://www.w3schools.com/ajax/ajax_intro.asp

[12] Android Developers, [Online]. Available: http://developer.android.com/images/system-architecture.jpg

9 ANNEX A: LIVE TRACKING IMPLEMENTATION USING AJAX

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps AJAX + mySQL/PHP Example</title>
    <script src="http://maps.google.com/maps/api/js?sensor=false"
            type="text/javascript"></script>
    <script type="text/javascript">

    setInterval("loader();",1000);

    var marker = null;
    var map;
    var infoWindow;
    var j =0;

    function load() {
      map = new google.maps.Map(document.getElementById("map"), {
        center: new google.maps.LatLng(47.6145, -122.3418),
        zoom: 18,
        mapTypeId: 'hybrid'
      });
      infoWindow = new google.maps.InfoWindow;
        loader();

      }

      function loader() {

var array = [];


      downloadUrl("phpsqlajax_genxml3.php", function(data) {
        var xml = data.responseXML;
        var markers = xml.documentElement.getElementsByTagName("marker");
        //for (var i = 0; i < markers.length; i++) {
          var i=0;
          var name = markers[i].getAttribute("name");
          var address = markers[i].getAttribute("address");
          var type = markers[i].getAttribute("type");
          var point = new google.maps.LatLng(
              parseFloat(markers[i].getAttribute("lat")),
              parseFloat(markers[i].getAttribute("lng")));
        array[j]=new google.maps.LatLng(markers[i].getAttribute("lat"),mark
ers[i].getAttribute("lng"));
        jj=j+1;
              var html = "<b>" + name + "</b> <br/>" + address;
          var icon = customIcons[type] || {};


var image = new google.maps.MarkerImage(
    "vani.ico",
    null, /* size is determined at runtime */
    null, /* origin is 0,0 */
    null, /* anchor is bottom center of the scaled image */
    new google.maps.Size(64, 64)
);

          if (marker==null) {
      // If the marker does not exist we create it
        marker = new google.maps.Marker({
        map: map,
        position: point,
        icon: image,
            });

        }

        else {
        marker.setPosition(point);
        map.panTo(point);
        }

        bindInfoWindow(marker, map, infoWindow, html);

      });

    }


    function bindInfoWindow(marker, map, infoWindow, html) {
      google.maps.event.addListener(marker, 'click', function() {
        infoWindow.setContent(html);
        infoWindow.open(map, marker);
      });
    }

    function downloadUrl(url, callback) {
      var request = window.ActiveXObject ?
          new ActiveXObject('Microsoft.XMLHTTP') :
          new XMLHttpRequest;

      request.onreadystatechange = function() {
        if (request.readyState == 4) {
          request.onreadystatechange = doNothing;
          callback(request, request.status);
        }
      };

      request.open('GET', url, true);
      request.send(null);
          }

          function doNothing() {}

        </script>
        </head>

        <body onload="load()">
          <div id="map" style="width: 1000px; height: 600px"></div>
        </body>
      </html>
```

10 ANNEX B: LIVE TRACKING, TRACING AND PROXIMITY ALERTS

```java
    public class downloadtask extends AsyncTask<String, Void, String> {

        StringBuilder str = new StringBuilder();

        @Override
        protected String doInBackground(String... urls) {

            try {

              final String urll = "http://www.afakwebmaster.dx.am/test/genjson.php";


              URL url = new URL(urll);

                BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));
                String line;

                while ((line = in.readLine()) != null) {
                    str.append(line);
                }
                in.close();

                setString(str.toString());

            } catch (MalformedURLException e) {
                System.out.println("Malformed URL: " + e.getMessage());
            } catch (IOException e) {
                System.out.println("I/O Error: " + e.getMessage());
            }

            return str.toString();

        }

        @Override
        protected void onPostExecute(String result) {
        }
    }

    public void callAsynchronousTask() {
        final Handler handler = new Handler();
        Timer timer = new Timer();
        TimerTask doAsynchronousTask = new TimerTask() {
            @Override
            public void run() {
                handler.post(new Runnable() {
                    public void run() {
                        try {

                            downloadtask down = new downloadtask();
                            down.execute().get();

                            System.out.println(tslim);

                            JSONObject object = new JSONObject(tslim);
                            latitude = object.getDouble("latitude");
                            longitude = object.getDouble("longitude");

                            System.out.println(latitude);
                            System.out.println(longitude);

                            double distance = (distances(latitude, longitude, center_alert_lat, center_alert_long));
                            if (distance <= 100) {
                                if (in_circle==false) {
                                    Toast.makeText(getActivity(), "You car entered a safe region", Toast.LENGTH_SHORT).show();
                                    notifying(getView(),true);
                                    in_circle=true;
                                    stopAudio();
                                    //System.out.println(distance + " Meters\n");
                                }
                            }else{
                                if (in_circle==true) {
                                    Toast.makeText(getActivity(), "Alert: Your car is out of the alert radius", Toast.LENGTH_SHORT).show();
                                    notifying(getView(),false);
                                    in_circle=false;
                                    playAudio();
                                    //System.out.println(distance + " Meters\n");
                                }
                            }
                        } catch (NullPointerException ex) {
                            ex.printStackTrace();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        } catch (ExecutionException e) {
                            e.printStackTrace();
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }

                        LatLng lt = new LatLng(latitude,longitude);
                        MarkerOptions mark_ops = new MarkerOptions()
                                .position(lt)
                                .title("Car")
                                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));
                        if(marker!=null){
                            marker.remove();}
                        marker = mMap.addMarker(mark_ops);

                        //marker.setPosition(lt);
                        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(lt, 17));
                              mMap.animateCamera(CameraUpdateFactory.newLatLng(lt));
                          }

                      });
                  }
              };
              timer.schedule(doAsynchronousTask, 0, 1000); //execute in every 1000 ms
          }


          public void playAudio() {
              Intent objIntent = new Intent(getActivity(), PlayAudio.class);
              getActivity().startService(objIntent);
          }

          public void stopAudio() {
              Intent objIntent = new Intent(getActivity(), PlayAudio.class);
              getActivity().stopService(objIntent);
          }
```

```java

           @SuppressWarnings("deprecation")
           public void notifying(View vobj, boolean in_circle){
               NM=(NotificationManager)getActivity().getSystemService(Context.NOTIFICATION_SERVICE);
               Notification notify=new Notification(android.R.drawable.stat_notify_more,"Alert",System.currentTimeMillis());
               PendingIntent pending=PendingIntent.getActivity(getActivity().getApplicationContext(),0, new Intent(),0);
               if(in_circle==true){
                   notify.setLatestEventInfo(getActivity().getApplicationContext(),"Alert","You car entered a safe region",pending);
               }else{
                   notify.setLatestEventInfo(getActivity().getApplicationContext(),"Alert","You car is out of the alert radius",pending);
               }
               NM.notify(0, notify);
           }

       }
```