Part E Human Robot Interface

In this part, the design and development principals of the two iOS mobile applications, which have been implemented in the context of this project, are presented. Both applications aim to enhance the user's experience in the field of human robot interface. The first is related to the visualisation of the radiation mapping and the second to the teleoperation of a UAS.

E.1 UAV teleoperation application

One of the popular and traditional methods of teleoperating a quad copter is using a Remote Control (RC). It is a well-established and tested method, which has its origin before the era of the UAV. Nowadays, advanced UAVs are equipped with many sensors and specific purpose equipment such as cameras or radiation spectrometers. An alternative method of getting high bandwidth data from a UAV, which is also convenient, is using telemetry. This method uses a client application, installed in a laptop, in order to make all the data transmitted from the UAV available.

The approach, which has been developed for this project, is based on the utilisation of smart devices (fig.E.1). These devices are equipped with powerful CPUs, extreme visualisation capabilities and a wide range of connectivity options. Needless to say that it is very popular and almost everyone in the developed countries has one of them.

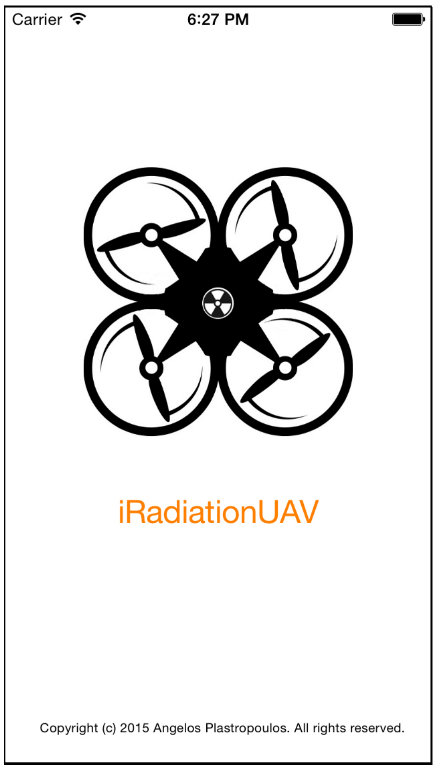

Carrier
6:27 PM
iRadiationUAV
Copyright (c) 2015 Angelos Plastropoulos. All rights reserved.

Fig.E.1 The splash screen of the teleoperation application

E.1.1 Topology

The communication between the iPhone and the UAV is accomplished via wireless connection (Wi-Fi). I have chosen this method because TCP/IP is the native communication protocol supported in ROS ecosystem and a wireless connection will be available in the next generation AARM system since it is going to have a companion computer.

Since I chose the communication method the next step was to decide the format in which the data should be encoded in and exchanged between peers. The selection of JSON (Javascript Object Notation) is a commonly accepted method. JSON is a lightweight data interchange format, easy for people to write and read yet, at the same time, it is easy for machines to parse and generate. In addition, there is a wide variety of available libraries that can be utilised to provide apparent and reliable implementations.

A. UAV Peer

In the side of the UAV there is one important requirement that has to be satisfied: UAV should be ROS-enabled. This is not restrictive because ROS is a very popular and trustworthy action plan in robotics' communities. Moreover, as it has been implemented in this project for safety and convenience reasons, it is possible to use a simulator for the development. The advantage here is that in ROS systems the same code can be used with a real robot without any change.

In a ROS system there is a messaging mechanism called topics (fig.E.2). The data from the installed on-board devices which are publicised in relevant topics and control commands related to the linear and angular velocities are also accepted from the appropriate receivers using associated topics accordingly. These topics should be available to the iPhone in order to have access to the data and control commands. At this point a specific ROS package comes to play. *Rosbridge* (Crick et al., 2011) which provides a JSON API to ROS functionality for non-ROS programs. This sounds ideal for the requirements of my project. *Rosbridge* provides a wide range of supporting interfaces. For my case, it was important that it included a WebSocket server for web browsers' interaction. The WebSocket Protocol is a network protocol designed for the web that enables browsers to have bi-directional communication with a server. Using this package, communication mechanisms can be provided to the non-ROS systems following a ROS-style approach. It supports topic publishing and subscribing, service request and response as serialized JSON objects.

One of the advanced equipment that a UAV could carry is a camera. It is extremely helpful and interesting to have real-time streaming of the video to the app. While *Rosbridge* allows streaming videos, seeing as it is just another message type from ROS, I preferred to use a different component. *MJPEG* (Motion JPEG) server (Pitzer, 2011), which is a ROS package, optimised for efficiently transferring stream images in binary format. Consequently, an additional communication channel was used to increase performance and enable clients to request specific video quality and frame size according to their needs.

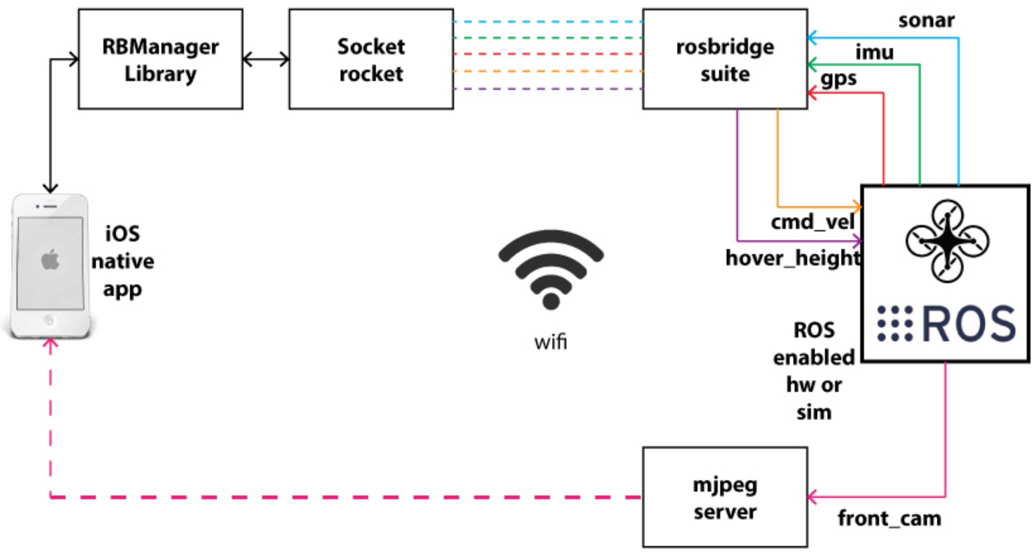


Fig.E.2 Topology of the teleoperation application

B. iOS device peer

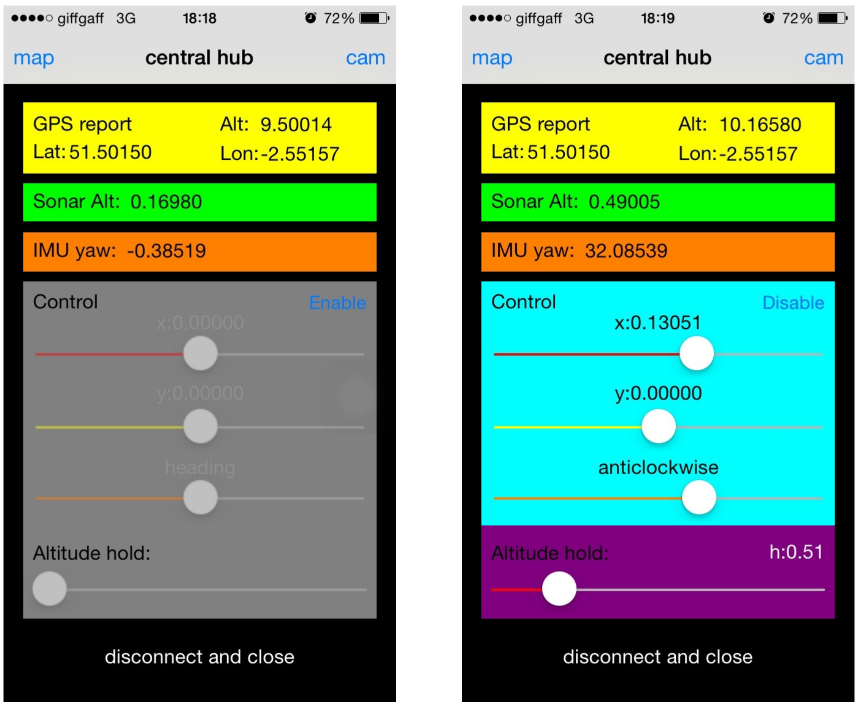
On the other side there is a native iOS application developed using Objective C. As with the ROS side, the application should leverage the appropriate libraries to gain access in the data generated in the ROS ecosystem as well as to send compatible data. For this case, a *WebSocket* library should be used. *SocketRocket* is an objective C WebSocket client library (Lewis, 2015) useful in native mobile applications. By using WebSockets, existing tools built for the web can be easily reused for the mobile applications. It is similar to how most mobile apps reuse HTTP for their APIs. One level deeper in the architecture, there is the powerful *RBManager* library (Goodhoofd, 2014) for communicating between iOS and ROS. This library is a wrapper for the *SocketRocket* WebSocket library in iOS. This approach follows the standard publisher, subscriber and service call architecture to send packets compatible with the *Rosbridge* protocol. It is well structured and easily expandable. It allows adding new message types and building on top of that. It has a simple syntax as long as the developer knows ROS.

Regarding the processing of the streaming video the case is simpler. The stream of the MJPEG server is essentially a video sequence in which each frame is separately compressed as a JPEG image. In the iOS library there is a *UIWebView* component (Developer.apple.com, 2015) available that enables the embedding of web content in iOS native applications.

Specific implementation details regarding the application are described in the related appendices chapter.

E.1.2 Functionality

The teleoperation application is a prototype of a fully working application yet has not optimised in areas such as functionality and appearance to be commercially available. The application consists of three main screens. The first screen appears when the application is loaded as the central hub (fig.E.3). From this point, the operator has a complete overview of the sensors' feedback. It shows data collected from the GPS unit, the IMU unit and the sonar, which measures the distance from the ground. In addition, in this screen the operator can enable the control tab. This tab provides the ability to control the linear velocity in the plane, x and y direction, as well as change the heading by manipulating the angular velocity in the z direction. This tab includes also the altitude lock mode. More precisely, the user can adjust the constant distance from the ground in which the UAV flies. Whenever the operators need to freeze the UAV (zero the velocities components of the control), they just have to toggle the enable/disable button. In this procedure the altitude remains constant. Finally, it contains a button to disconnect from the UAV and to terminate the application.



●●●●○ giffgaff 3G 18:18 72%

map central hub cam

GPS report Alt: 9.50014
Lat: 51.50150 Lon: -2.55157

Sonar Alt: 0.16980

IMU yaw: -0.38519

Control Enable
x:0.00000
y:0.00000
heading

Altitude hold:

disconnect and close

●●●●○ giffgaff 3G 18:19 72%

map central hub cam

GPS report Alt: 10.16580
Lat: 51.50150 Lon: -2.55157

Sonar Alt: 0.49005

IMU yaw: 32.08539

Control Disable
x:0.13051
y:0.00000
anticlockwise

Altitude hold: h:0.51

disconnect and close

Fig.E.3 The central hub screen with control tab enabled and disabled

The second screen shows the exact location of the UAV in a map (fig.E.4). It takes the longitude and the latitude from the on-board GPS unit. The third screen shows the streaming video of the on-board front camera (fig.E.4). The user having locked the altitude could also drive the UAV from this screen.

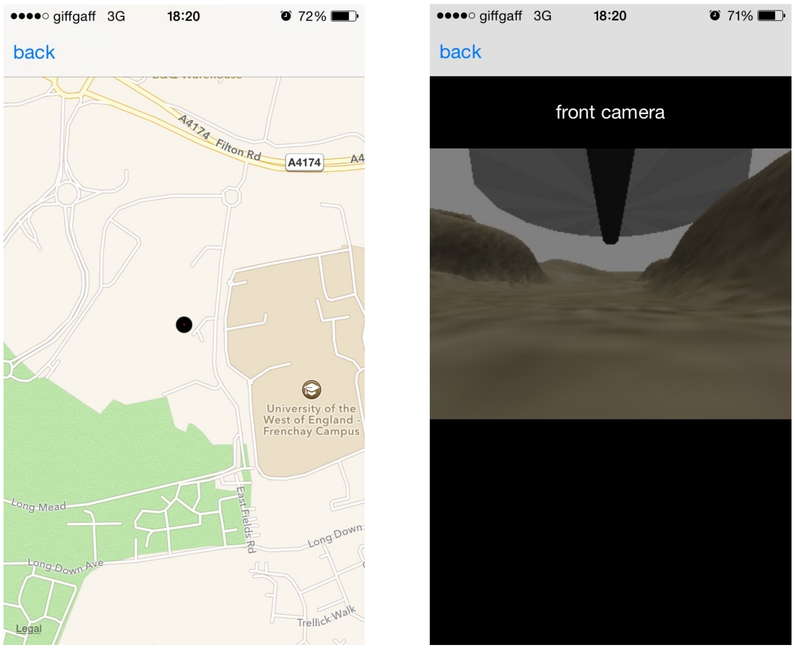

●●●●○ giffgaff 3G 18:20 72%
back
A4174 Filton Rd
A4174
University of the West of England - Frenchay Campus
Long Mead
Long Down Ave
East Fields Rd
Long Down
Trellick Walk
Legal
●●●●○ giffgaff 3G 18:20 71%
back
front camera

Fig.E.4 The map and the camera screen

More technical details about the implementation of the application can be found on the “Mobile applications Configurations” section in the appendices.

E.2 Radiation mapping application

This mobile application constitutes an important part of this project since it is directly related to the radiation mapping visualisation. Up until now the radiation mapping is a procedure which is conducted offline. To illustrate, the operator could carry a hand held measurement device that stores the data in a SD memory card. Upon the completion of the procedure, the researcher should access this memory card by inserting it in a memory reader using a laptop. The desired requirement at this point is that the researcher will be able to get the status of the radiation measurement unit and the measurements displayed on a map during the procedure in real-time (fig.E.5).

Carrier
6:48 PM
iRadiationApp
IAC
Copyright (c) 2015 Interface Analysis Centre. All rights reserved.

Fig.E.5 The splash screen of the radiation mapping application

E.2.1 Topology

The communication between the iPhone and the measurement unit is accomplished via Bluetooth low energy (LE). BLE is a wireless personal area network technology. It provides considerably reduced power consumption and reduced cost while maintaining a similar communication range with the classic Bluetooth. In addition, using BLE I avoided the process of requesting a specific license from Apple, called MFi ("Made for iPhone/iPad/iPod"), which is required when the classic Bluetooth is employed.

The operation of the measurement is based on an Arduino IDE sketch developed at Interface Analysis Centre (IAC). This sketch was used as a base to build the server-part of my application. The architecture of the sketch, as it was built, was to enable the GPS connection to get the longitude and latitude and then to obtain cps measurements via the USB protocol from the spectrometer. When all the required data were available, the sketch was forming a data entry to the text file stored in the SD card. My task was to maintain the working part while simultaneously integrating the required parts for the iOS application.

The first addition for the measurement unit was to install the BLE shield. After that, the sketch had to change in order to access desired bits of information, form compound packets and send them to the link. The formation of specific data packets was a demand since the payload of the transmitted package has a maximum size of 20 bytes for the gives BLE shield (fig.E.6)

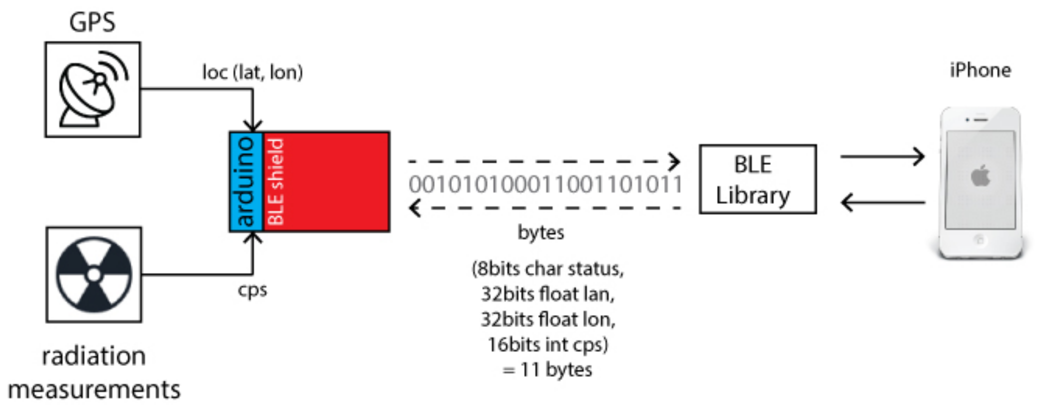


Fig.E.6 The overview of the radiation mapping scenario

According to the application's specifications the status of the GPS should be reported. To save space in the messages payload, I encoded the status information in one byte. Consequently, since the GPS lock is an on-off condition, it needs one bit. The remaining 7 bits of the byte are used to report the number of connected satellites. After that, the latitude and the longitude should be represented using 32bits floating-point numbers. To manage this, I break each value to four parts of byte size and I assemble them again in the iPhone side. Moreover, the radiation counts per second is encoded as a two bytes integer. Finally, having formed the payload of the BLE message (fig.E.7), this message is transmitted to the other side at the end of every cycle.

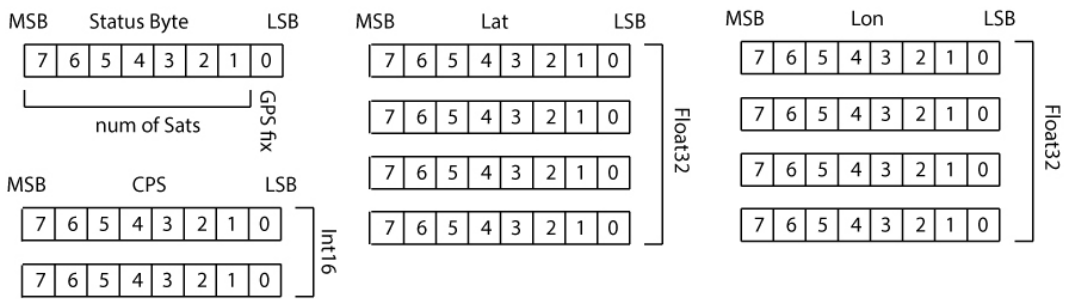
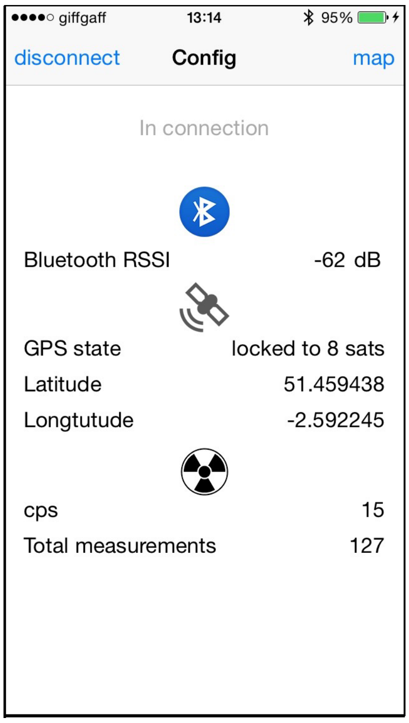


Fig.E.7 The payload of the BLE message

On the other side there is an iPhone device with the Bluetooth connection enabled. In order to decode the transmitted messages, a specific library is needed. The manufacturer of the BLE shield has released an appropriate library for this task, called BLE. Using the provided features, the iPhone can search for compatible BLE devices to connect and disconnect with them. When the iPhone has connected to the measurement unit, there is the possibility to measure the RSSI level (Received Signal Strength Indication) and access the contents of the payload. Using the reverse procedure, the iPhone script assembles the primitive data types from the raw bytes and visualises the information using various means.

E.2.2 Functionality

The radiation mapping application has been developed using the directions of the experienced IAC researchers. It is a very specific application and in order to be useful and a real tool in the hands of the user, it has to be specifically defined and designed according to their needs.

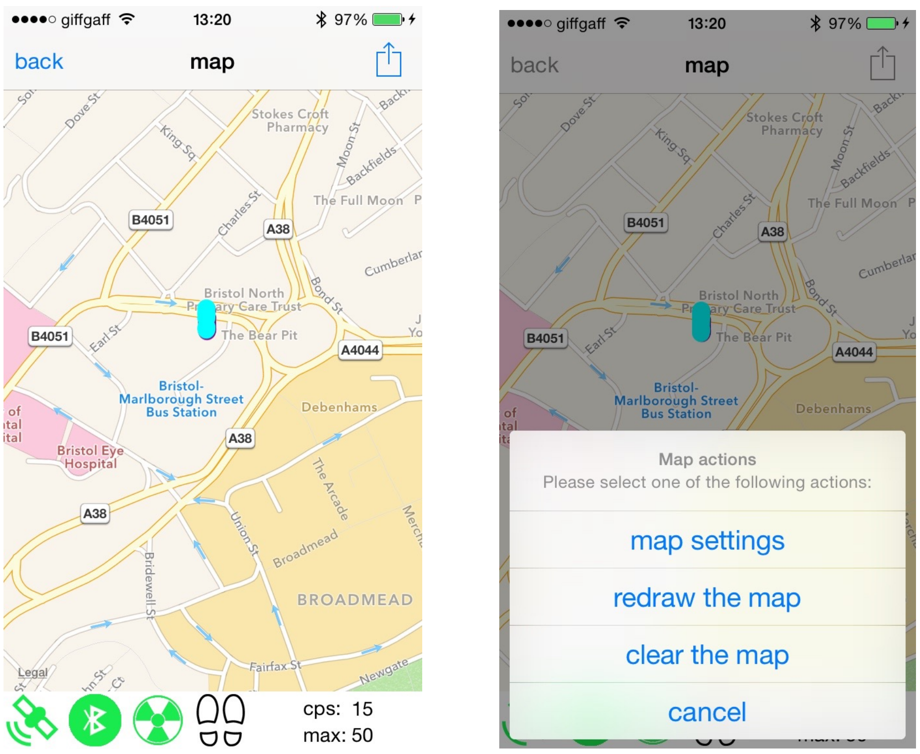


In connection

Bluetooth RSSI	-62 dB
GPS state	locked to 8 sats
Latitude	51.459438
Longtutude	-2.592245
cps	15
Total measurements	127

Fig.E.8 The configuration screen of the radiation mapping application

The configuration screen displays all the useful information to the user (fig.E.8). In the fist part of the screen there is a message board that displays informative messages regarding the Bluetooth connection. In this display, the user can be informed about the availability of other BLE devices, the name of the device that they are eventually connected to and other relevant information. After a successful connection the RSSI value is continuously updated to reflect the signal strength of the Bluetooth connection between the measurement unit and the iPhone. At this point, with an active Bluetooth connection, the user can read the status of the measurement device in the iPhone screen. Consequently, when the GPS unit locks, both the relevant message and the number of the connected satellites will be displayed to the screen as a sign of the signal strength. Finally, there is a section related to data which come from the radiation spectrometer. In this part, the actual CPS value is reported together with total number of measurements that has been performed.



cps: 15
max: 50

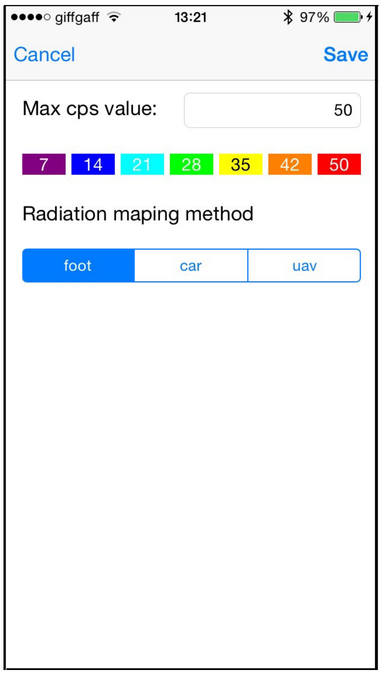
Map actions
Please select one of the following actions:

Fig.E.9 The map screen (left) and the map screen with the actions button enabled (right)

The next screen displays the visual representation of the measurements in a map (fig.E.9). Every time a new measurement is received, the application stores this value in an internal data structure and at the same time it displays a colour-encoded point at the exact longitude and latitude in the map. Colour depends on the CPS value. At the lower part of the application there is an information tab. Since the users spend most of the time on this screen, they need a way to know the overall status of the procedure. The summary part, which is available, contains the following:

- Three status icons corresponding to GPS, Bluetooth and radiation device connectivity. The first two are green when the connection is active, otherwise they are grey. The third one is animated like a heart pulse to indicate that the CPS storing and visualisation proceeds flawlessly.
- There is an additional icon in the tab that represents the method of the radiation mapping. Furthermore, there are two character strings displaying information about the current CPS value and the visual colour-encoded CPS range.

There is a button in the navigation part of the screen, which contains a set of map related tasks. Using this button the user can clear and redraw the map. An additionally important task is the display of the map settings screen.



Max cps value: 50

7	14	21	28	35	42	50

Radiation maping method

[✓ foot] car uav

Fig.E.10 The map settings screen of the radiation mapping application

In the map settings screen the user can adjust various visualisation parameters (fig.E.10). Firstly, there is the option to select the radiation mapping which is used to perform the measurements. There are three methods available, measurements taken on foot, by UAV or using a car. According to the selection, the zoom level of the map is changed to allow a reasonable display detail level. Moreover, in this point an attenuation factor can be added since there is a different sensitivity level if the user takes measurements inside of a car or on foot. Secondly, the user can adjust the colour encoding of the measurements. More precisely, the red colour represents measurements close to 500 CPS and purple close 0. In cases where the area is considered safe the colour will remain purple without the ability to represent minor fluctuations of 10 to 20 CPS difference. Thus, in this screen the user is able to define the maximum CPS value and based on this value the CPS range is rescaled accordingly. As a result, with a maximum of 50 CPS the range of each colour will be approximately 7 instead of the initially 71 for the 500 CPS. All in all, when the user returns to the map the measurements are redrawn reflecting the new colour range.

More technical details about the implementation of the application can be found on the "Mobile Applications Configurations" section in the appendices.